Department of Computer Science
Series of Publications A
Report A-2014-4

Continuous Context Inference on Mobile Platforms

Sourav Bhattacharya

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIV, University Main Building, on August 25th, 2014, at 12 o'clock noon.

University of Helsinki
Finland

Supervisors

Dr. Petteri Nurmi, University of Helsinki, Finland
Dr. Patrik Floréen, University of Helsinki, Finland

Pre-examiners

Dr. Nicholas Lane, Microsoft Research Asia
Prof. Dr.-Ing. Jörg Ott, Aalto University, Finland

Opponent

Prof. Antonio Krüger, German Research Center for Artificial Intelligence (DFKI) and Saarland University, Germany

Custos

Prof. Petri Myllymäki, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/
Telephone: +358 294 1911, telefax: +358 9 191 51120

ISSN 1238-8645
ISBN 978-951-51-0046-7 (paperback)
ISBN 978-951-51-0047-4 (PDF)
Computing Reviews (1998) Classification: G.3, H.3.3, I.2.6, I.5.2
Helsinki 2014
Unigrafia

Continuous Context Inference on Mobile Platforms

Sourav Bhattacharya

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
sourav.bhattacharya@cs.helsinki.fi
http://www.cs.helsinki.fi/sourav.bhattacharya

PhD Thesis, Series of Publications A, Report A-2014-4
Helsinki, June 2014, x + 94 + 67 pages
ISSN 1238-8645
ISBN 978-951-51-0046-7 (paperback)
ISBN 978-951-51-0047-4 (PDF)

Abstract

In this thesis we develop novel methods for continuous and sustained context inference on mobile platforms. We address challenges present in real-world deployment of two popular context recognition tasks within ubiquitous computing and mobile sensing, namely localization and activity recognition. In the first part of the thesis, we provide a new localization algorithm for mobile devices using the existing GSM communication infrastructures, and then propose a solution for energy-efficient and robust tracking on mobile devices that are equipped with sensors such as GPS, compass, and accelerometer. In the second part of the thesis we propose a novel sparse-coding-based activity recognition framework that mitigates the time-consuming and costly bootstrapping process of activity recognizers employing supervised learning. The framework uses a vast amount of unlabeled data to automatically learn a sensor data representation through a set of extracted characteristic patterns and generalizes well across activity domains and sensor modalities.

Computing Reviews (1998) Categories and Subject Descriptors:

G.3 Probability and Statistics: Probabilistic algorithms
H.3.3 Information Search and Retrieval: Clustering
I.2.6 Learning: Knowledge acquisition
I.5.2 Design Methodology: Classifier design and evaluation

General Terms:
Algorithms, context, and machine learning.

Additional Key Words and Phrases:
Context-awareness, location-awareness, mobile platforms, radio map, fingerprinting, position and trajectory tracking, energy-efficiency and robustness trade-off, activity recognition, machine learning, pervasive computing, and ubiquitous computing.

Acknowledgements

I am grateful to my supervisors Dr. Petteri Nurmi and Dr. Patrik Floréen for their advice, encouragement, and support during the entire duration of my PhD studies. I also thank them for their efforts in managing the Adaptive Computing group, where I had the pleasure of working since early 2007. Special thanks to Petteri for providing guidance in the field of mobile sensing and making it possible to work with the most talented researchers in the field.

I would like to thank the Department of Computer Science of the University of Helsinki and Helsinki Institute for Information Technology HIIT for providing excellent computing facilities, office space and sports facilities. I gratefully acknowledge financial support from the Department of Computer Science, Future Internet Graduate School (FIGS), Helsinki Graduate School in Computer Science and Engineering (Hecse), Doctoral programme in Computer Science (DoCS) and the Foundation of Nokia Corporation. I sincerely acknowledge the financial support from the Secure Systems group, headed by Prof. N. Asokan, during writing of this thesis. I sincerely thank Prof. Petri Myllymäki and Prof. Esko Ukkonen for their constant encouragements and providing an excellent working environment.

I thank my pre-examiners Dr. Nicholas Lane and Prof. Jörg Ott for their time, comments, and suggestions. I extend my sincere gratitude to Dr. Mikkel Baun Kjærgaard, Dr. Henrik Blunck, Dr. Thomas Plötz, Dr. Santi Phithakkitnukoon, and Dr. Arto Klami for excellent research collaborations and great mentorship during my PhD studies. Special thanks to Mikkel, Henrik, and Thomas for being great hosts during my stay in Aarhus, Denmark and in Newcastle, UK. In addition to my supervisors and collaborators, I also like to thank Caroline Werner, Teemu Pulkkinen, Samuli Hemminki, and Marina Kurtén for providing valuable feedback on this thesis.

I warmly thank the present and past fellow members of the Adaptive Computing group, especially Teemu Pulkkinen, Samuli Hemminki, Joel Pyykkö, Tony Kovanen, Yina Ye, Haipeng Guo, Taneli Vähäkangas, Andreas Forsblom, Jara Uitto, Joonas Kukkonen, Eemil Lagerspetz and Yiyun Shen for making research fun. I have had countless inspiring discussions with them on science and on random topics. Especially our discussions over Pulla (Finnish sweet roll) have been always refreshing, irrespective of the weather outside. Among my friends in Finland special thanks go to Priyotosh Bandyopadhyay, Sayantan Hore, Kirtiman Ghosh, Ankur Sinha, Indrajit Chaudhuri, Debjani Chadhuri and Ramkrishna Ghosh for such a great time. I also thank Arijit Chattopadhyay, Sayantani Sarkar, Partha Sarathi Dey, and Abhishek Tripathi for great friendship and time for frequent Skype calls. A big thanks to Caroline for her encouragement and support during my PhD.

I take this opportunity to extend my gratitude to Dr. Parthasathi Barat for inspiring me to do higher studies and research since my high school days. I thank him for all his time and effort in giving thoughtful answers to all my questions over the past years. I would like to thank my grandmothers and other family members for being supportive throughout my studies. Special thanks to my youngest uncle Dr. Rajat Bhattacharyya for providing encouragement, inspiration, and support to pursue higher studies.

Lastly, and most importantly, I would like to thank my parents, Sankar and Rupashree Bhattacharya, and my younger brother Sounak for their love, unconditional support, inspirations, and constant interests on my progress. Without them next to me, I could not have achieved this.

In memory of my dada and dadan

Helsinki, July 9, 2014
Sourav Bhattacharya

Original Publications

The PhD thesis is based on the following four original publications. Throughout the thesis the publications are referred to as Articles I – IV. The articles are reprinted at the end of the thesis.

Article I Petteri Nurmi, Sourav Bhattacharya, and Joonas Kukkonen. A Grid-based Algorithm for On-device GSM Positioning, In *Proceeding of the 12th ACM international Conference on Ubiquitous computing (UbiComp), pages 227–236, September 2010.*
doi:10.1145/1864349.1864385.

Article II Mikkel Baun Kjærgaard, Sourav Bhattacharya, Henrik Blunck, and Petteri Nurmi. Energy-efficient Trajectory Tracking for Mobile Devices, In *Proceeding of the 9th International Conference on Mobile Systems, Applications and Services (MobiSys), pages 307–320, June 2011.*
doi:10.1145/1999995.2000025.

Article III Sourav Bhattacharya, Henrik Blunck, Mikkel Baun Kjærgaard, and Petteri Nurmi. Robust and Energy-efficient Trajectory Tracking for Mobile Devices, In *IEEE Transactions on Mobile Computing*, April 2014.
doi:10.1109/TMC.2014.2318712.

Article IV Sourav Bhattacharya, Petteri Nurmi, Nils Hammerla, and Thomas Plötz. Using unlabeled data in a sparse-coding framework for human activity recognition, In *Pervasive and Mobile Computing, Elsevier*, May 2014.
doi:10.1016/j.pmcj.2014.05.006.

Contents

Chapter 1

Introduction

Modern smartphones not only serve as the primary communication device, but also have become a popular choice of computing platform [76]. Past years have seen rapid technological development in the sensing capabilities and miniaturizing of sensors. Today, off-the-shelf modern smartphones readily support a rich set of on-device sensors, such as GPS, WiFi, GSM, accelerometer, gyroscope, magnetometer and NFC, to name but a few. These sensors can capture various aspects of the surroundings of a user in real time, unobtrusively, and at an astounding rate [7, 140].

The novel sensing capabilities on the mobile platform, the ease in the development of custom mobile applications, opportunities of a large-scale distribution of applications through various app stores, and the availability of scalable computing resources utilizing existing cloud infrastructures have catalyzed an unprecedented growth in mobile sensing research. A large number of innovative applications, which have impacted our everyday lives, are now available to millions of users. Examples of such application domains include location-based services (LBS) [2, 72, 91, 114, 146], healthcare [27, 80, 105], environmental monitoring [96, 114], social networking [94, 141], and transportation [49, 128].

Following Moore's law, as the number of transistors packed per unit area is doubling every 18 months, mobile phone manufacturers are continuously increasing the list of supported features to utilize the available transistor budget [140]. However, the pace of battery capacity improvement is much slower compared to the development of power demanding features of the smartphones [63]. The limited battery power available on the mobile platform thus poses a big challenge to *continuous and sustained sensing*. For example, continuous sampling of the on-device GPS receiver, employed to track a user's motion within a city, can completely deplete the battery of a smartphone in a few hours, thereby limiting the usability of such applications. Thus, novel solutions are needed that optimize the selection and the use of various available sensors under a given energy budget to improve the overall running time of mobile sensing applications.

Emerging mobile sensing applications often build on high-level contexts inferred from noisy sensor measurements on the device, as privacy concerns and costs of repeated radio communications do not always make utilization of cloud services feasible. Examples of high-level contexts include personally meaningful places [99], the type of activity being performed [9], current transportation mode [115], user's destination [69], and health conditions [109]. High-level contexts provide important clues of a user's situational needs, and mobile applications can utilize the inferred contexts to improve usability and user experience through suitable adaptations [119]. Often, the context inference task is formulated as a *classification* problem. Classification belongs to a popular family of algorithms well established within the field of *supervised machine learning* [12]. However, the real-world deployment of these sensing applications face the challenge of a cumbersome and time-consuming *bootstrapping* process needed for the context recognizer to work. For example, the bootstrapping process in a supervised learning approach requires suitable feature representation of the sensor measurements, in addition to the collection of a large training set. The collection of training data, especially on the mobile platforms, is a time-consuming, error prone, and costly process [77, 125]. Additionally, obtaining optimal feature representations is not a trivial problem [13]. For example, extraction of a large number of features requires significant processing power and often the optimal set of features depends on the target activities, thus restricting the generalizability of one feature set to a different target domain or sensor modality. The bootstrapping task thus requires a significant effort, by both the users and the developers of mobile sensing applications, for real-world deployments, thereby limiting the usability of such applications.

The aim of this thesis is to address some of the shortcomings of the existing approaches in ubiquitous computing and mobile sensing with the central research theme:

> *How do we facilitate continuous and sustained context inference on mobile platforms?*

1.1 Thesis Outline

This thesis contributes to the research directions in mobile sensing and ubiquitous computing that are related to the development, understanding, and deployment of continuous context recognition methods from embedded sensors, with a special focus on resource-constrained mobile platforms. The thesis addresses challenges present in two most popular context inference tasks, namely *energy-efficient localization* and *activity recognition.* Accordingly, the thesis is divided into two parts.

In the first part of this thesis we focus on continuous location sensing and tracking on mobile devices. We begin by proposing a new localization algorithm that relies on the existing GSM communication networks. The localization algorithm runs completely on a user's personal mobile phone and uses signal intensity information of the serving GSM cell tower to estimate the user's locations over time. One of the benefits of using GSM-based localization algorithms is that no additional energy is required for sensing the surrounding radio environment. Contrary to the existing approaches to GSM localization, we also develop techniques for maintaining an accurate *radio map* through location dependent GSM signal-intensity modeling and performing model parameter synchronizations over time. A centralized server is used to propagate changes in the radio map to mobile clients. Details of the algorithm are presented in Chapter 2 of the thesis, which is based on Article I.

Often the positioning accuracy of a GSM localization system is limited, e.g., around hundreds of meters. In contrast, on-device GPS receivers provide highly accurate location information outdoors and are a popular choice for localization by the vast majority of LBS [102]. However, integrated GPS receivers are known to have a high energy consumption [103, 129]. Additionally, LBS often require other power consuming features such as the activation of the screen to display maps and the use of a radio chip to send or receive data for their operations [63]. To improve the running time of LBS, in Chapter 3 we focus on methods for accurate and energy-efficient location tracking on feature-rich mobile platforms. We propose the EnTracked$_{RT}$ system that employs novel *sensor-management strategies*, *trajectory simplification*, and an *opportunistic transmission protocol* to optimize the power consumption of tracking a user or a mobile object over time, without violating *application tunable* accuracy requirements. The system judiciously selects between GPS, accelerometer, and compass sensors to achieve energy-efficiency. The EnTracked$_{RT}$ system also provides easy means to ensure *robustness* in tracking. Through application-tunable parameters the system allows for energy consumption and robustness *trade-off.* The EnTracked$_{RT}$ system is described in Chapter 3, which is based on the contents of Article II and Article III. The work on energy-efficient and robust tracking was completed in collaboration with Aarhus University, Denmark.

In the second part of the thesis we address challenges in supervised learning-based activity recognition, which is another important and popular example of a high-level context inference task within ubiquitous computing. We propose a new activity recognition framework based on *sparse-coding* [100, 110] that significantly reduces the efforts needed in the bootstrapping of an activity recognizer. The proposed framework automatically learns *characteristic patterns* present in human motion from a large amount of unlabeled sensor measurements and completely overcomes the requirement of domain-specific feature engineering. The proposed approach requires a small amount of ground-truth data to train a classifier, which is later used to recognize activities from novel sensor measurements. Chapter 4 describes the sparse-coding-based framework for activity recognition. This work was completed in collaboration with Newcastle University, UK.

1.2 Author Contributions

Article I. The algorithmic development of the probabilistic grid-based particle filtering approach and the online radio map synchronization method are joint work with Petteri Nurmi. The gird-coordinate mapping algorithms are joint work with Joonas Kukkonen. The present author also contributed in writing a major part of the article.

Article II. The initial idea of the EnTracked$_T$ system (precursor of the EnTracked$_{RT}$) was given by Mikkel Baun Kjærgaard. The present author contributed in the development of the trajectory simplification algorithms, update protocol, system implementations, mobile deployments, data analysis, and in the writing of the paper.

Article III. The idea of the velocity buffering to improve robustness in tracking and the evaluation of the trade-off between energy consumption and robustness are given by the present author. Other technical aspects of the article are joint work with Henrik Blunk. The author also contributed significantly to the data analysis and writing of the article.

Article IV. The original idea of developing a sparse-coding-based activity recognition framework is given by the present author. The development of the codebook pruning technique and the evaluation framework are joint work with Thomas Plötz. The present author was responsible for the entire implementation, deployment, and analysis of the recognition framework. The writing of the article was joint work with the co-authors.

Part I

Energy-efficient Localization

Chapter 2

On-device GSM Localization

Identifying the accurate location of a user or a mobile device has long been at the heart of ubiquitous computing research [68, 101]. Location sensing is also a core component of context-awareness [48]. Significant improvements in mobile user interfaces, ease in the development of custom mobile applications, availability of application distribution channels, and increasing popularity of mobile internet usage have resulted in a new surge of interest in location-based services (LBS). The revenue generated from LBS is expected to exceed USD 12.7 billion by the end of 2014 [81]. The usability of LBS mainly relies on the accurate knowledge of users' current locations, and the past decade has seen a lot of research on the development of location-sensing technologies and on providing appropriate infrastructural support for LBS.

GPS provides accurate location estimates outdoors; the median error in GPS-based location estimation is within 10–20 meters [61, 75]. However, GPS does not work reliably indoors, where people spend the majority of their time [75]. The performance of GPS is also poor in areas where clear view of the sky is not available, e.g., in so-called *urban canyons* with high-rise buildings. Although the number of GPS-equipped mobile phones is on the rise, only 55% of the mobile devices in 2014 are estimated to have an inbuilt GPS receiver [89]. Moreover, the lack of accurate location coverage by GPS in our everyday lives, in terms of percentage of time [75], has developed a significant research interest in localization using existing wireless communication infrastructures.

The infrastructure of modern wireless communication is based on cellular networks. Examples of such networks include *Global System for Mobile communications* (GSM), *Universal Mobile Telecommunications System* (UMTS), *Wireless Local Area Network* (WLAN), and *Digital Enhanced Cordless Telecommunications* (DECT). Due to proliferation of mobile phone usage, GSM has become the most popular cellular telephony standard in the world. By January 2014, GSM has been deployed in more than 248 countries[1]. In terms of the coverage area, the GSM network is farther outreaching than the existing WiFi networks [101]. Hence, at the moment, GSM is the most ubiquitous wireless cellular infrastructure in the world.

Research efforts on GSM-based localization saw a boost after the E911 mandate by the United States Federal Communications (FCC), which requires the network providers to locate a caller within 125 m, when dialing the emergency number [1]. A similar mandate (E112) exists in Europe. A localization approach using GSM signal information has several benefits, including:

- No requirement of additional hardware. Almost all phones use the GSM network for basic communications.
- No significant burden on the battery in terms of sensing, as opposed to the localization solutions using GPS or WiFi.
- Easily deployable with wide coverage area.
- Cost of deployment is also small, compared to other approaches based on radio beacons.

In this chapter we introduce an on-device algorithm for *narrow* GSM fingerprint-based localization. A narrow GSM fingerprint consists of signal intensity information from only one *base transceiver station* (BTS) [135], contrary to other types of GSM fingerprints such as *six-strongest cells* and *wide* fingerprints [101]. Although the use of additional cell tower information, in general, improves the localization accuracy [23], modern mobile phone APIs often provide access to only one BTS information. We propose an on-device algorithm that has comparable positioning accuracy to the current state-of-the-art GSM localization solutions. The proposed algorithm provides good optimization on the storage requirement of the signal intensity models, and enables an easy way of creating, maintaining, and updating changes in GSM signal environments.

The content of the chapter is based on Article I.

2.1 Approaches to GSM Localization

In this section we summarize previous work on GSM localization by dividing existing research broadly into two categories: (i) fingerprinting-based

[1] `http://www.worldtimezone.com/gsm.html` [Retrieved: July 17, 2014]

approaches and (ii) other approaches to GSM localization including geometric modeling, signal propagation modeling, and machine learning.

2.1.1 Fingerprinting

Fingerprinting is a popular localization technique. Generally, a fingerprinting approach begins with a *training phase* to generate a *radio map* of the environment of interest [135]. During the training, a mobile device is used to record *signal intensity* measurements[2] from a group of radio sources, e.g., GSM, WiFi, Bluetooth, FM radio, or TV stations, at various locations within the environment. Once the radio map is generated, a client can estimate its location by performing a radio scan and then measuring the similarities of the signal intensities obtained from the scan with the radio map. Several different approaches can be employed when computing the similarity between two signal intensity measurements. In case of position estimation, often the similarity computation is formulated as the task of finding a measurement from the radio map that has the same radio sources and has similar signal intensities. The performance of a fingerprint-based localizing algorithm mainly relies on two properties of the fingerprints [68, 101]:

- Spatial variability, i.e., the radio signal intensity observed by a mobile device should exhibit considerable variations, when the device is moved by 1–10 m.

- Temporal stability, i.e., the signal strength of a given radio source at a given location should be consistent over time.

A combination of both properties indicate that the constructed radio map is *feature-rich* in space and reasonably consistent over time [136]. Localization algorithms can exploit the learned radio map later for accurate position estimations. However, the temporal stability of fingerprints may not hold for a very long period of time (e.g., months or years). Especially in GSM networks, changes in signal environment may occur due to incorporation or deletion of BTSs and adjustment of the signal level made by the service provider. Therefore, in case of fingerprint-based GSM positioning, one challenge is:

> *To capture local changes in the radio environment over time and to integrate changes to the previously learned radio map.*

[2] A measurement is composed of several readings, i.e., one value for each radio source in range.

One of the earliest works on GSM localization is by Laitinen et al. [74], where the authors use a GSM fingerprinting-based technique to locate a *mobile station* (MS) in urban and suburban areas of Helsinki, Finland. Initially, *war-driving*[3] is carried out to construct a database of location dependent GSM fingerprints. The fingerprints are composed of signal strength information from six neighboring GSM BTSs. The position of a MS is inferred on a remote server by minimizing a difference score, which includes a channel-based penalty term. The score is computed between the sensed fingerprint and the fingerprints present within the database. The proposed approach achieves a 90-percentile accuracy of 90 m (meters) and 190 m in urban and suburban areas, respectively.

The PlaceLab initiative by LaMarca et al. [75] is probably the best known work on device positioning using radio beacons. The main objective of the PlaceLab approach is to maximize the coverage of a location-sensing method in people's daily lives. The system employes a war-driving approach to construct a radio map of the environment and applies a Bayesian particle filter for position inference. While using only GSM beacons, PlaceLab achieves a median accuracy of 107 m in urban areas and 216 m in suburban areas.

In case of GSM positioning within metropolitan areas, Chen et al. [23] conduct an extensive study of five factors influencing on-device positioning accuracy: (i) the deployed positioning algorithm (e.g., centroid, fingerprint, and Monte Carlo localization with a Gaussian Process-based signal model), (ii) the number of nearby BTSs used, (iii) simultaneous use of cell information from different providers, (iv) testing on different devices, and (v) the calibration data density. The fingerprinting algorithm achieves the best median accuracy of 94 m in areas with high cell density (e.g., downtown), whereas the Gaussian Process-based algorithm achieves the best accuracy of 196 m in low density areas (e.g., residential areas). In general, the presence of additional cell tower information helps both the fingerprinting and the Gaussian Process-based algorithm to achieve better positioning accuracy. When different devices are used for training and testing, the fingerprinting algorithm exhibits poor generalizability, whereas the Gaussian Process-based algorithm shows a superior performance.

Otsason et al. [101] show that wide GSM signal-strength fingerprints can be used for accurate localization within indoor environments. The wide fingerprints include information of up to 29 GSM channels, in addition

[3] War-driving is a process of searching for wireless network availability (e.g., Bluetooth, GSM or WiFi) at a geographical location (usually obtained from GPS), while traveling by car.

to the six-strongest cell towers. However, the system requires a specialized hardware, e.g., a GSM modem, to collect the wide fingerprints. The authors use the k-nearest neighbor algorithm for position estimation and report a median accuracy of five meters in large multi-floor buildings.

The SkyLoc system by Varshavsky et al. [136] uses a GSM fingerprinting technique to locate the current floor of a user within multi-floor buildings. During the training phase, the authors construct building-specific radio maps by scanning available GSM networks across multiple providers. During the evaluation, the authors show that a floor-specific feature selection method helps to improve the overall recognition accuracy of the system significantly, compared to the naive nearest-neighbor algorithm, which does not employ any feature selection. The SkyLoc system identifies the floor of a user, within tall buildings (e.g, 9, 12 and 16 stories), correctly in 73% of the test cases, and the estimation falls within two floors in 97% of the cases.

2.1.2 Non-fingerprinting Approaches

The simplest way to locate a mobile phone is to approximate its position by the (*latitude, longitude*) coordinates of the BTS currently handling the phone's communications. The method assumes that the phone is always connected to the nearest BTS, which is often not the case. Additionally, the approach requires mobile network operators to provide accurate location information of the BTSs, which is often kept private. Trevisani and Vitaletti [132] report that the accuracy of the simple Cell-ID localization approach is around 500–800 m and the accuracy relies mainly on the size of the cell, the cell density, and environmental characteristics. However, the performance of the simple algorithm can be improved by performing a weighted averaging of the locations of the nearby BTSs [75].

The position of a MS can be derived in a number of ways by measuring various signal characteristics of a GSM cellular network and then applying geometry-based methods. The most common signal measurements used for GSM positioning include propagation time, time difference of arrival (TDOA), angle of arrival (AOA), and signal phase [32]. Each of the measurements define a locus on which the MS should be present. For example, a propagation time measurement generates a circular locus, a TDOA measurement creates a hyperbolic locus, and an AOA measurement generates a locus of a straight line. The position of the MS can thus be estimated as the intersection point of several measurement loci. However, the positioning accuracy is susceptible to environmental characteristics, e.g., multi-path propagation, signal attenuation, reflection, and refraction. Additionally, for obtaining adequate accuracy, often additional hardware is needed.

Roos et al. [117] propose a statistical learning approach that models signal propagation of BTSs to infer the position of a MS. The model first learns the dependency of the observed signal intensity to the location. The authors formulate the statistical learning as a propagation parameter estimation task from the data. Once the propagation parameters are learned, the location of a MS is determined using the maximum a posteriori approach over a discrete grid. Instead of modeling signal propagation, Wu et al. [144] propose the use of support vector regression for GSM signal intensity-based position estimation. The authors propose a *sum of exponential kernels* to handle the sparsity in training data.

Schwaighofer et al. [120] propose a Gaussian Process-based model to estimate a mobile user's location using received signal intensity information from network base stations. Instead of using the RBF kernel, the authors employ the Matérn kernel to model the *likelihood* of signal intensities of a base station at various locations. The Matérn kernel allows to control the smoothness using a parameter ν, and can be reduced to the exponential ($\nu = 0.5$) or the Gaussian kernel ($\nu \to \infty$), therefore allowing a greater flexibility [41]. The position of the mobile user is then estimated by maximizing the joint likelihood with respect to the unknown position. The proposed approach outperforms the RADAR system [8], a fingerprinting approach employing nearest neighbor search, especially when the calibration data is sparse. Ferris et al. [35] combine the Gaussian process with a Bayesian filtering approach to estimate a user's location using the signal intensity of wireless base stations. The authors propose a novel motion model employing a mixed graph/free space representation of the environment. In case of GSM positioning, the proposed approach results in a median error of 128 m, 208 m, and 236 m respectively in downtown, residential, and in suburban areas.

2.2 Probabilistic GSM Localization

We follow a probabilistic localization approach, based on Bayesian filtering [38], to estimate the location of a MS within a GSM network. The proposed algorithm belongs to the family of algorithms known as *Monte Carlo Localization* (MCL). MCL is a recursive Bayes filter that periodically estimates the *posterior* probability distribution of the state of a system conditioned on the observed sensor measurements [131]. In the case of GSM positioning, the task is to estimate the posterior at time t, over the mobile device's state or location x_t, conditioned on all measurements $D_{0:t}$ obtained from the surrounding GSM environment. The posterior is also commonly referred to as the *belief* and is denoted as:

$$Bel(x_t) = p(x_t \,|\, D_{0:t}) \tag{2.1}$$

In our case, the dataset $D_{0:t}$ is composed of a sequence of GSM scans performed by the MS, where each scan returns the identifier c of the BTS currently serving the phone and the associated signal strength s (dBm). More specifically:

$$D_{0:t} = \{d_0, \ldots, d_t\}, \text{where } d_i = (c_i, s_i) \tag{2.2}$$

Hence, the belief at time t can be written as:

$$Bel(x_t) = p(x_t \,|\, d_t, d_{t-1}, \ldots, d_0) \tag{2.3}$$

Applying the Bayes rule, Equation 2.3 can be transformed to:

$$Bel(x_t) = \frac{p(d_t \,|\, x_t, d_{t-1}, \ldots, d_0)\, p(x_t \,|\, d_{t-1}, \ldots, d_0)}{p(d_t \,|\, d_{t-1}, \ldots, d_0)} \tag{2.4}$$

The denominator of the above equation is a normalizing constant and is independent of x_t. The equation can be rewritten as:

$$Bel(x_t) = \eta\, p(d_t \,|\, x_t, d_{t-1}, \ldots, d_0)\, p(x_t \,|\, d_{t-1}, \ldots, d_0), \tag{2.5}$$

where $\eta = p(d_t \,|\, d_{t-1}, \ldots, d_0)^{-1}$. As the number of sensor measurements increases with time, exact computation of the belief $Bel(x_t)$ becomes intractable [38]. To overcome the problem, the Bayes filter relies on the *Markov assumption* [131], i.e., given the knowledge of the current state, future sensor observations are independent of the past observations. The Markov assumption helps to simplify Equation 2.5 as:

$$Bel(x_t) = \eta\, p(d_t \,|\, x_t)\, p(x_t \,|\, d_{t-1}, \ldots, d_0), \tag{2.6}$$

Now, expanding the right-most term of the above equation we get:

$$\begin{aligned} Bel(x_t) &= \eta\, p(d_t \,|\, x_t) \int p(x_t, x_{t-1} \,|\, d_{t-1}, \ldots, d_0)\, dx_{t-1} \\ &= \eta\, p(d_t \,|\, x_t) \int p(x_t \,|\, x_{t-1})\, p(x_{t-1} \,|\, d_{t-1}, \ldots, d_0)\, dx_{t-1} \\ &= \eta\, p(d_t \,|\, x_t) \int p(x_t \,|\, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1} \end{aligned} \tag{2.7}$$

Thus:

$$Bel(x_t) \propto p(d_t \,|\, x_t) \int p(x_t \,|\, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1}, \tag{2.8}$$

where $\propto$ refers to proportionality. The above recursive equation forms the core of a Bayes filter [38, 130, 131] and is updated whenever a new GSM scan is performed on the phone.

We use *particle filtering*, which is a nonparametric implementation of the Bayes filter, to maintain the belief $Bel(x_t)$ of a MS's location. The basic idea in particle filtering is to represent the posterior $Bel(x_t)$ by a set of random state variables or particles instead of representing the distribution in a parametric form, e.g., using a multivariate Gaussian distribution. The filtering approach relies on an accurate estimation of the term $p(d_t|x_t)$ in Equation 2.8, which is called the *likelihood*. In our proposed GSM localization approach, we first construct a radio map of the signal environment, which is later used to estimate the likelihood of GSM signal intensities on a 2D grid over the earth's surface. Section 2.2.1 describes our employed likelihood estimation approach in detail. In addition to the simplicity in its implementation, the use of particle filtering allows for representing arbitrary distributions and modeling nonlinear transformations [130]. The details of our particle filtering implementation are described in Section 2.2.2.

2.2.1 Likelihood Estimation

We use a discrete representation of the world and employ a GSM *signal intensity model* for each grid cell to construct the likelihood (e.g., $p(d_t|x_t)$ in Equation 2.8) for geography-specific GSM signal characteristics. Accordingly, we lay a grid over the world with a patch dimension of $dim \times dim$ (m^2). The parameter dim plays an important role in determining the trade-off between the target localization accuracy and the radio map size [46]. In our experiments we set dim = 20 m, which is selected based on the GPS accuracy. More specifically, the lower bound for the parameter dim should be at least the average accuracy of the position system used to obtain the calibration or training data.

Grid-Coordinate Mappings

We have developed *forward* and *backward mapping* algorithms for seamless transitions between the grid cells and the GPS coordinates. Given the (*latitude, longitude*) coordinates[4] of a point, the forward mapping returns

[4] We assume that a location is represented using the WGS 84 reference system.

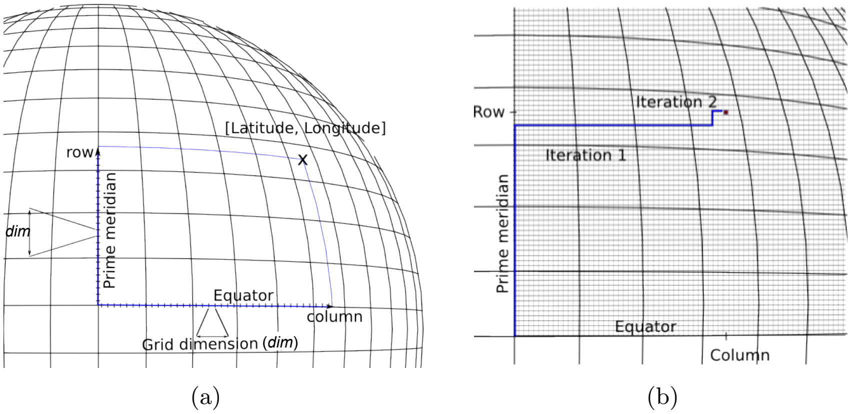


Figure 2.1: Illustrating the concepts of (a) forward mapping, i.e., from coordinates to grid cell and (b) backward mapping, i.e., from grid cell to coordinates.

the row and column identifiers of the grid cell containing the point. When computing the column identifier, the forward mapping algorithm first computes the geodesic distance[5] $dist_x$ of the given location from the point that is located on the *prime meridian* and has the same latitude. Then the column identifier is computed as:

$$column = sgn(longitude) \left\lfloor \frac{dist_x}{dim} \right\rfloor, \tag{2.9}$$

where $sgn(x) = \frac{x}{|x|}$. Similarly, the row identifier is computed as:

$$row = sgn(latitude) \left\lfloor \frac{dist_y}{dim} \right\rfloor, \tag{2.10}$$

where $dist_y$ is the geodesic distance of the given location from the point on the *equator* having the same longitude. Figure 2.1(a) illustrates the basic idea of the forward mapping algorithm.

The backward mapping algorithm, on the other hand, computes the (latitude, longitude) coordinates of the corner of the grid cell closest to (0, 0) with given row and column identifiers. The algorithm initializes the estimated coordinates = (0, 0), $\Delta_x = |column| * dim$ m, and $\Delta_y = |row| * dim$ m, where $|\cdot|$ represents the absolute value. Then the algorithm iteratively updates the estimates by moving first Δ_y m keeping the current

[5] We use Vincenty's algorithm [138] to compute the geodesic distance.

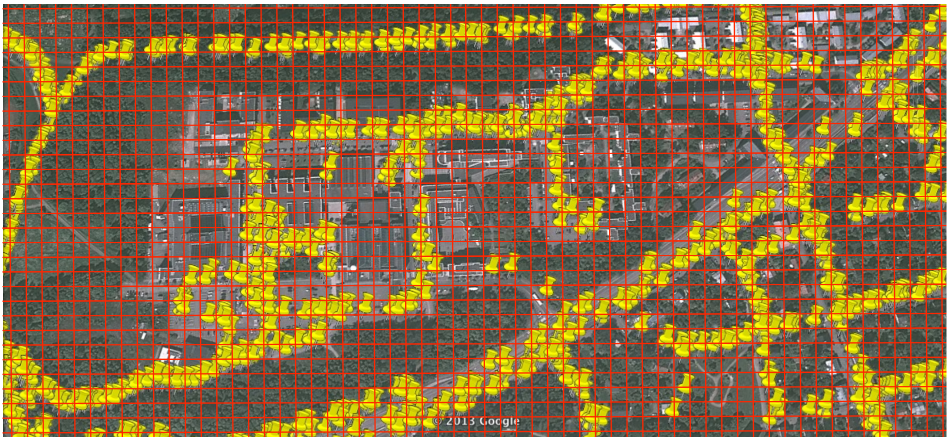


Figure 2.2: Example of GSM training data locations obtained around Kumpula, Helsinki during the war-driving on top of the grid system.

longitude fixed, and then moving Δ_x m keeping the latest latitude fixed. After the translations, the new estimate of the location coordinates are (x, y). Especially when Δ_x or Δ_y is large, geodesic translations may introduce some deviations due to rounding errors [138]. To overcome the problem, we update $\Delta_x = |column| * dim - geodist(x, y, 0, y)$ and $\Delta_y = |row| * dim - geodist(x, y, x, 0)$. Here, $geodist(lat_1, lon_1, lat_2, lon_2)$ refers to the geodesic distance between the coordinates (lat_1, lon_1) and (lat_2, lon_2). The algorithm then iteratively performs similar translations (as before) starting from the latest coordinates (x, y) until both Δ_x and Δ_y become negligible. Figure 2.1(b) summarizes the basic idea of the backward mapping algorithm.

Radio Map Generation

We used war-driving to collect GSM narrow fingerprints from various areas within Helsinki, Finland. We chose different areas with varying GSM cell tower densities (low-density and high-density residential areas) to collect the training data. During the war-driving, we used Nokia S60 devices to scan the surrounding GSM environment, and the location of the scan was recorded using external GPS receivers. For example, Figure 2.2 illustrates the geographic locations around Kumpula, Helsinki (yellow push pins) on GoogleEarth[6] from where GSM scans were obtained during the war-driving. The figure also depicts the grid structure used to facilitate the training pro-

[6] http://www.google.com/earth/ [Retrieved: July 17, 2014].

cedure. Using the forward mapping algorithm, we next construct the radio map from the training data. The constructed radio map is composed of records containing the grid cell identifier (g), the number of data points obtained on the grid cell (n), the mean (μ), and the standard deviation (σ) of the GSM signal intensities observed within the grid cell from a specific BTS, along with the BTS identifier (c). Keeping only the summary statistics of the training data makes the radio map size invariant to the size of the training dataset.

Signal Intensity Model

The signal intensity model captures the variation of GSM signal characteristics from a specific BTS on the discretized world. For simplicity, we assume that the signal intensity variations within a grid cell follow a Gaussian distribution, and the probabilities of observing a GSM fingerprint on two different grid cells are independent of each other. More specifically, the probability of observing the latest GSM scan data, $d_t = (c_t, s_t)$, given a grid cell g is computed as:

$$p(d_t \mid g) = \mathcal{N}(\mu_{g,c}, \sigma_{g,c}) = \frac{1}{\sqrt{2\pi\sigma_{g,c}^2}} \exp\left(-\frac{(s_t - \mu_{g,c})^2}{2\sigma_{g,c}^2}\right), \tag{2.11}$$

where $\mu_{g,c}$ and $\sigma_{g,c}$ are the summary statistics (present in the radio map) for BTS[7] c in grid cell g. If no radio map data is present for the BTS c within the set of grid cells $G' = \{g_1, \ldots, g_n\}$, then we consider $p(d_t \mid g_i) = 0, \forall g_i \in G'$. The zero probabilities and Equation 2.11, combined with the independence assumption, constitute the overall likelihood, i.e., $p(d_t \mid x_t)$ in Equation 2.8.

2.2.2 Particle Filtering

Once the likelihood is computed, we can implement the recursive Bayes filtering, as given in Equation 2.8, using a particle filtering approach developed on the same grid layout used during the generation of the radio map. More specifically, we represent the belief $Bel(x_t)$ by a set of N weighted particles [131] on the grid, i.e.:

$$Bel(x_t) \approx \{x_t^i, w_t^i\}_{i=1}^N, \tag{2.12}$$

where each particle x_t^i represents a hypothesis, in our case a potential location (grid cell) of the MS, and has a non-negative weight or *importance*

[7] For notational simplicity we drop the time index, i.e., here $c = c_t$.

factor w_t^i. As the number of particles N goes to *infinity*, the discrete probability function, represented by the set of particles, converges to the true posterior $Bel(x_t)$. In [67] we studied extensively the relationship between the number of particles N and the achieved localization accuracy. We found that the best localization accuracy is achieved when $N \geq 3,000$. However, we currently use $1,000$ particles for near real-time[8] position estimation on the mobile devices. In our implementation if a particle is assigned to a grid cell, it is placed at the center of the cell.

Particle Initialization

At the beginning of a tracking period, given the first GSM scan data d_0, we sample N particles based on the grid likelihood $p(d_0 \,|\, g)$, computed using Equation 2.11, for different grid cells. More specifically, the probability that a particle will be sampled at a grid location g is:

$$p(x_0^i = g) \sim C^{-1}\, p(d_0 \,|\, g), \tag{2.13}$$

where C^{-1} is a normalization constant that ensures the probabilities $p(d_0 \,|\, g)$ over all grids add to unity. Initially all particles are assigned equal weights of $1/N$. Hence, the initialization process tend to assign more particles to the grid cell with higher likelihood of observing the data d_0.

Particle Motion Model

We employ a simple Gaussian motion model while computing the probability $p(x_t \,|\, x_{t-1})$, see Equation 2.8. Given the current location (at time t) of a particle at grid cell g, we assume the probability that the particle will move by a distance of k meters along the same row and column of the grid cell g is given by a Gaussian distribution. More specifically, we assume that the row-wise and the column-wise particle movements are independent and the amount of movement is sampled from a Gaussian distribution, i.e., $k \sim \mathcal{N}(0, \epsilon)$. The standard deviation of the Gaussian motion model ϵ is selected based on the number of particles used for approximating the $Bel(x_t)$, and we use $\epsilon = \sqrt{N/3}$ in our implementation as a proof of concept. Thus a large number of particles would assign a higher standard deviation for the Gaussian motion model, which will favor wider explorations in the hypothesis space. Moreover, the motion model generates data in meters, and we discretize the error into grid width by using the floor function and the grid dimension parameter *dim*. The motion model thus distributes some of

[8] The convergence speed of a particle filtering approach is $\mathcal{O}(1/\sqrt{N})$ [131].

the particles assigned to a grid cell to other nearby grid cells. The motion model can be summarized as:

$$x_t^i \sim x_t^i + \lfloor \mathcal{N}(0, \epsilon)/dim \rfloor \tag{2.14}$$

Importance Adjustment and Resampling of Particles

Once the motion model is applied, we compute the importance or weight of an individual particle. We assume that each grid cell with non-zero likelihood contributes to the weight of a particle, and the contribution depends on the distance between the grid cell and the current location of the particle. More specifically, we update the weight of a particle as:

$$w_t^i = \sum_{j=1}^{M} p(d_i \,|\, g_j) \exp\left(-\frac{||x_t^i - g_j||_1}{2}\right), \tag{2.15}$$

where M is the number of grid cells having non-zero likelihood for the data d_i, and $||x_t^i - g_j||_1$ is the Manhattan distance between the grid location of the particle x_t^i and the grid cell g_j. Normalization of the particle weights is done before resampling N new particles using the distribution represented by the particle weights.

Location Estimation

At any time instant t, the set of particles collectively represents the posterior probability distribution of the MS on the discrete world. Hence, the grid cell location of the MS can be estimated by the weighted centroid of the particle distribution, i.e.,

$$\hat{g} = \frac{\sum_{i=1}^{N} w_t^i x_t^i}{\sum_{i=1}^{N} w_t^i}. \tag{2.16}$$

The (latitude, longitude) coordinates from the grid location estimate $\hat{g}$ can be computed using the backward mapping algorithm (see beginning of the current section).

2.3 Localization Performance

In this section of the thesis, we only report the localization accuracy of the proposed GSM localization algorithm. Readers are referred to the evaluation section of Article I for an in-depth analysis on the localization

	High Density Area			Low Density Area		
	Error (m)			Error (m)		
	50%	90%	100%	50%	90%	100%
Grid + particle filter	139.7	399.2	**480.5**	**107.6**	328.6	**769.0**
Grid likelihood	174.9	366.4	710.0	116.3	**312.2**	795.7
Nearest neighbor [23]	**112.2**	**306.3**	795.7	116.6	312.3	795.7

Table 2.1: Comparison of positioning accuracies.

accuracy, cross-device generalization performance, and radio map storage requirements of the proposed algorithm.

We evaluated the localization performance of the algorithm in two residential areas in Helsinki with varying GSM cell-tower (BTS) densities. The overall median error of our proposed grid-based GSM localization algorithm was found to be within 150 m in both high-density and low-density residential areas. For comparison, we consider (i) a variant of our proposed system that does not use the particle filtering (denoted by grid likelihood) and (ii) the k-nearest neighbor algorithm by Chen et al. [23]. Table 2.1 summarizes different percentiles of the localization error demonstrated by the three algorithms. In both test areas, our proposed localization algorithm achieves a median error of 139.7 m and 107.6 m respectively. Thus, our proposed algorithm can efficiently use the signal strength information from a single BTS to provide localization service to a large number of mobile devices without requiring significant deployment costs.

2.4 Radio Map Maintenance

The accuracy of a fingerprint-based localization system depends mainly on the availability of a dense calibration data [135] and on the consistency of the radio map, more specifically on the accuracy of the likelihood model with time [70]. GSM signal intensity measurements are inherently noisy, and measurements obtained from the same location tend to vary due to several factors including time of the day, dynamic load on the BTSs, different hardware used for obtaining the measurement [32], and addition/deletion of BTSs by the mobile phone operator. A GSM localization algorithm thus needs to detect and update changes in the radio environment to stay consistent over time. As one of the salient novelties of our proposed localization approach, in this section we introduce ways of creating, maintaining, and updating the GSM radio map over time.

Online Learning of Radio Map Parameters

Mobile phones equipped with a GPS receiver can be used to learn the radio map parameters of the grid cells, i.e., $\mu_{g,c}$ and $\sigma_{g,c}$, locally. Once the radio map parameters are learned, the user can switch off the GPS to save energy, yet can achieve accurate location estimates based on the GSM signal information only. Moreover, when the learned radio map is shared, it can benefit a large number of users in the proximity, especially those without an integrated GPS receiver on their phones.

A number of approaches have been proposed to maintain a radio map across devices over time, e.g., one salient approach is proposed by Park et al. [42], where the authors use Kernel density estimation to maintain radio map of WiFi access points across heterogeneous devices. In our case of GSM localization, we follow an online estimation technique for learning the likelihood parameters. Online estimation of summary statistics of the signal intensity model overcomes the requirement of storing measurements from the GSM scans on the device, thereby optimizing the storage requirement on the phone. Specifically, if a BTS c is observed with a signal strength of s while in a grid cell g, the online estimates of the mean μ and the sum of squared deviation from the mean S can be computed using [20]:

$$\begin{aligned} \mu_{g,c}^{n} &= \mu_{g,c}^{n-1} + \frac{1}{n}(s - \mu_{g,c}^{n-1}), \qquad (2.17) \\ S_{g,c}^{n} &= S_{g,c}^{n-1} + \frac{n-1}{n}(s - \mu_{g,c}^{n})^2, \qquad (2.18) \end{aligned}$$

where n is the number of samples used to estimate the parameters for the grid cell and BTS pair. We also store the value of n in the radio map. The standard deviation of the signal intensity measurements is then computed as:

$$\sigma_{g,c}^{n} = \max\left\{5, \sqrt{\frac{S_{g,c}^{n}}{n-1}}\right\} \qquad (2.19)$$

We set a lower bound of 5 to the standard deviation parameter to avoid a highly skewed signal intensity model for a grid cell. The tool for learning a radio map on a user's mobile phone has the potential of large-scale deployment, where the radio map can be learned incrementally using crowdsourcing techniques. We support the distribution of a radio map learned on a user's mobile phone with the help of a centralized server.

Radio Map Synchronization

The opportunity of learning radio map parameters locally on a user's mobile phone also helps to detect changes in the surrounding GSM radio environment. If the local radio map changes significantly from the original deployment, the new set of parameters should be propagated to other devices for accurate positioning. We use the *joint Kullback-Leibler* (JKL) divergence metric to identify changes in signal intensity models. For example, let $p(x)$ and $q(x)$ denote two arbitrary probability distributions, then the KL-divergence between $p(x)$ and $q(x)$ is given by:

$$D_{KL}(p\,||\,q) = \int p(x) \log \frac{p(x)}{q(x)} dx \tag{2.20}$$

The joint KL-divergence[9] between the two distributions is given by:

$$D_{JKL}(p\,||\,q) = D_{KL}(p\,||\,q) + D_{KL}(q\,||\,p) \tag{2.21}$$

In case the two distributions are both univariate and Gaussian, i.e., $p(x) \sim \mathcal{N}(\mu_p\,,\,\sigma_p)$ and $q(x) \sim \mathcal{N}(\mu_q\,,\,\sigma_q)$, the joint KL-divergence can be computed as:

$$D_{JKL}(p\,||\,q) = \frac{1}{2}\left[\frac{\sigma_p^2}{\sigma_q^2} + \frac{\sigma_q^2}{\sigma_p^2} + (\mu_p - \mu_q)^2(\frac{1}{\sigma_p^2} + \frac{1}{\sigma_q^2}) - 2\right] \tag{2.22}$$

As our likelihood is modeled using a Gaussian distribution, we use Equation 2.22 to detect changes in the radio map locally and trigger an update, if the D_{JKL} between the previously deployed signal intensity distribution and the latest distribution for a grid cell exceeds a predefined threshold. In our current implementation we set the threshold experimentally.

Moreover, on the server side, a radio map integration method needs to be deployed when several clients send updates of radio map parameters for the same grid cell. We follow a simple weighted averaging scheme for this purpose, where the weights are determined based on the number of samples used to compute the individual estimates. Specifically, let N^0 and N^C denote the number of measurements used to compute the first parameter estimate for a grid cell stored on the client device and the number of measurements used when the parameter update is triggered for the same grid cell. Moreover, let N^S is the number of samples on the server used to compute the parameters for the same cell. While performing the weighted averaging, as the weight for the client parameters we use $W_C = N^C - N^0$

[9] Joint KL-divergence is commutative, whereas KL-divergence is not.

and we use $W_S = N^S - N^0$ as the weight for the server side parameters. Formally, we obtain the new parameters after a client update as:

$$\mu^{new} = \frac{W_S \times \mu^S + W_C \times \mu^C}{W_S + W_C} \tag{2.23}$$

$$\sigma^{new} = \frac{W_S \times \sigma^S + W_C \times \sigma^C}{W_S + W_C} \tag{2.24}$$

$$N^{new} = N^S + N^C - N^0 \tag{2.25}$$

Hence, with the deployment of the online estimation of the likelihood parameters and their synchronized update, the server maintains a global view of the radio map that is consistent over time. The server is also responsible for the periodic distribution of the latest radio map to the mobile phones requesting the localization service.

2.5 Discussion

The proposed GSM localization algorithm relies on the availability of a radio map of the environment on the phone, where the localization service is requested. Moreover, when constructing a radio map, the storage requirement depends linearly on the number of BTS and grid cell pairs the device has seen. Additional storage space is also needed for the indexing scheme used for fast retrieval of the relevant radio map parameters. For example, a radio map that stores the signal intensity model parameters from one BTS for all the grid cells ($20\,m \times 20\,m$) would require around 5 Mb storage space for a coverage area of $100\,\text{km}^2$. However, the radio map storage requirement for everyday activities is significantly smaller and can be incrementally fetched from the server as required, e.g., when the MS connects to a new BTS, the new radio map for the BTS can be downloaded to the device.

Mobile devices of different manufacturers have different sensitivity levels, e.g., two mobile devices from different manufacturers may report different GSM signal strength information, even though the phones are located next to each other. Thus the radio map generation method can be adapted based on the knowledge of the embedded operating system and is left as a future work.

Lastly, the proposed parameter synchronization method is susceptible to attacks such as erroneous reporting by a mobile device or sybil attack [30], where a single mobile device sends update parameter requests for a large number of grid cells claiming different identities. Thus, further research is needed to deploy better radio map synchronization techniques that can identify the reliability and the identity of mobile devices, e.g., with the help of reputation scores to provide protections against miscreants.

Chapter 3

Energy-efficient and Robust Tracking

Emerging LBS prefer localization using GPS on smartphones as it provides accurate location estimates outdoors, compared to GSM- or WiFi-based localization solutions [102]. However, GPS receivers in general consume high energy to obtain location fixes and an application sampling GPS continuously can deplete the battery in a few hours, thereby limiting the usability of the application. In addition to spending energy due to localization, LBS running on a mobile device often consume power by keeping the device display on for rendering information on a map and using the radio transceiver for data communications. Therefore, the success of LBS rely on the level of power consumption while using a phone's features, especially if the services run continuously over an extended period of time [63]. From a developer's perspective, a high energy footprint of an application is detrimental, as it decreases end-user satisfaction and can even lead to the application being classified as *malware* [133]. Hence, the limited battery power available on the mobile platform poses a significant challenge for the deployment of LBS requiring continuous location sensing [148]. Moreover, optimizing only the energy consumption is not enough, as often tracking applications require a guarantee on the accuracy violation rate or *robustness*; this has not been addressed adequately in the literature.

In this chapter we introduce the EnTracked$_{RT}$, a tracking system for mobile devices, that addresses two problems prevalent in smartphone-based tracking solutions: (i) high *energy consumption* and (ii) lack of measures for maintaining high robustness. The content of the chapter is based on Articles II and III.

3.1 Challenges in Continuous Location Sensing

Application distribution channels, such as Google Play[1] and Apple App Store[2], contain a plethora of location-based applications. Emerging LBS often require continuous and accurate location information of a user for location-sensitive information delivery. Examples of such applications include health-care applications [137], microblogging [40], sports tracking [2], location-based games [91, 112], local traffic monitoring [54], cyclist experience mapping [33], personal reminders [86], place recognition [6, 99], location-based social networking [146], participatory sensing [114], and many others.

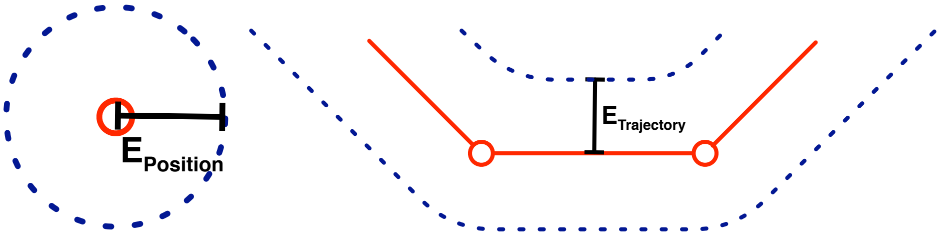
Though more accurate, on-device GPS receivers have a comparatively large energy footprint [103, 129]. For instance, on Nokia N95 mobile devices GPS consumes on an average around 320 mW (milliwatt) [63, 102], and on Android platforms the consumption is around 260 mW [82, 145]. With limited battery power available on the mobile platforms, sampling of GPS in a continuous manner is infeasible, as it completely depletes the battery within 6–7 hours (e.g., in case of Nokia N95 and Android Developer phones) [40, 148]. Therefore, continuous GPS sensing is a major barrier to all-day smartphone usage. One simple solution to reduce the power consumption of GPS is to apply *duty-cycling* with large time intervals (e.g., 3–5 min). In other words, energy can be saved by putting the GPS receiver to sleep in-between two successive samplings. This simple strategy, however, cannot guarantee that the error in position tracking is bounded, especially if the target has moved significantly during the sleeping period of the GPS. Hence, a key challenge in mobile phone-based tracking is:

> *To provide accurate location information while spending as little energy as possible.*

However, the objective of accurate location tracking is conflicting with the simultaneous objective of minimizing energy consumption. For example, GPS measurements obtained at short and regular intervals (e.g., every second) provide highly accurate location information during the tracking period, at the expense of high power consumption. Hence, an effective solution to energy-efficient tracking should intelligently deploy strategies to obtain a trade-off between the requested tracking accuracy and the available energy budget. In this chapter we introduce EnTracked$_{RT}$, which takes into account other external factors, such as regularities in human movements,

[1] `https://play.google.com/store/apps` [Retrieved: July 17, 2014].

[2] `http://www.apple.com/itunes/charts/` [Retrieved: July 17, 2014].



(a) Position tracking.

(b) Trajectory tracking.

Figure 3.1: Illustrating the concept of (a) position tracking and (b) trajectory tracking with uncertainty regions.

the transportation mode, and the speed information in order to provide an energy-efficient and robust tracking solution for mobile devices.

3.2 Approaches to Energy-efficient Tracking

Over the past years several attempts have been made to improve the overall energy consumption of tracking systems. In this section we summarize relevant related work addressing two main problems in energy-efficient tracking: (i) position tracking and (ii) trajectory tracking.

3.2.1 Position Tracking

The basic idea in energy-efficient position tracking is illustrated in Figure 3.1(a). In the figure, the latest sensed position of a moving object is depicted by the small red circle. An energy-efficient system may operate by scheduling the next position sensing when the true location of the object can no longer be ensured to lie within the circular uncertainty region, defined by an error radius or threshold $E_{Position}$.

Existing systems achieve energy saving by applying duty-cycling on the GPS receiver and opportunistically using other sensors with relatively low power requirements. While applying duty-cycling, one key challenge is:

> *To dynamically determine the time of the next GPS sensing, as static duty-cycling, i.e., always sleeping for the same period of time, is error-prone due to being oblivious to the underlying human movements.*

Paek et al. [102] propose the Rate Adaptive Positioning System (RAPS) based on the observation that in urban environments GPS exhibits poor location accuracy. For energy efficiency duty-cycling is employed on the GPS receiver. Moreover, a blacklist of GSM cell towers, e.g., environments where previous attempts of obtaining GPS measurements have failed, is maintained to further delay GPS activations. The system also uses duty-cycling on the accelerometer to estimate the average activity level of a user and only allows new GPS sensing when the user's current movement exceeds an accuracy bound computed from past motion history. Through Bluetooth, the RAPS system shares position information among nearby devices in an attempt to further reduce the power consumption.

The a-Loc system, proposed by Lin et al. [82], optimizes energy consumption in positioning for mobile search applications. Combined with a probabilistic location prediction approach, the system employs a sensor selection algorithm to dynamically select among GPS-, WiFi-, Bluetooth- and GSM-based localization approaches under different accuracy requirements. The accuracy requirement is estimated based on the user's current location and the spatial density of the entities being searched. For example, while searching for the five nearest pizza places, a finer location granularity is used if the density of nearby pizza places is high.

Zhuang et al. [148] propose four design principles for energy-efficient location sensing, namely: (i) *substitution*, (ii) *suppression*, (iii) *piggybacking* and (iv) *adaptation*. To save energy consumption, the system substitutes GPS with less power consuming networking-based positioning whenever possible. This is achieved by learning area-specific profiles based on past availability and accuracy of positioning. Additionally, the system employs an accelerometer to detect motion and suppresses unnecessary location sensing while the device is stationary. When more than one location-based applications are running, the system piggybacks new sensing requests with existing requests. Lastly, to further improve the running time, the frequency of sensing is adapted based on the current battery level.

Based on the observation that people spend the majority of their time in personally meaningful indoor places, Chon et al. [24, 25] propose the SmartDC system for energy-efficient everyday location monitoring. The SmartDC system begins by collecting the mobility data of a user by polling location sensors such as GPS, WiFi and GSM at regular intervals (e.g., 2 minutes) and builds a predictor model for estimating residence time at a meaningful place. The system then optimizes for a sensing schedule under a given energy constraint by employing an adaptive duty-cycling estimated using the Markov decision process.

Another energy-efficient position tracking system is EnTracked by Kjærgaard et al. [65], which is a precursor of the EnTracked$_{RT}$ system. The EnTracked system employs two sensor management strategies to optimize the use of on-device GPS and accelerometer sensors. Adaptive sensor duty-cycling is achieved using estimation and prediction of system parameters, as well as the underlying mobility of the user. The system supports an application-specific error threshold $E_{Position}$ to adapt GPS sampling intervals to the accuracy requirements of different LBS.

Contrary to all previous approaches which treat the GPS receiver as a black box, the low energy assisted GPS (LEAP) system proposed by Ramos et al. [111] divides the processing pipeline of a GPS receiver and only outputs the sub-millisecond part of the *time-of-flight*[3] [61] encoded within GPS signals. Energy is saved by applying duty-cycling on the *Radio Frequency* (RF) front-end of the receiver and offloading tasks such as decoding time stamps from satellite signals, least square location computation and output generation (e.g., latitude, longitude) to a cloud.

3.2.2 Trajectory Tracking

Contrary to the task of position tracking, a trajectory tracking system tries to ensure that the error in sensed motion history (irrespective of current time) of an object always stays within an error bound $E_{Trajectory}$. Figure 3.1(b) illustrates the notion of trajectory tracking, where the error bound corresponds to an *error corridor* surrounding the true path of motion (shown in red) of a moving target.

Trajectory data analysis, e.g., data collection, simplification, storage, processing, and query performance have been extensively studied in the field of moving object databases [45]. Among early approaches, Wolfson et al. [142] propose position update policies for moving objects to minimize overall cost due to database updates and inaccuracies in position queries. Accordingly, the authors model the position update frequency as a function of the ratio between the update cost and the cost of inaccuracies in position estimates. Given the exact route of the moving object, the cost due to inaccuracies is computed by *integrating* a deviation function over a period of time.

Thiagarajan et al. [129] propose the CTrack system for energy-efficient trajectory *mapping* for mobile devices using GSM fingerprints. The system employs a two-pass Hidden Markov Model (HMM) to match a sequence of GSM fingerprints to road segments. The mapping algorithm runs on a re-

[3] Time required for the GPS signal to travel from a satellite to the GPS receiver.

mote server and receives sensor measurements from a device every minute over a wireless link. Uncertainties in GSM localization is reduced by inferring binary clues from on-device sensors. For example, an accelerometer is used to detect if the device is stationary, and a compass or gyroscope is used to detect if the device is turning. Though energy-efficient, the use of coarse-grained cellular fingerprints limits the overall accuracy of the CTrack system in case of trajectory tracking.

The CAPS system by Paek et al. [103] is another example of energy-efficient tracking that uses GSM cell-id sequence matching. The system is motivated by the fact that people often take similar routes every day and observe similar GSM cell transitions. The system opportunistically learns the user's trajectory by sampling the on-device GPS and maintaining a database. For GSM sequence matching, the CAPS system employs an approach based on dynamic programing and selects the route with the highest matching score. Once the best sequence is found in the database, CAPS uses temporal interpolation for estimating the user's current location on the route. However, on failing to find a match, the system triggers GPS sensing for position estimation.

Lange et al. [78] propose a generic real-time *trajectory simplification* protocol for mobile objects for accurate trajectory tracking. The proposed tracking system employs a *linear dead-reckoning* approach to trigger transmission of motion history to a remote server based on a given error bound. Before transmitting novel motion measurements, the system performs trajectory simplification (see Section 3.4.3 for details) on the device to minimize the cost of data communications. The system, however, employs a periodic GPS sampling strategy for position estimation and therefore exhibits high power consumption.

3.3 Robustness in Trajectory Tracking

The majority of the previous systems (described in Section 3.2) only try to optimize the power consumption. However, optimizing power consumption solely is not sufficient, as trajectory tracking applications often require a guarantee on the robustness, i.e., the tracking system should never or rarely violate the accuracy requirement or the error threshold. For example, the robustness requirement of an application assisting a visually impaired person is much higher than an application mapping jogging paths. However, providing a universal guarantee on the robustness is difficult as the accuracy of positioning, and therefore the robustness, depends on the hardware used, the surrounding environment and the handling of the device [15], among others. We argue that the problem can be mitigated, if a tracking system adapts not only to the accuracy error bound, but also to the application-specific robustness requirements. Below, we provide a formal definition of robustness, used in our analysis, by first defining a trajectory and an error metric.

A trajectory can be defined as a sequence of location data points obtained at discrete time intervals. The locations are generally linearly interpolated to represent a user's motion history or trajectory. Formally, following Lange et al. [78], we adopt the definition of a trajectory $\ddot{a}$ as a piecewise-linear function which is defined by an ordered sequence of data items $\{a_1, a_2, \ldots, a_n\}$. Each data item $a_i = (a_i.t, a_i.\overrightarrow{p})$, has an associated timestamp t and a position $\overrightarrow{p}$ (e.g., latitude, longitude) information. In other words, a trajectory $\ddot{a}$ can be viewed as a mapping from time to space and is defined through *spatio-temporal line segments* $\{\overline{a_i a_{i+1}}\}_{i=1}^{n-1}$ as follows:

$$\ddot{a} : t \mapsto \overline{a_i a_{i+1}}(t) \ \text{, where } a_i.t \leq t \leq a_{i+1}.t \tag{3.1}$$

Given the line segment and time, the position of the moving object can be determined using linear interpolation as follows:

$$\overline{a_i a_{i+1}} : t \mapsto \frac{(a_{i+1}.t - t)a_i.\overrightarrow{p} + (t - a_i.t)a_{i+1}.\overrightarrow{p}}{a_{i+1}.t - a_i.t}. \tag{3.2}$$

The sensed trajectory on a mobile device can deviate significantly from the user's actual trajectory due to a number of reasons, including sub-optimal duty-cycling of the positioning sensor, on-device simplification of the sensed trajectory and inaccuracies of the positioning sensors. The error present in the sensed trajectory $\ddot{s}$, given the actual trajectory of motion $\ddot{a}$, can be computed according to a chosen metric, e.g., we use the *time uniform distance* metric $\mathcal{E}_u(t)$, which is defined as:

$$\mathcal{E}_u(t) = d(\ddot{s}(t), \ddot{a}(t)), \forall t \in [a_1.t, a_n.t], \tag{3.3}$$

where $d(\overrightarrow{p_i}, \overrightarrow{p_j})$ is the geodesic distance in meters between the position coordinates $\overrightarrow{p_i}$ and $\overrightarrow{p_j}$, which can be computed using Vincenty's algorithm [138]. The error function $\mathcal{E}_u(t)$ is defined on the entire time domain of the actual trajectory $\ddot{a}$. Now with the time-based error measure for trajectory error, the robustness achieved by a tracking system can be defined as:

> *The rate of violation of tracking error with respect to the requested error bound $E_{Trajectory}$.*

More exactly, the robustness (in percent) of a system $\mathbb{R}_S$ is computed as follows:

$$\mathbb{R}_S = \frac{100}{n} \sum_{i=1}^{n} \mathbb{1}_{(\mathcal{E}_u(a_i.t) > E_{Trajectory})}(\mathcal{E}_u(a_i.t)), \tag{3.4}$$

where, $\mathbb{1}_A(\cdot)$ is the *indicator* function defined as:

$$\mathbb{1}_A(\cdot) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

The smaller the value of $\mathbb{R}_S$, the more accurate the system is in case of trajectory tracking.

3.4 EnTracked$_{RT}$: System Overview

The proposed EnTracked$_{RT}$[4] system mitigates two fundamental problems present in any GPS-based tracking system: (i) rapid depletion of battery life, and (ii) lacking guarantee on obeying the tracking error bound. Thus EnTracked$_{RT}$ allows for continuous and accurate position/trajectory sensing on mobile platforms for a sustained period of time. The main contributions of the EnTracked$_{RT}$ system are as follows:

- The system provides a unified framework for both position and trajectory tracking.
- It introduces a novel compass-based sensor management strategy for energy-efficient tracking. The strategy employes an orthogonal distance function to overcome the effect of noise and the requirement of frequent compass calibrations.
- The system improves overall robustness in tracking by introducing two methods: *robustness coefficient* and *speed adaptation.*
- Extensive cross-platform experiments exhibit superior adaptabity of EnTracked$_{RT}$, while performing trade-off between accuracy requirement and energy expenditure, compared to the state-of-the-art tracking systems.

The EnTracked$_{RT}$ system builds on and enhances capabilities of the previous EnTracked system [65]. Figure 3.2 illustrates the software architecture

[4] RT stands for robust trajectory.

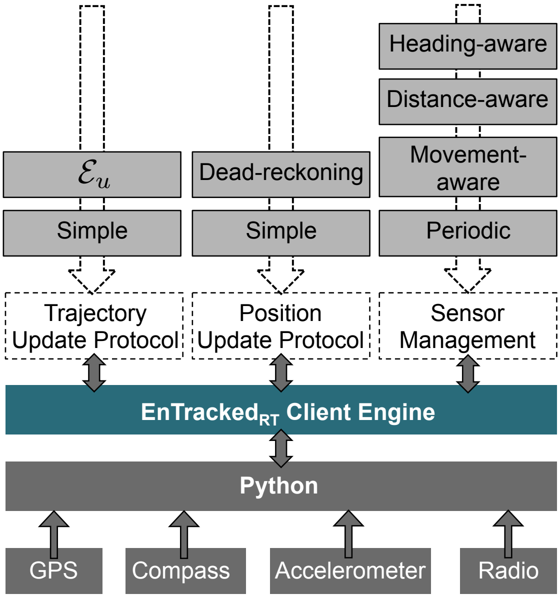


Figure 3.2: Software architecture of the EnTracked$_{RT}$ client engine.

of the EnTracked$_{RT}$ system. The client engine of the EnTracked$_{RT}$ system has been developed in Python, which follows a layered architecture. At the top level, different protocols and sensor management strategies can be plugged into the system. The client engine layer is responsible for maintaining platform integration to retrieve various sensor data through suitable API calls and running the system logic. The EnTracked$_{RT}$ system implements three sensor management strategies, employs an opportunistic data transmission protocol, and uses an on-device trajectory simplification algorithm to minimize the overall energy consumption due to sensing and radio transmission, while maintaining the required level of robustness. In the following, we describe various sensor management strategies, the transmission protocol, and the trajectory simplification employed by the EnTracked$_{RT}$ system in detail.

3.4.1 Sensor Management Strategies

The EnTracked$_{RT}$ system uses on-device accelerometer and compass to estimate uncertainties in tracking and minimize the overall power consumption by optimizing the use of GPS. The current version of the system supports three sensor management strategies: (i) a *movement-aware* strategy that uses the accelerometer to detect non-stationary periods and is responsible

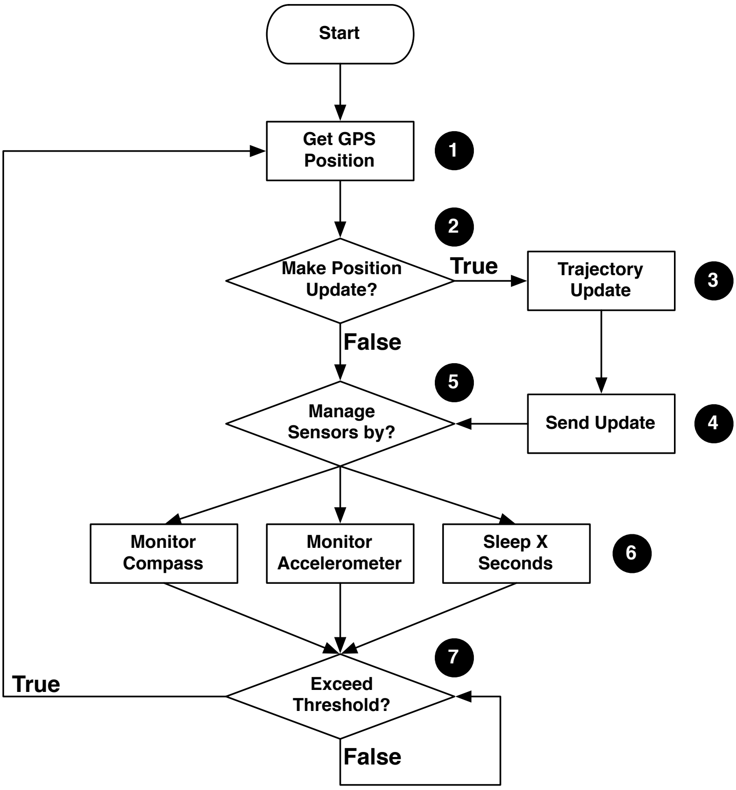


Figure 3.3: Overview of the EnTracked$_{RT}$ sensor management logic.

for triggering other strategies, (ii) a *distance-aware* strategy that dynamically selects the sleeping period for the GPS between two successive location measurements and (iii) a *heading-aware* strategy which uses the compass to detect significant changes in the direction of motion. The EnTracked$_{RT}$ system periodically selects the most suitable sensor management strategy satisfying the underlying tracking requirements. The selection process is based on the estimated power consumption and accuracy of the strategies given the tracking parameters. While estimating the power consumptions of the individual strategies, we take into account sensor- and platform-specific variations, as well as power on/off delays [65]. Strategies which do not comply to the accuracy requirements are assigned a penalty cost. For example, GPS exhibits poor positioning accuracy where a clear view of the sky is not available, e.g., indoors [64, 102] and therefore strategies using GPS should be penalized in those locations. EnTracked$_{RT}$ selects the strategy with the smallest overall cost. Details of the selection process are given in [65].

Figure 3.3 provides an overview of the control flow of the EnTracked$_{RT}$ tracking system. For instance, at the beginning of a tracking period, EnTracked$_{RT}$ requests a GPS location (1). Next, the deployed position update protocol decides if a new position update is to be made (2). If a position update is scheduled, the trajectory update protocol running on the device *piggybacks* novel trajectory data with the position update (3–4). Then, based on the current tracking requirements and the available sensor management strategies, the least power consuming strategy is selected (5). The system then either monitors the accelerometer, or monitors the compass, or sleeps for a certain period of time (6). The process is restarted when the deployed sensor management strategy requests a new GPS position (7). In the following, we describe the sensor management strategies employed by EnTracked$_{RT}$ in detail.

Movement-aware Strategy

The movement-aware strategy is motivated by the fact that when a user is static then there is no need to sample GPS for location information. The strategy uses the on-device accelerometer and the latest speed information obtained from the GPS to robustly detect stationary periods. When the device is detected as stationary, the GPS and the compass sensors are put to sleep to save energy.

To detect stationary periods, we first apply a low-pass filtering on the last 30 acceleration samples and compute the variance on individual axes. If the sum of the variances, obtained from all the three axes, is below a device-specific threshold, we predict the user's state as stationary [65]. Further energy-efficiency is achieved by applying duty-cycling on the accelerometer with a period of:

$$\Delta t = \frac{\min(E_{Trajectory}, E_{Position})}{s_{max}}, \tag{3.6}$$

where s_{max} is the estimated upper bound on the speed of the user, which could be adjusted based on the knowledge of the user's transportation mode. However, the stationary detection running on the mobile device should be accurate to ensure robustness in trajectory tracking where a *false positive* stationary prediction is costlier than a *false negative* prediction. False positive predictions refer to the cases when the user's state is predicted as stationary, but in reality the user is moving. False positives can often decrease the robustness by violating the prescribed error bound. On the other hand, false negative predictions, i.e., the user's state is predicted as non-stationary while the user is static, will not decrease robustness, but will slightly increase the power consumption by scheduling additional GPS sensing. While detecting the user being stationary, employing variance-based approaches using the accelerometer, false positives often arise during commuting by motorized vehicles, e.g., traveling by bus or car with a constant speed on a smooth road. To decrease the number of false positives, we use the speed information from the last GPS measurement to coarsely detect the transportation mode of the user either as *pedestrian* or *motorized*. More specifically, if the user is moving at a speed of over 1.5 m/s (i.e., 5.4 km/h), we consider the transportation mode to be motorized and suspend use of the movement-aware strategy.

Distance-aware Strategy

The main task of the distance-aware strategy, when activated, is to estimate the period of time Δt the GPS receiver can be safely put to sleep in between two successive activations. The strategy uses the latest speed estimate s_{est} obtained from the GPS to dynamically update the sleeping period so that the moving object does not violate the prescribed error bound. A simple estimate for Δt can thus be the time the device would take to move beyond the error bound from the last sensed position. However, the GPS receiver often reports positions with varying accuracies and to take the variations into account, similarly to the work of Farrell et al. [34], we include an error model function e_{model} in the equation and estimate Δt as:

$$\Delta t = \frac{\min(E_{Trajectory}, E_{Position}) - e_{model}}{s_{est}} \tag{3.7}$$

The error model e_{model} combines (i) the uncertainty in the latest GPS measurement u_{GPS}, (ii) the timestamp of the last GPS measurement t_{GPS}, (iii) the estimated speed s_{est} and (iv) the current time t_c as:

$$e_{model} = u_{GPS} + (t_c - t_{GPS})\, s_{est} \tag{3.8}$$

The estimate for u_{GPS} is computed primarily from the *horizontal dilution of precision* (HDOP) value commonly returned by a GPS receiver. While activated, the distance-aware strategy assumes that:

> *The estimated speed s_{est} of the moving object remains constant during the entire sleeping period Δt of the GPS receiver.*

However, the assumption on invariability of speed is restrictive, and it is susceptible to error threshold violation especially in situations when the object starts to accelerate within GPS sleeping periods. To mitigate the limitation, we introduce two concepts: (i) *robustness coefficient* and (ii) *speed adaptation*, which are described in detail in Section 3.5.

Heading-aware Strategy

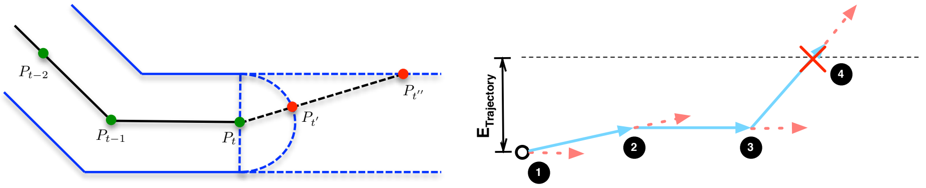
Compared to the previous attempts of developing energy-efficient trajectory tracking systems, one novelty of the EnTracked$_{RT}$ system is the use of the compass as a *turn point* detector. We demonstrate that, in case of trajectory tracking only, the concept of error corridor can significantly reduce power expenditure by monitoring device heading changes with a compass. For example, Figure 3.4(a) shows the difference in GPS activation times between a position and a trajectory tracking system when both systems operate using the same error threshold, i.e., $E_{Position} = E_{Trajectory}$. In the figure, P_t denotes the latest GPS reading location. A position tracking system will schedule the next GPS sensing at location $P_{t'}$, whereas a trajectory tracking system can save energy by further delaying the next GPS sensing till location $P_{t''}$ (with $t' < t''$), assuming the compass did not detect significant changes in the direction of motion.

The heading-aware strategy is motivated by the fact that when an object is traveling along a straight line with constant speed, there is no need to activate the GPS for position sensing. Figure 3.4(b) summarizes the basic idea of the heading-aware strategy. While activated, the strategy keeps track of the accumulated orthogonal distance from the initial direction of motion reported by the GPS. In the figure, the initial direction of motion is shown by the dotted arrow (1). As new compass measurements are obtained, the accumulated orthogonal distance is updated using the estimated speed s_{est} and the change in the direction of motion compared to the initial direction, e.g., (2)–(3). When the accumulated orthogonal distance exceeds the trajectory error threshold $E_{Trajectory}$, a GPS sensing is triggered (4).

Formally, given the trajectory error threshold $E_{Trajectory}$ and a set of noisy compass heading measurements $[\theta_1, \ldots, \theta_n]$, the performance of the heading-aware strategy is governed by an accumulated orthogonal distance function $D_\perp(t)$, which we define as:

$$D_\perp(t_k) = \sum_{i=1}^{k} (t_i - t_{i-1}) s_{est} |\sin(\theta_{start} - \theta_i)|(1 + \sigma), \tag{3.9}$$

where s_{est} and θ_{start} are the estimated speed and the initial direction of motion respectively, obtained from the last GPS measurement. The time when the compass measurement θ_i is recorded is denoted by t_i, with t_0 representing the time of the last GPS fix. Additionally, the definition includes



(a) Position vs. trajectory tracking.
(b) Heading-aware strategy.

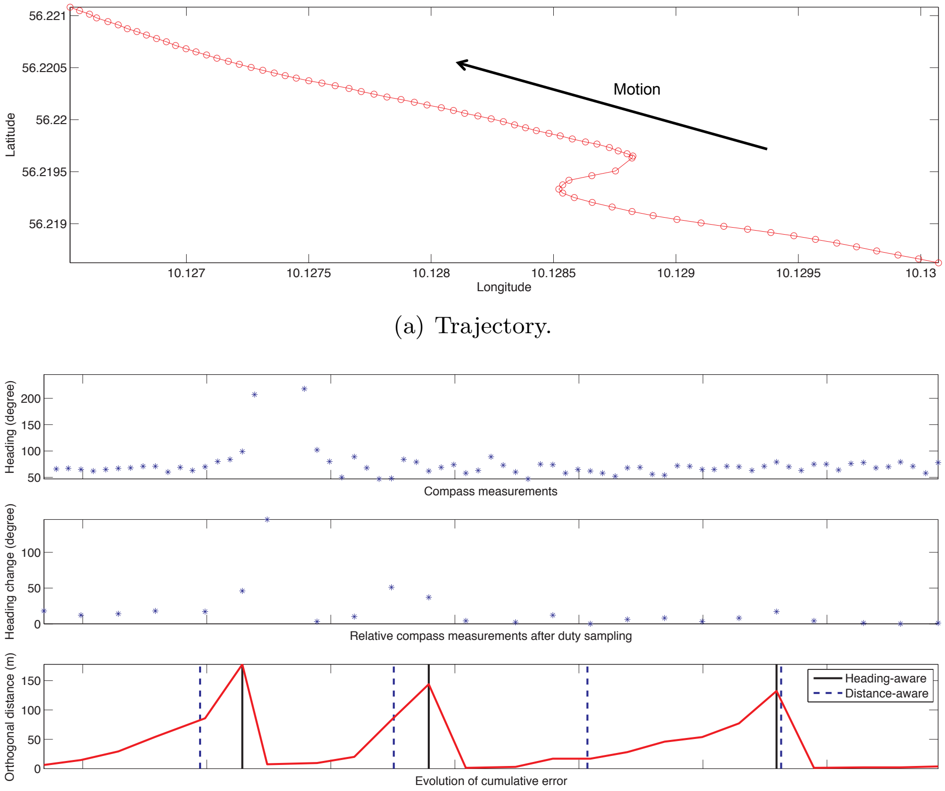
Figure 3.4: (a) Showing the basic difference between position and trajectory tracking. (b) Illustrating the basic idea of the heading-aware strategy. A new position update is requested when the orthogonal distance from the initial direction of motion exceeds the given trajectory threshold.

a parameter $\sigma \in [0,1]$ to compensate for the error in compass measurement. We experimentally set $\sigma = 0.1$. With the definition of $D_\perp(t)$, the heading-aware strategy triggers GPS sensing when $D_\perp(t)$ exceeds $E_{Trajectory}$ and then reinitializes $D_\perp(t) = 0$.

Though the compass provides a cheap way to obtain the heading information of a device with the help of a magnetometer, compass measurements are often noisy and susceptible to electromagnetic inference and disturbance [121]. Additionally, while using the compass for absolute heading measurements, often the compass needs to be calibrated precisely, e.g., as used in positioning and navigation applications [28, 141]. The heading-aware strategy overcomes the frequent need for compass calibration by only considering *relative changes* in heading measurements. For instance, the term $|\sin(\theta_{start} - \theta_i)| \in [0,1]$ in Equation 3.9 only relies on relative change in the direction of motion. Moreover, the function $D_\perp(t)$ is *monotonically increasing*, and when $s_{est} > 0$, a set of noisy compass measurements would rapidly increase the accumulated orthogonal distance and will trigger a GPS sensing sooner. Premature GPS sensing due to noise in compass measurements further helps to maintain robustness in tracking.

As in the distance-aware strategy, we assume that the user's speed remains constant between two successive GPS activations and to save energy we also apply dynamic duty-cycling on the compass. However, we use a much smaller duty-cycle period[5], compared to the distance-aware strategy (e.g., $\Delta t/10$), and provide an upper bound on the GPS sleeping time when using the heading-aware strategy. The upper bound on the GPS sleep period triggers occasional GPS activations, when the moving object is traveling along a straight line for a long period of time. The upper bound is

[5] The sleeping period Δt can be computed as given in Equation 3.7.



(a) Trajectory.

(b) From top: (i) raw heading measurements obtained from the compass, (ii) relative heading changes after duty sampling and (iii) evolution of $D_{\perp}(t)$ (in red) along with time instances of GPS polling by both the heading-aware and the distance-aware strategies.

Figure 3.5: Example of the heading-aware strategy using real-word data.

particularly necessary to ensure reporting the latest location information to a remote server, when position tracking is additionally requested.

To illustrate the performance of the heading-aware strategy, we collected periodic compass and GPS measurements during a bike ride, whose trajectory is shown in Figure 3.5(a). From the top, the plots in Figure 3.5(b) respectively show (i) the raw compass heading measurements recorded during the ride, (ii) the relative change in heading directions after duty-cycling and (iii) the evolution of the $D_{\perp}(t)$ function (in red). For comparison, the bottom-most plot in Figure 3.5(b) also indicates the time instances of GPS activations by the distance-aware strategy.

3.4.2 Transmission Protocol

The sensor management strategies, described in the previous section, optimize energy consumption due to position/trajectory sensing on the mobile platforms. However, often a mobile application running on the device requires updating novel position information to a remote server to fetch or deliver suitable information. A naive transmission protocol that transmits location information as soon as they are sensed by the device would result in high radio cost. Thus, in addition to the sensor management strategies, an opportunistic data transmission protocol is required, which further optimizes the energy consumption due to repeated radio transmissions.

The EnTracked$_{RT}$ system employs a simple and light-weight protocol that uses an adaptive transmission scheme based on the application specific error requirements. The deployed protocol takes into account the power consumption profile of the radio communication module, as the amount of power needed for transmission depends not only on the amount of data being transmitted, but also on the number of connections being established.

The EnTracked$_{RT}$ system optimizes the power consumption due to data transmission by: (i) adopting a batch transmission approach and (ii) performing on-device trajectory simplification. In case of batch transmission, the device maintains a buffer of sensed novel position information and begins transmission when a dead reckoning [143] algorithm running on the device schedules an update. The dead reckoning algorithm keeps track of the deviation in the device's location from the server estimate and triggers an update, when the deviation exceeds the error threshold $E_{Position}$. Further energy and cost of transmission is minimized by reducing the amount of data transmission through on-device trajectory simplification. The trajectory simplification procedure selects a subset of the novel positions for transmission without significantly degrading the motion history. The EnTracked$_{RT}$ system piggybacks the simplified trajectory information, while reporting the latest GPS position to the remote server. The following section describes the on-device trajectory simplification in detail.

3.4.3 Trajectory Simplification

Modern mobile phones can generate position information at a very high rate, e.g., polling the GPS every second would generate around 2.6 million data points over a period of one month. The amount of space required to store trajectory information of a large number of moving objects is thus enormous and the computation burden to process the information is also significant [17]. One attractive solution to both storage and processing

Algorithm 1 The Douglas-Peucker line simplification algorithm.

```
1: procedure TrajectorySimplifyDP($\ddot{a}$, $i$, $j$, $E_{Trajectory}$)
2:     Find the data point $a_k$ farthest from the line $\overline{a_i a_j}$ with distance $dist$.
3:     if $dist > E_{Trajectory}$ then
4:         return TrajectorySimplifyDP($\ddot{a}$, $i$, $k$, $E_{Trajectory}$)
5:         return TrajectorySimplifyDP($\ddot{a}$, $k$, $j$, $E_{Trajectory}$)
6:     else
7:         return $(a_i, a_j)$
8:     end if
9: end procedure
```

problems is the application of spatio-temporal data reduction. The intuition behind GPS-based trajectory data reduction is similar to line simplification, where a location coordinate is eliminated if it can be approximated with reasonable accuracy by interpolating two adjacent coordinates.

The line simplification problem has been extensively studied in the field of digital cartography, as well as in the field of computer graphics. The problem is defined as follows: Given an ordered set of $n + 1$ points $\{p_0, p_1, \ldots, p_n\}$ in a plane, which forms a *polygonal chain* C, i.e., a sequence of n line segments, $\overline{p_0 p_1}, \ldots, \overline{p_i p_{i+1}}, \ldots, \overline{p_{n-1} p_n}$, the line simplification problem asks for a chain $C^{'}$, with fewer segments than n, that represents C well [50].

The EnTracked$_{RT}$ system uses the well known *Douglas-Peucker* line simplification algorithm [31] for trajectory simplification. The Douglas-Peuker algorithm follows a divide-and-conquer paradigm and can be best described recursively [50], see Algorithm 1 for details. In case of trajectory tracking on mobile devices, we adapt the original algorithm and apply it only on the set of novel location measurements present within the device buffer. However, any trajectory simplification introduces error due to approximation, and there could be several measures for quantifying the error. In case of robust trajectory tracking, we use the time uniform distance metric $\mathcal{E}_u$ (see Equation 3.3) as the error measure. The Douglas-Peucker algorithm then selects a subset of the data points such that the error of the simplified trajectory does not exceed the given error threshold $E_{Trajectory}$.

The trajectory simplification algorithm used by the EnTracked$_{RT}$ system begins with the line segment $\overline{a_1 a_n}$, where a_1 and a_n are respectively the first and the last trajectory data points within the buffer. The algorithm then identifies the point a_k farthest from the line segment with respect to the chosen distance metric $\mathcal{E}_u$(line 2). In case the distance of the farthest data point a_k exceeds the error threshold $E_{Trajectory}$, the algorithm adds the point a_k to the simplified list and then it is called recursively for data points $\{a_0, \ldots, a_k\}$ and $\{a_k, \ldots, a_n\}$ (line 4–5). The algorithm terminates when all points of the original trajectory are within a distance $E_{Trajectory}$ from its simplification.

The output of the trajectory simplification is influenced by the presence of outlier data points resulting from poor location estimations. To overcome the problem, we perform a validation on the GPS location measurements prior to the simplification. During the validation, GPS estimates with less than four satellites and an HDOP value greater than 6.0 are neglected [11, 99]. Experiments conducted on real-world datasets exhibit the $\mathcal{O}(n \log n)$ running time of the Douglas-Peucker line simplification algorithm on mobile devices.

3.4.4 Energy Consumption and Robustness

Next we summarize the performance of the EnTracked$_{RT}$ system in terms of energy consumption and robustness. For performance comparisons, we consider the EnTracked system [65], the system of Lange et al. [78], which is the current state-of-the-art for GPS-based tracking, and two variants of our proposed EnTracked$_{RT}$ system namely: periodic and EnTracked$_{RT_{-\mathrm{h}}}$. As sensor management, the periodic system employes continuous GPS sampling with a fixed time period and the EnTracked$_{RT_{-\mathrm{h}}}$ system employs the movement- and distance-aware strategies. Moreover, to measure platform specific variations we run emulation experiments on real-world datasets collected on Android NOne and Nokia N97 devices.

When reporting the average power consumption, we consider two scenarios (i) simultaneous position and trajectory tracking, and (ii) trajectory tracking only. Table 3.1 reports average power consumptions for a fixed position error threshold $E_{position} = 500$ m and varying trajectory error thresholds $E_{Trajectory}$, i.e., the case of simultaneous trajectory and position tracking. See Article III for further details on the experimental setup and the choices of the scenarios. Similarly, Table 3.2 presents the results when only trajectory tracking is requested. Irrespective of the scenarios, the EnTracked$_{RT}$ system achieves the best energy-efficiency on both mobile platforms and for all trajectory error thresholds. The system of Lange et al. [78] and EnTracked exhibit the same power requirements in both scenarios, as they have been designed solely for position tracking and cannot decouple the two tracking tasks.

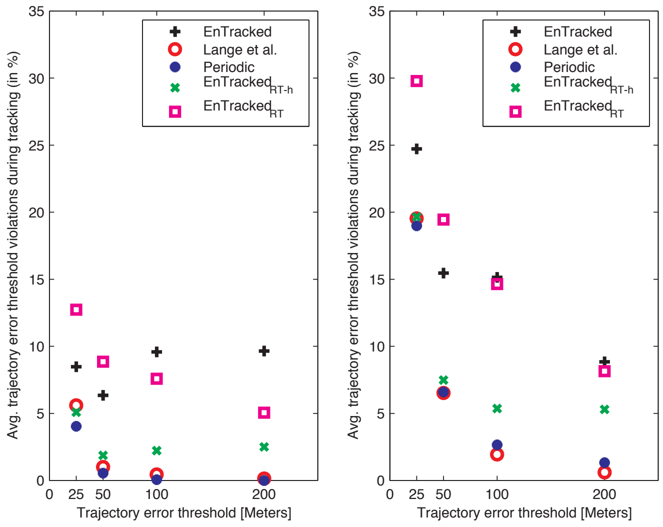
Figure 3.6 illustrates the robustness performance of all the tracking systems on NOne and N97 devices. Though energy-efficient, the robustness performance of the EnTracked$_{RT}$ system, especially for smaller trajec-

	NOne				N97			
	Avg. Energy [mW]				Avg. Energy [mW]			
	25m	50m	100m	200m	25m	50m	100m	200m
EnTracked	373	286	226	184	482	367	275	211
Lange et al. [78]	430	384	350	331	504	417	362	332
Periodic	314	314	314	314	307	307	307	307
EnTracked$_{RT\text{–h}}$	373	286	227	184	482	367	275	212
EnTracked$_{RT}$	**151**	**116**	**94**	**84**	**196**	**160**	**129**	**112**

Table 3.1: Variation of average energy consumption with $E_{Trajectory}$ and fixed $E_{Position} = 500$ m.

	NOne				N97			
	Avg. Energy [mW]				Avg. Energy [mW]			
	25m	50m	100m	200m	25m	50m	100m	200m
EnTracked	373	286	226	184	482	367	275	211
Lange et al. [78]	430	384	350	331	504	417	362	332
Periodic	297	297	297	297	287	287	287	287
EnTracked$_{RT\text{–h}}$	214	159	116	82	263	227	175	121
EnTracked$_{RT}$	**131**	**93**	**71**	**61**	**176**	**135**	**99**	**83**

Table 3.2: Variation of average energy consumption with $E_{Trajectory}$ for trajectory tracking only.



(a) NOne (b) N97

Figure 3.6: Robustness in trajectory tracking.

tory thresholds, is poor. The periodic tracking system and the system of Lange et al. [78] provide the best robustness performance on both mobile platforms. The results also highlight that an attempt to minimize energy consumption solely often sacrifices robustness and hence new measures are needed for performing optimal trade-offs between energy consumption and robustness. In the following section we describe features of the EnTracked$_{RT}$ system that help to improve the overall robustness without significantly sacrificing energy-efficiency.

3.5 Robustness

Contrary to the majority of the existing tracking systems, which only focus on energy-efficiency, the EnTracked$_{RT}$ system provides additional means for the LBS to control the robustness during tracking. The current implementation of EnTracked$_{RT}$ supports two mechanisms, application of a robustness coefficient and velocity adaptation, to ensure improved robustness at the expense of slightly higher energy consumption.

3.5.1 Robustness Coefficient

One way to improve robustness in tracking is to sample GPS positions relatively frequently. In other words, when the distance- and heading-aware strategies are running, triggering GPS sensing prematurely, i.e., activating before the scheduled time, can improve overall robustness. This can be achieved simply by allowing a shorter sleeping period for GPS as estimated by the sensor management strategies. Accordingly, we define a robustness coefficient $R_c \in (0, 1]$ to internally use an error threshold $E^*_{Trajectory}$ that is smaller than the prescribed threshold $E_{Trajectory}$. More specifically, the internal threshold $E^*_{Trajectory}$ relates to $E_{Trajectory}$ as:

$$E^*_{Trajectory} = R_c \cdot E_{Trajectory} \tag{3.10}$$

The robustness coefficient influences the GPS sleeping time approximately[6] linearly and a small non-zero value for R_c thus allows smaller GPS sleeping periods and consequently reduces the rate of error threshold violations. Hence, an application can select an appropriate vale for R_c to trade-off between overall energy consumption and robustness[7]. Interestingly, the movement-aware strategy is not influenced by the robustness coefficient as

[6] The error model term in Equation 3.7 and the dynamic heading changes due to the underlying motion in Equation 3.9 violate the linear relationship.

[7] If not specifically mentioned, robustness coefficient is initialized as $R_c = 1$.

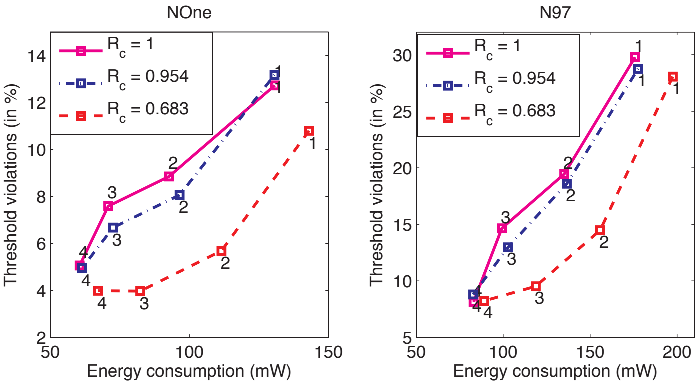


Figure 3.7: Influence of robustness coefficient R_c on robustness and energy consumption for different trajectory error thresholds (1 = 25 m, 2 = 50 m, 3 = 100 m and 4 = 200 m) on two mobile platforms.

it is only activated when the user is stationary. In the evaluation we only consider two values[8] for R_c, i.e., $R_c = 0.954$ and $R_c = 0.683$. Figure 3.7 illustrates that the robustness coefficient can trade-off between overall energy consumption and robustness. For comparison the plots also include the case $R_c = 1$ as a baseline.

3.5.2 Speed Adaptation

The estimated current speed of the user plays an important role while determining the sleeping period of the GPS, e.g., Equations 3.7, 3.8 and 3.9 all include the user's current speed estimation s_{est}. Hence, the validity and accuracy of the speed estimate obtained from GPS influence the overall robustness of the tracking system.

As mentioned before, the assumption that the user's speed remains constant during the GPS sleeping period is restrictive and prone to error threshold violations. The assumption works relatively well for small values of the tracking error threshold and during pedestrian transportations. However, during motorized transportation, the assumption often fails due to the significant dynamic changes observed in the user's speed [49]. To improve the overall robustness, we introduce speed buffering as a method to enforce a lower bound on the current speed estimate of the user based on his/her motion history. Specifically, the EnTracked$_{RT}$ system accepts a

[8] Exemplary values for R_c are chosen based on the three-sigma rules of a normal distribution. `http://en.wikipedia.org/wiki/68-95-99.7_rule` [Retrieved: July 17, 2014]

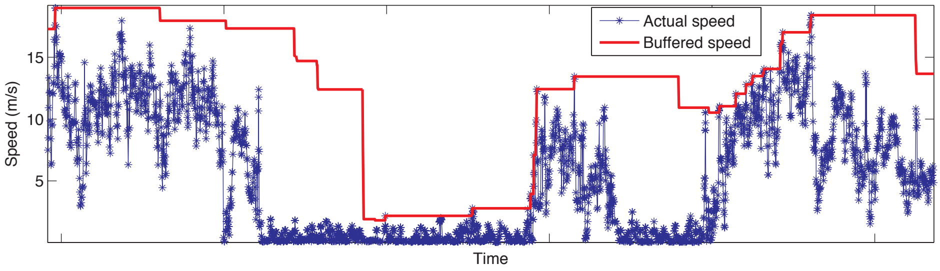


Figure 3.8: Effect of speed buffering on a user's current speed estimations when using a buffer width of 5 minutes, i.e., $B_w = 300$ seconds.

buffer width parameter B_w (in seconds) and a function that can be applied on the buffered speed values to obtain a better speed estimate based on the user's recent acceleration and deceleration profile. The buffered speed estimate can be computed as:

$$s^*_{est} = \mathcal{F}(\{s^t_{est}\}^{t_c}_{t=t_c-B_w}), \tag{3.11}$$

where t_c is the current time and s^t_{est} is the speed estimate obtained from GPS at time t. The role of the function $\mathcal{F}(\cdot)$ is to allow relatively smaller sleeping periods for the GPS receiver exploiting situational speed history. It can be achieved by extracting various high percentiles of the buffered speed values. The EnTracked$_{RT}$ system currently uses the *maximum* value from the speed buffer as an estimate for the user's current speed to be used by the distance- and heading-aware strategies. For example, Figure 3.8 illustrates the effect of speed buffering ($B_w = 300$ s) on current speed estimates of the user. The maximum function overestimates the current actual speed of the user and allows for smaller GPS sleeping time compared to the case when no buffering is employed. The increased amount of GPS activations thus improves the overall robustness in tracking with little increase in overall energy consumption. Contrary to the robustness coefficient, the speed buffering also influences the movement-aware strategy. For instance, the speed estimates, based on a user's motion history, help to distinguish between pedestrian and other transportation modalities accurately. Figure 3.9 illustrates the effect of velocity buffering on overall robustness and energy consumption. For comparison the plots also include the case when no velocity buffering is employed, i.e., $B_w = 0$ s.

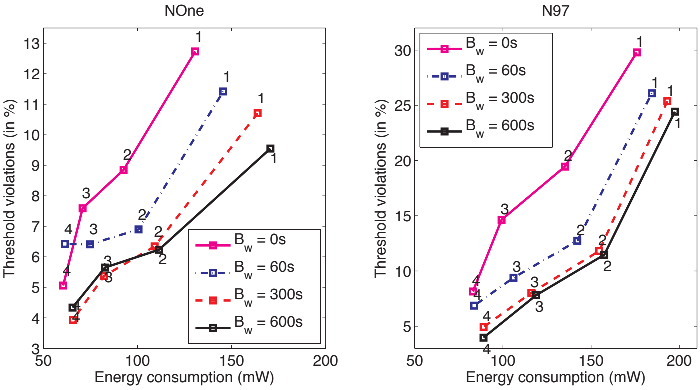


Figure 3.9: Influence of the velocity buffer width B_w on robustness and energy consumption for different trajectory error thresholds (1 = 25 m, 2 = 50 m, 3 = 100 m and 4 = 200 m) on two mobile platforms.

3.6 Energy and Robustness Trade-off

The vast number of LBS present in the application stores vary significantly in terms of their energy and robustness requirements. For instance, an application for navigation would assign more importance to the accuracy, whereas a micro-blogging application puts more emphasis on the energy-efficiency. To capture the performance of a tracking system under these diverse requirements, we measure the *utility* of a tracking system U_S by defining the following function:

$$U_S = p(1 - \frac{\mathbb{R}_S}{100}) + (1-p)(1 - \frac{\mathbb{E}_C}{500}), \tag{3.12}$$

where $\mathbb{R}_S$ is the robustness achieved by the system (see Equation 3.4) and $\mathbb{E}_C$ is the average power consumption of the system in mW. The parameter $p \in [0, 1]$ in the above equation can be used to assign different weights on the requirements of robustness and consequently on the energy consumption. The normalizing factors are values in the same unit as the numerators (e.g., 100% for $\mathbb{R}_S$ and 500 mW for $\mathbb{E}_C$), which make U_S a *unit-less* quantity. Moreover, the normalizing factors penalize the tracking system if either $\mathbb{R}_S$ or $\mathbb{E}_C$ is high. For example, if a tracking system frequently violates the error threshold then $\mathbb{R}_S$ will be close to 100, and thus $1 - \frac{\mathbb{R}_S}{100}$ will be close to *zero*. Contrary to the case of robustness, a much higher normalizing constant (500 mW) is used for the energy consumption, which is an upper bound on the average power consumptions observed in our experiments. Hence, under equal importance of robustness and energy consumption, i.e., $p = \frac{1}{2}$,

	NOne				N97			
with $p = \frac{1}{3}$	25m	50m	100m	200m	25m	50m	100m	200m
EnTracked	0.47	0.60	0.67	0.72	0.28	0.46	0.58	0.69
Lange et al.	0.41	0.49	0.53	0.56	0.26	0.42	0.51	0.56
Periodic	0.59	0.60	0.60	0.60	0.55	0.59	0.61	0.61
EnTracked$_{RT_{-\mathrm{h}}}$	0.70	0.78	0.84	0.88	0.58	0.67	0.75	0.82
EnTracked$_{RT}$	**0.78**	**0.85**	**0.88**	**0.90**	**0.67**	**0.76**	**0.82**	**0.86**
with $p = \frac{1}{2}$	25m	50m	100m	200m	25m	50m	100m	200m
EnTracked	0.58	0.68	0.73	0.77	0.40	0.56	0.65	0.74
Lange et al.	0.54	0.61	0.64	0.67	0.40	0.55	0.63	0.67
Periodic	0.68	0.70	0.70	0.70	0.62	0.68	0.70	0.71
EnTracked$_{RT_{-\mathrm{h}}}$	0.76	0.83	0.87	**0.91**	0.64	0.74	0.80	0.85
EnTracked$_{RT}$	**0.81**	**0.86**	**0.89**	**0.91**	**0.68**	**0.77**	**0.83**	**0.88**
with $p = \frac{2}{3}$	25m	50m	100m	200m	25m	50m	100m	200m
EnTracked	0.70	0.77	0.79	0.81	0.51	0.65	0.72	0.80
Lange et al.	0.68	0.74	0.76	0.78	0.53	0.68	0.75	0.78
Periodic	0.78	0.80	0.80	0.80	0.68	0.76	0.79	0.80
EnTracked$_{RT_{-\mathrm{h}}}$	0.82	**0.88**	**0.91**	**0.93**	**0.69**	**0.80**	**0.85**	**0.89**
EnTracked$_{RT}$	**0.83**	**0.88**	0.90	**0.93**	0.68	0.78	0.84	0.88

Table 3.3: Utility scores for different trade-offs between energy and robustness: with energy being twice as important ($p = \frac{1}{3}$) , equally important ($p = \frac{1}{2}$), and half as important ($p = \frac{2}{3}$) as robustness, respectively.

a change of 5 mW in energy consumption is equivalent to a 1% change in robustness. Table 3.3 illustrates the utility scores of all the tracking systems studied in our experiments for three different tracking scenarios. Additionally, Table 3.4 illustrates that the use of the robustness coefficient and the velocity buffering helps the EnTracked$_{RT}$ system to achieve the highest utility score on both mobile platforms and for all error thresholds, when robustness is twice as important as the energy-efficiency (i.e., $p = \frac{2}{3}$).

3.7 Discussion

The heading-aware strategy employed by the EnTracked$_{RT}$ system is agnostic to the sensor type used for the heading tracking and will work, without requiring any modifications, with a more accurate sensor for heading information, e.g., gyroscope. However, a drawback of using a gyroscope is its high power requirement. For example, on an Android platform (Sam-

	NOne			
	25m	50m	100m	200m
EnTracked$_{RT}$, $R_c = 0.683$, $B_w = 60$	**0.84**	**0.89**	**0.92**	**0.93**
	N97			
	25m	50m	100m	200m
EnTracked$_{RT}$, $R_c = 0.683$, $B_w = 300$	**0.69**	**0.80**	**0.86**	**0.90**

Table 3.4: Utility scores of EnTracked$_{RT}$, when robustness is twice as important as energy, i.e., $p = \frac{2}{3}$.

sung Galaxy SII) the gyroscope requires on an average 130 mW compared to a mere 48 mW by the compass. Hence, to maintain the overall energy consumption at a low level we use the compass as the heading sensor.

As demonstrated before, the orthogonal distance function $D_{\perp}(t)$ helps to overcome the noise often present in compass measurements. However, our extensive experiments on real-world datasets indicate that the robustness performance of the heading-aware strategy suffers especially in the case of motorized transportations (for details see Article III). This is due to the bias of the magnetometer measurements towards the electromagnetic field of the vehicle [19]. This situation can be easily identified using magnetic field anomaly detection, e.g., using an entropy-based measure on the compass data. Under such anomaly situations, the sensor manager of the EnTracked$_{RT}$ system should not rely on the heading-aware strategy and this can be easily achieved by assigning a penalty score to the heading-aware strategy. We keep this improvement of the heading-aware strategy for our future release of the EnTracked$_{RT}$ system.

The sensor management strategies can be adapted based on the current battery level, e.g., using $R_c > 1$ when battery is low. The simple adaptation based on the current battery level is also part of our future release. Lastly, the utility objective can be integrated in the sensor management decision making and is also left as a future extension of the EnTracked$_{RT}$ system.

Part II

Activity Recognition

Chapter 4

Activity Recognition: Learning from Unlabeled Data

Identifying a user's physical activities such as 'walking', 'running', 'opening a door', 'taking a bus', and 'steering a wheel' using sensors, e.g., accelerometer and gyroscope, is a major research area within ubiquitous and pervasive computing [7, 76]. Accurate knowledge of a user's current activity, e.g., whether a person is moving, also plays an important role in the development of a highly energy-efficient position/trajectory tracking system such as the EnTracked$_{RT}$. Other prominent application areas of activity recognition include industrial training sectors [92, 122], smart homes [9, 84, 106], sports and entertainment [71, 73], situated support [53], and monitoring of mental health and physical wellbeing [27, 80, 105, 109]. Lately, activity recognition has been successfully used in commercial game consoles such as Nintendo Wii.

In line with general pattern recognition systems, a typical activity recognition pipeline consists of the following stages [16]: (i) *data acquisition*, (ii) *preprocessing*, (iii) *frame extraction* or *segmentation*, (iv) *feature extraction*, and (v) *classification*. An illustration of the pipeline is given in Figure 4.1. The classification stage of the pipeline operates in two different modes. In training mode, the extracted features and the ground truth labels are used as inputs to train a classification model. Whereas in prediction mode, the previously trained model is used to infer the class of a set of features.

Common approaches to activity recognition predominantly follow the well-established supervised learning paradigm [12, 36, 123, 124], where heuristic, domain specific, and handcrafted features are first extracted from the sensor data, and then combined with a large amount of ground truth annotations a classifier is trained. Examples of supervised learning algorithms, popular in activity recognition research, include Nearest

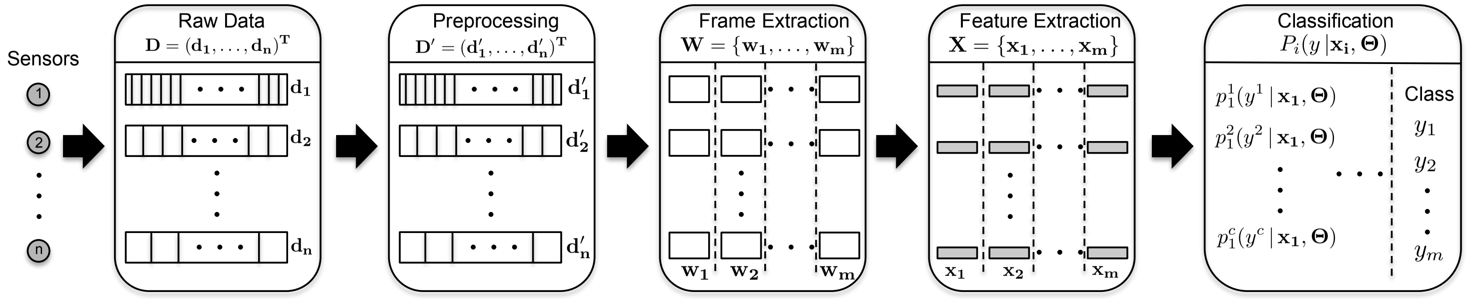

Sensors
1
2
n
Raw Data
$\mathbf{D} = (\mathbf{d_1}, \ldots, \mathbf{d_n})^\mathbf{T}$
$\mathbf{d_1}$
$\mathbf{d_2}$
$\mathbf{d_n}$
Preprocessing
$\mathbf{D'} = (\mathbf{d'_1}, \ldots, \mathbf{d'_n})^\mathbf{T}$
$\mathbf{d'_1}$
$\mathbf{d'_2}$
$\mathbf{d'_n}$
Frame Extraction
$\mathbf{W} = \{\mathbf{w_1}, \ldots, \mathbf{w_m}\}$
$\mathbf{w_1}$
$\mathbf{w_2}$
$\mathbf{w_m}$
Feature Extraction
$\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$
$\mathbf{x_1}$
$\mathbf{x_2}$
$\mathbf{x_m}$
Classification
$P_i(y\,|\,\mathbf{x_i}, \boldsymbol{\Theta})$
$p_1^1(y^1\,|\,\mathbf{x_1}, \boldsymbol{\Theta})$
$p_1^2(y^2\,|\,\mathbf{x_1}, \boldsymbol{\Theta})$
$p_1^c(y^c\,|\,\mathbf{x_1}, \boldsymbol{\Theta})$
Class
y_1
y_2
y_m

Figure 4.1: Example of a typical activity recognition pipeline to recognize activities using n different sensors. An activity recognition pipeline is composed of [16]: (i) data acquisition, (ii) preprocessing, (iii) frame extraction (or segmentation), (iv) feature extraction, and (v) classification stages.

Neighbors [134], Decision Trees [9, 84], Naive Bayes [127], Support Vector Machines (SVM) [57], Hidden Markov Models (HMM) [80], and various Boosting techniques [113]. A successful deployment of an activity recognition system thus requires to perform the *bootstrapping* of the recognizer efficiently, i.e., (i) obtaining a good feature representation for the sensor measurements and (ii) collecting a large amount of ground truth annotations.

Recent technological developments in sensing abilities and miniaturization of sensors have catalyzed a rapid growth in the number of sensors readily available on modern smartphones. As mobile phones are typically carried around by people during their everyday activities, sensor-rich mobile platforms have become an attractive choice of performing continuous activity recognitions. However, existing supervised learning-based approaches to activity recognition are not well suited for the mobile platforms due to two main limitations. Firstly, the overall performance of the recognizer is, in general, highly sensitive to the feature set used, and often the optimal feature set varies depending on the target activities [36, 59, 66]. Therefore, the generalizability of such approaches across application domains and sensor modalities is limited. Secondly and more importantly, a large amount of reliable annotation or ground truth data is required to obtain adequate classification performance, which is difficult, costly, time-consuming, and error prone to collect [16, 56, 77, 125].

Often annotations for activities are obtained through self-reporting by users. While going about their everyday activities, people do not pay much attention to the phones or their placements (in the pocket, in the backpack etc.), and they sporadically interact with the device for answering a call or explicitly using a service (e.g., information retrieval). Thus, getting active support from users on annotating real-life sensor measurements is not feasible. Moreover, privacy and ethical considerations hinder collecting ground truth data from real-life situations (outside the laboratory) through direct observations made by an observer or through video recordings. Hence, activity recognition applications targeted at real-world deployments suffer from the lack of opportunity to collect ground truth information directly. Other alternatives for collecting ground truth information include *experience-sampling*, i.e., prompting the user to label the current or previous activities [9, 123], opportunistic use of other sensors, e.g, mobile phone cameras and microphones [84], activity diary maintained by the users [58], and a combination of these approaches. The indirect methods alleviate the ground truth collection problem to some extent and can be used to collect ground truth for a smaller subset of the unlabeled data. However, the ground truth information obtained by an indirect method is often prone to errors.

Despite the difficulties in collecting ground truth information, collecting a large amount of unlabeled data is straightforward. For example, a script can be run on a mobile phone to record sensor measurements continuously in the background. Moreover, the absence of requirements to follow scripted activity patterns or protocols during the unlabeled data collection period allows users to perform natural activities without necessarily being conscious about the underlying data collection. Hence, major challenges in mobile activity recognition under poor availability of ground truth data are:

> *What insights can be learned from an easily obtainable large amount of unlabeled sensor measurements? Can we exploit the insights to improve the overall performance of an activity recognizer?*

In this chapter we propose a novel framework for activity recognition that overcomes the requirement of a large amount of ground truth data by learning characteristic patterns present in human motion from unlabeled sensor measurements. Based on the *self-taught learning* paradigm [110], we develop a *sparse-coding* algorithm for representing sensor data obtained during human physical motion. The unsupervised learning of the sensor data representations also helps the proposed activity recognition framework to overcome the time consuming and heuristic task of *feature-engineering*, thereby making the bootstrapping process for activity recognizers easy.

We extend the sparse-coding method, which was mainly developed for images and audio signals [43], by introducing a *codebook* selection technique. The selection process primarily reduces redundancy present in the codebook and improves the running time of the feature extraction method on resource constrained mobile platforms. The features, computed using the learned codebook, are then used to train a classifier, which requires a significantly smaller amount of ground truth measurements than a supervised learning technique. Lastly, we show the easy generalizability of the proposed activity recognition framework across different sensor modalities. For example, the proposed algorithm works well on accelerometer and gyroscope without requiring any modeling changes.

The content of this chapter is based on Article IV.

4.1 Approaches to Activity Recognition

The main objective of the work, described in this chapter, is to develop a generalizable and feature-engineering free framework that utilizes a vast amount of easily collectable unlabeled sensor data to augment training of an activity recognizer for real-world deployment. So far, very little work can be found that utilizes unlabeled data in activity recognition research. However, incorporation of unlabeled data and automatic feature representations have been studied extensively in general machine learning research. Accordingly, we divide the relevant related work into non-supervised learning and feature learning-based approaches.

4.1.1 Non-supervised Learning

Semi-supervised learning [21] approaches focus on learning from both unlabeled and labeled data. The most well-known semi-supervised learning algorithms are based on *generative models*, where the unknown data distribution $p(\boldsymbol{x})$ is modeled as a mixture of class-conditional distributions $p(\boldsymbol{x} \mid y)$, where y is the unobserved class variable. The *Expectation Maximization* (EM) algorithm can be used to learn the mixture components from the unlabeled and the labeled datasets. The predictive distribution, i.e., $p(y \mid \boldsymbol{x})$ can be computed by applying the Bayes theorem. Other approaches of semi-supervised learning include *self-training*, *co-training*, *transductive* SVM (TSVM), *graphical models*, and *multiview learning* [21, 147][1].

Semi-supervised learning has been successfully applied in activity recognition research. For example, Guan et al. [44] propose an *ensemble* method based on co-training (En-Co-Training) to identify locomotion-related activities such as, ‘lying’, ‘sitting’, ‘standing’, and ‘walking’ from measurements obtained from accelerometers attached to the trousers. Stikic et al. [125] follow a feature-engineering-based approach combined with self-training [21] and co-training [14] to identify activities within smart homes. Moreover, the authors study *active learning* [97] strategies to reduce the amount of ground truth annotations needed. However, in order to be effective, a semi-supervised learning approach requires a number of assumptions to hold [21, 97, 98]. The strongest assumption made by a semi-supervised learning approach is that the unlabeled and the labeled datasets are drawn from the same distribution, i.e., $\mathcal{D}^u = \mathcal{D}^l$ [104]. In other words, the set of activities present within the unlabeled dataset should match the target activities of the recognizer. This requirement substantially limits the applicability and makes the semi-supervised learning approach error prone

[1] Details are out of scope of this thesis.

in real-world scenarios, where users perform extraneous activities or no activity at all [122]. The proposed sparse-coding framework for activity recognition relaxes the equality requirement and allows different data distributions for the unlabeled and the labeled datasets, i.e., $\mathcal{D}^u \neq \mathcal{D}^l$.

Allowing for different data distributions is also related to another popular machine learning technique known as *transfer learning* [104]. Transfer learning aims at helping to improve a learning task in one domain from the knowledge of a similar learning task in another domain. It acts as a bridge between different application domains with different target classes or sensing modalities [18]. Hu et al. [56] apply transfer learning to identify novel activities within smart homes under the assumption that (i) a mapping exists between the source and target domain activities and (ii) the domains share an identical feature space. Kasteren et al. [62] employ the use of a meta-feature space, a common feature space for various source domains, to transfer model parameters in order to identify the same set of activities of daily living in a target home. Lane et al. [77] propose the Community Similarity Network (CSN) to personalize an activity classifier for a user by selectively recruiting training data from other users with similar traits. However, transfer learning techniques applied in activity recognition research often do not directly reduce the amount of ground truth annotations required. The data gathering task is shifted to other domains where model training is easier.

Active learning [97] is another approach that optimizes the need for ground truth annotations. In active learning the main focus is given to obtaining ground truths for the most uncertain instances, such that the new information would help the training phase the most. The instances for which manual labels are requested are obtained automatically by applying an information theory criterion. Active learning strategies have been applied by Alemdar et al. [3] and Stikic et al. [125] to identify activities within smart homes. Liu et al. [83] combine a $C4.5$ decision tree-based classification with active learning to identify, e.g., ‘vacuuming’, ‘folding laundry’, and locomotion activities from accelerometers attached to the hip and the dominant wrist. However, one drawback of the active learning is that it works on a pre-selected feature set, and hence is sensitive to the feature-engineering process. In contrast, the sparse-coding framework learns the feature representations automatically and thus has better generalizability.

Multi-instance learning has also been applied in activity recognition research to alleviate the requirement of ground truth annotations. Contrary to the standard supervised learning techniques, where each instance within the training data is assigned a label, multi-instance learning techniques only assign a label to a set or a bag of instances [4]. The objective of the learner is then to predict the labels of previously unseen bags. Stikic et al. [123, 124] propose the use of multi-instance SVM [5], where labels of the instances are considered hidden variables and a standard SVM is trained to minimize the expected loss in classification using the labels of the bags. Similar to the active learning approaches, multi-instance learning also operates on a pre-selected feature set and thus has limited generalizability.

4.1.2 Feature Learning

Activity recognition approaches typically extract handcrafted, domain-specific features from the raw sensor measurements. A typical example is to compute, e.g., statistical and frequency domain properties from analysis windows extracted from continuous time-series data using, e.g., a sliding-window procedure [36]. However, the predefined feature set optimal for one domain may not be suitable for another similar application domain, e.g., when the target activities [59, 66] or the sensor type change. Therefore, developers are often required to start from scratch to find a good data representation or feature set while deploying an activity recognizer to a different application domain.

One solution to overcome the sensitivity of traditional and domain specific feature-engineering is to learn a meaningful data representation automatically from the data. For example, an unlabeled dataset can be utilized to learn a compact and meaningful representation of sensor measurements, and automatic feature learning is an active area of research in the machine learning community [26]. An important objective of a feature learning procedure is to identify regularities present in the data [52]. However, learning automatic features for activity recognition in particular has been a less explored area, and only recently a few attempts have been made to learn features from accelerometer measurements.

Among early attempts, Mäntyjärvi et al. [90] compared the use of Principal Component Analysis (PCA) [12] and Independent Component Analysis (ICA) [60] to automatically learn features from sensor data. After applying either PCA or ICA to the raw acceleration measurements, the authors apply a sliding window procedure to extract measurement windows. Later, Wavelet-based [88] features are extracted from the measurement windows, and a multilayer perceptron is trained for activity inference. Plötz et al. [107] employ PCA to learn features from tri-axial accelerometer frame windows. However, instead of applying PCA to raw sensor measurements, the authors apply an Empirical Cumulative Distribution Function (ECDF) -based normalization [47] to modify the frames. Additionally, the authors investigate the use of Restricted Boltzmann Machines (RBM) [52] to train an autoencoder network for feature learning. Contrary to the PCA-based feature learning, the sparse-coding-based approach allows deviating from a purely linear input-output relationship, thereby enabling sparse-coding to capture complex non-linear relationships present within the data. Moreover, the upper limit of the feature-space dimension in the PCA-based approach is bounded by the effective dimension of the input data, whereas sparse-coding can extract an over-complete feature set, i.e., the feature-space dimension is higher than the input data dimension. An over-complete feature set shows more resilience to noise and sensor degradations [100].

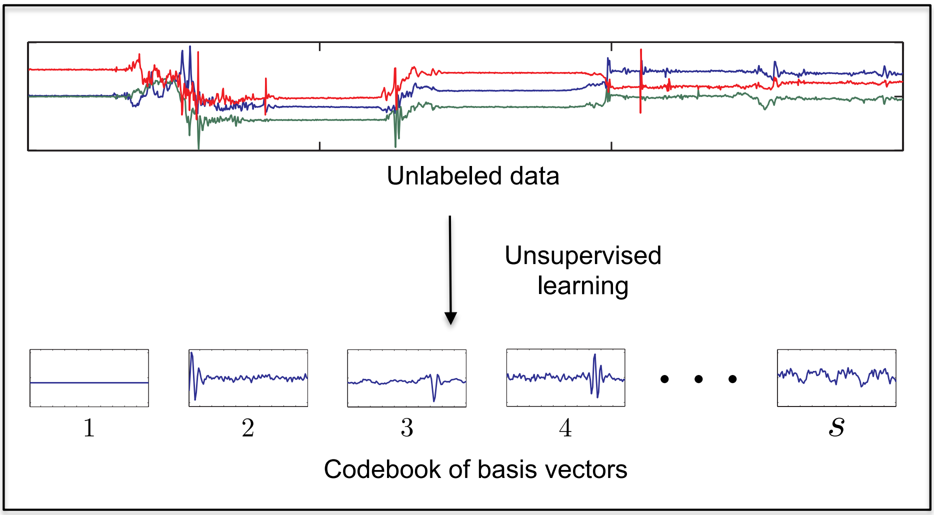
Minnen et al. [95] consider activities as sparse motifs and propose an unsupervised algorithm to extract such motifs from multi-dimensional time series data. Frank et al. [39] propose a related approach that applies time-delay embeddings to learn features from sensor measurement frames and use the extracted features to train a classifier for activity inference.

Contrary to the popular Fourier and Wavelet representations, the sparse-coding is a data adaptive approach [55], i.e., it is tailored to the statistics of the data and the basis vectors are learned directly from the data itself. Other examples of data-adaptive approaches include PCA, ICA, and matrix factorization.

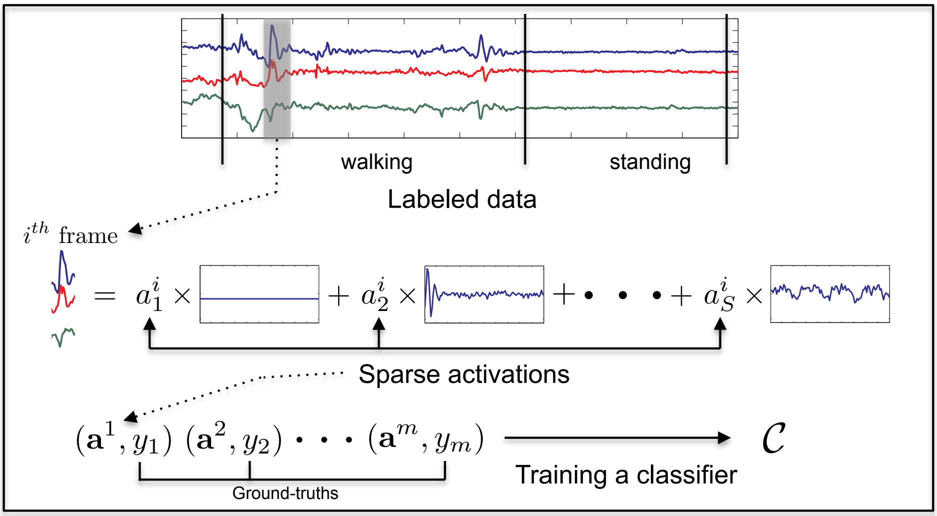
4.2 Activity Recognition Framework

The first step in our activity recognition framework is to collect a large quantity of unlabeled sensor measurements. In case of an accelerometer-based activity inference on the mobile phone, unlabeled data can be collected simply by logging measurements continuously from the inbuilt tri-axial accelerometer[2]. We then learn a codebook, comprising of a set of *basis vectors*, from the unlabeled dataset (see Section 4.2.1). The codebook captures characteristic patterns present in human movements that can be observed from the sensor measurements. Figure 4.2(a) provides an overview of the codebook learning phase. Once the codebook is learned and optimized, we transform the signal measurements to the feature space, *spanned* by the basis vectors, through sparse representations (see Section 4.2.3). Lastly, extracted features are used to train an activity classifier with a small amount of labeled measurements. Figure 4.2(b) illustrates the overall recognition process. Below we provide a detailed description of the proposed activity recognition framework.

[2] This holds also for other sensors.



(a) Phase I: Learning a codebook of basis vectors from an unlabeled dataset.


walking
standing
Labeled data
i^{th} frame
$= a_1^i \times + a_2^i \times + \cdots + a_S^i \times$
Sparse activations
$(\mathbf{a}^1, y_1)\ (\mathbf{a}^2, y_2) \cdots (\mathbf{a}^m, y_m)$
Ground-truths
Training a classifier
$\mathcal{C}$

(b) Phase II: The learned codebook (Phase I) is used to extract features from a small amounts of training data to train a classifier for activity inference.

Figure 4.2: Overview of the proposed sparse-coding-based activity recognition framework.

4.2.1 Codebook Learning from Unlabeled Data

To learn a codebook from sequential sensor measurements, we apply a sliding window procedure, a popular approach in activity recognition research [36], to extract overlapping and fixed-length frames. Sensor measurements belonging to the i^{th} frame are represented by the vector $\boldsymbol{x}_i \in \mathcal{R}^n$, where n is the number of measurements in the frame. The vector $\boldsymbol{x}_i$ can be constructed by concatenating the measurements observed by the three axes of the sensor and for the same frame i, i.e.:

$$\boldsymbol{x}_i = \{A_i^X(1), \ldots, A_i^X(w), A_i^Y(1), \ldots, A_i^Y(w), A_i^Z(1), \ldots, A_i^Z(w)\}, \quad (4.1)$$

where $A_i^X(j), A_i^Y(j)$, and $A_i^Z(j)$ represent the j^{th} measurement of the i^{th} frame recorded by the axes A^X, A^Y and A^Z respectively and w is the frame-width[3]. Vector $\boldsymbol{x}_i$ can also be constructed by aggregating the measurements from the axes. For example, a common approach in accelerometer-based sensing is to construct $\boldsymbol{x}_i(j)$ by taking the magnitude of the acceleration components recorded on the three axes:

$$\boldsymbol{x}_i(j) = \sqrt{A_i^X(j)^2 + A_i^Y(j)^2 + A_i^Z(j)^2} \quad (4.2)$$

The magnitude-based sensor data aggregation is often employed to achieve *rotation invariance*, i.e., making sensor measurements less sensitive to the orientation of the phone, for smartphone-based activity recognition [85, 115, 139].

Once the measurement frames are extracted, the unlabeled dataset is defined as:

$$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K\}, \ \boldsymbol{x}_i \in \mathcal{R}^n \quad (4.3)$$

where, K is the total number of frames obtained. We use the unlabeled dataset $\boldsymbol{X}$ to learn a codebook $\boldsymbol{\mathcal{B}}$ that captures latent and characteristic patterns present in the sensor measurements. Codebook $\boldsymbol{\mathcal{B}}$ contains S basis vectors, i.e., $\boldsymbol{\mathcal{B}} = \{\boldsymbol{\beta_j}\}_{j=1}^S$, where each basis vector[4] $\boldsymbol{\beta_j} \in \mathcal{R}^n$ represents a pattern. Once the codebook has been learned, any frame of sensor measurements can be accurately approximated as a linear superposition of the learned basis vectors, i.e.,

$$\boldsymbol{x}_i \approx \sum_{j=1}^{S} a_j^i \boldsymbol{\beta}_j, \quad (4.4)$$

[3] Since we consider tri-axial accelerometer measurements, we have $n = 3\,w$.

[4] The dimension of a basis vector $\boldsymbol{\beta}_j$ is identical to the dimension of a frame of measurements $\boldsymbol{x}_i$.

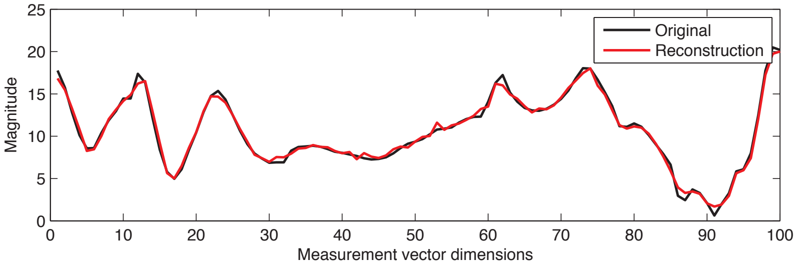


Figure 4.3: Reconstruction of an accelerometer measurement frame using 54 basis vectors from a codebook comprising 512 basis vectors.

where a_j^i is the activation for the j^{th} basis vector when representing the measurement vector $\boldsymbol{x}_i$. Moreover, while representing frames of sensor measurements using basis vectors, we look for a sparse solution, where a very few basis vectors contribute to the accurate reconstruction of the frame. In other words, the activations of the majority of the basis vectors during the reconstruction are zero, i.e., $\boldsymbol{a^i}$ is a sparse vector. Figure 4.3, for example, illustrates the reconstruction of a measurement frame by a small number of basis vectors from a codebook.

The task of learning the codebook $\boldsymbol{\mathcal{B}} = \{\boldsymbol{\beta}_j\}_{j=1}^{S}$ from unlabeled data $\boldsymbol{X}$ that favors sparse solutions can be formulated as a regularized optimization problem [79, 100, 110]. Specifically, we obtain the codebook as the optimal solution to the following minimization problem:

$$\min_{\boldsymbol{\mathcal{B}},\boldsymbol{a}} \sum_{i=1}^{K} ||\boldsymbol{x}_i - \sum_{j=1}^{S} a_j^i \boldsymbol{\beta}_j||_2^2 + \alpha||\boldsymbol{a}^i||_1 \qquad (4.5)$$
$$\text{s.t.} \quad ||\boldsymbol{\beta}_j||_2 \leq 1, \forall j \in \{1, \ldots, S\}.$$

The quadratic term (first term) of the above minimization equation tries to learn a codebook that minimizes the reconstruction error, computed using the squared L_2−norm, over the entire unlabeled dataset. The L_1-regularized term (second term) favors sparse activations of the basis vectors. Moreover, the trade-off between the reconstruction quality and the sparseness of the activations can be controlled by the parameter α. For example, smaller values of α allow the quadratic term in Equation 4.5 to dominate, thereby generating basis vectors whose weighted combination can represent input signals accurately. In contrast, large values (e.g., $\alpha \approx 1$) shift the importance towards the regularization term, thereby encouraging sparse solutions, where the activations have a small L_1-norm, i.e., the input signal is represented using only a few basis vectors. Moreover, the minimization equation also needs to satisfy a set of constraints on the magnitude of the basis vectors. The constraints on the magnitude of each basis vector $\boldsymbol{\beta}_j$ are essential to avoid trivial solutions, e.g., very large $\boldsymbol{\beta}_j$ and very small activations $\boldsymbol{a}^i$ [55].

Equation 4.5 does not provide any restrictions on the possible number of basis vectors S that can be learned from the unlabeled dataset. We learn an *over-complete* codebook, which contains more basis vectors than the effective dimension of the input space, i.e., $S \gg n$. Over-completeness is a desirable property, as it is more robust to noise and other sensor degradations [100]. Sparsifying basis vectors, on the other hand, enables deviating from a purely linear relationship between the input and output and helps the codebook to capture complex and high-order patterns present in the data [100, 110].

The minimization problem given in Equation 4.5 has two optimization variables: codebook $\boldsymbol{\mathcal{B}}$ and activations $\boldsymbol{a} = \{\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^K\}$. Though the minimization problem is not convex on both $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{a}$ simultaneously, it can be easily divided into two convex sub-problems. The sub-divisions allow for iterative optimization of both $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{a}$, thereby keeping one variable constant while optimizing for the other. In other words, optimizing for $\boldsymbol{\mathcal{B}}$ keeping $\boldsymbol{a}$ constant corresponds to solving a L_2-constrained least square problem, whereas optimizing for $\boldsymbol{a}$ while keeping $\boldsymbol{\mathcal{B}}$ constant corresponds to solving a L_1-regularized least square problem [110]. In case of learning a highly over-complete codebook from a very large dataset, the solution to the minimization problem is computationally expensive. Following Lee et al. [79], we use a fast iterative algorithm to learn an over-complete codebook (see Article IV for details).

Estimation of Codebook Size

A practical approach employing sparse-coding-based recognition requires learning a large number of basis vectors. However, knowing a good codebook size a priori is often difficult. To overcome the problem, based on average reconstruction quality, we adopt a greedy binary search procedure to estimate a good codebook size. Formally, we assess the quality of a codebook by computing the average reconstruction root-mean-square error (RMSE) over the unlabeled dataset as:

$$Q_S = \frac{\sum_{i=1}^{K} \left(\frac{1}{n} ||\boldsymbol{x_i} - \sum_{j=1}^{S} a_j^i \boldsymbol{\beta}_j ||_2^2 \right)^{\frac{1}{2}}}{K}, \tag{4.6}$$

where, n is the number of samples within a frame. To find a good estimate for the codebook size, we begin by initializing the current codebook size

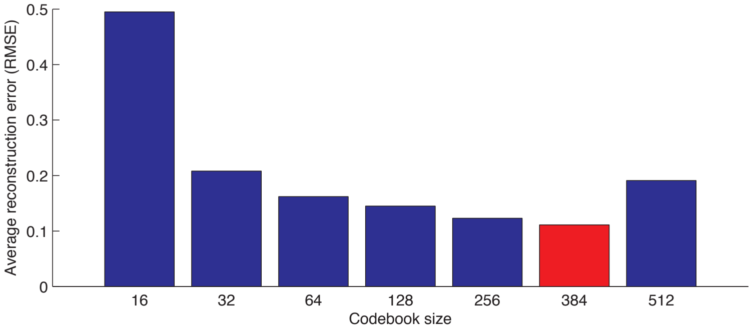


Figure 4.4: Variation of reconstruction quality with codebook size.

with a small value (e.g., $S = 16$) and continue doubling the codebook size as Q_S continues to decrease. When Q_S fails to decrease further, we backtrack by updating the next value for S to be halfway between the current codebook size and the previously found S with lowest Q_S. We stop the backtracking procedure if the current codebook size achieves the lowest Q_S. In our current implementation we set an upper limit on the size of the codebook to 512. The constraint on the codebook size helps to limit the amount of redundancy within the learned codebook and also allows us to learn the codebook in reasonable time. Figure 4.4 illustrates an example of the codebook size estimation procedure for $n = 100$ by plotting the variation of average RMSE in reconstruction (Q_S) with various codebook sizes. In the example the backtracking procedure estimates $S = 384$ (shown in red). The figure also demonstrates that an over-complete codebook, i.e., $S > n$, generally improves the accuracy in reconstruction.

4.2.2 Codebook Selection

The highly over-complete codebook learned from sequential or time-series data, using Equation 4.5, exhibits structural redundancies within the set of extracted basis vectors [43]. More specifically, some of the basis vectors are similar in shape or structure, but shifted in time. The time-shifted basis vectors are particularly problematic in case of measurement vector representations and subsequent classifier training. For example, the sliding window procedure that extracts measurement frames is agnostic of the activity types, the activity durations, and the activity boundaries. Hence, two frames of the same activity class often show a lag in time, e.g., two successive frames with a large overlap. When representing the frames using the learned codebook with redundancy, the sparse activation vectors for the

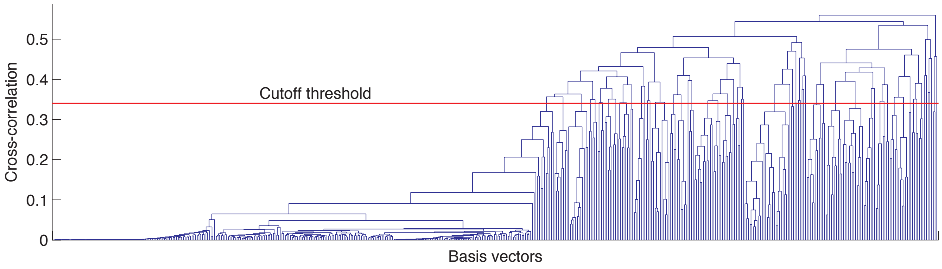


Figure 4.5: Dendrogram showing the hierarchical relationship, with respect to cross-correlation, present within a codebook of 512 basis vectors. The plot also indicates the cutoff threshold used to generate 52 clusters.

frames can vary significantly, i.e., structurally similar, but different basis vectors are activated while reconstructing the frames of the same activity. In other words, the sparsity criterion can scatter the measurement frames of the same activity class in the space spanned by the basis vectors, thus making classifier training difficult.

To alleviate the problem, Grosse et al. [43] propose a convolution-based technique and modify the minimization equation to learn a shift-invariant codebook. In their approach, the extracted basis vectors have smaller dimensions than the measurement vectors, and the basis vectors can be placed at all possible time-shifted positions within a measurement vector. However, the convolution-based approach requires significant computations and is not suitable for resource-constrained mobile devices. Instead of updating the minimization problem, we employ a codebook selection process that reduces redundancies by removing specific basis vectors that exhibit structural similarities with other basis vectors.

As the first step in our codebook selection procedure, we cluster the basis vectors learned using Equation 4.5. More specifically, we employ the complete linkage clustering algorithm [10] to form a hierarchical representation of the set of extracted basis vectors. As the measure of structural similarity between two basis vectors $\boldsymbol{\beta}$ and $\boldsymbol{\beta'}$ we use the maximum of the unnormalized *cross-correlation*. The unnormalized cross-correlation $\boldsymbol{\beta} \star \boldsymbol{\beta'}$ of two basis vectors is computed as:

$$\begin{aligned} \boldsymbol{\beta} \star \boldsymbol{\beta'}(t) &= \sum_{\tau=\max(1,t-n+1)}^{\min(n,t)} \boldsymbol{\beta}(\tau)\boldsymbol{\beta'}(n+\tau-t), \\ &\qquad \forall t \in \{1,\dots,2n-1\} \end{aligned} \tag{4.7}$$

where n is the length of the basis vectors. The similarity between the two basis vectors is then computed as:

$$sim(\boldsymbol{\beta}, \boldsymbol{\beta}') = \max(\boldsymbol{\beta} \star \boldsymbol{\beta}'). \tag{4.8}$$

To extract individual clusters from the hierarchical representation, we cut the hierarchy at a level such that it generates at least $\lceil S/10 \rceil$ clusters, where S is the cardinality of $\boldsymbol{\mathcal{B}}$. For illustration, Figure 4.5 shows the dendrogram of hierarchical relationships found within a codebook of 512 basis vectors and the cutoff threshold used to generate 52 clusters. Next, from each of the clusters we remove basis vectors that are not sufficiently informative. Accordingly, we order the basis vectors within a cluster based on their empirical entropy computed using a histogram-based approach:

$$H(\boldsymbol{\beta}) = -\sum_q p_q \cdot \log p_q, \tag{4.9}$$

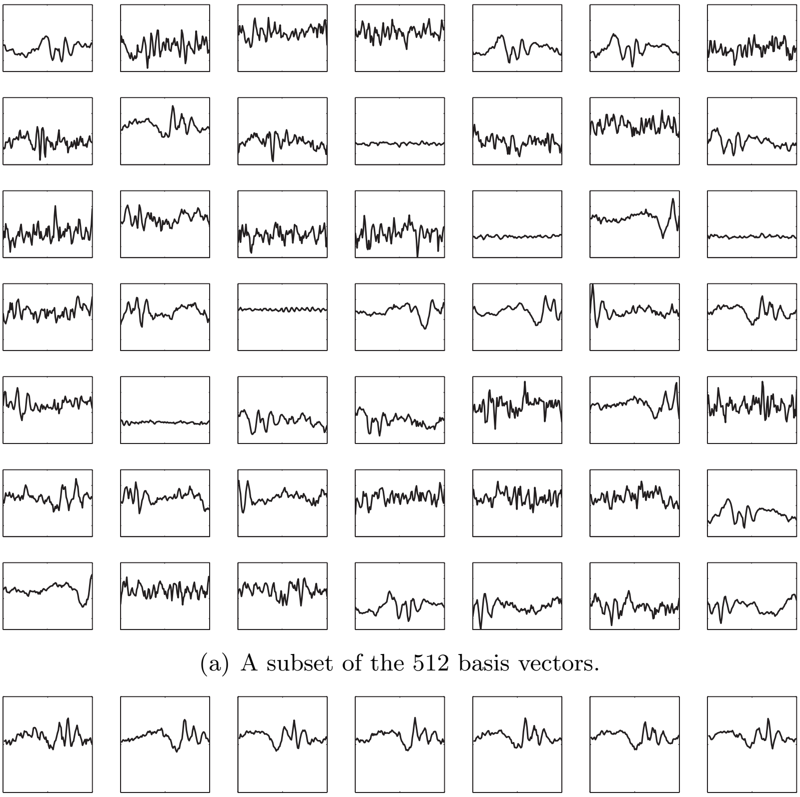
where p_q if the probability of the q^{th} histogram bin. We then discard the lowest 10-percentile of the basis vectors in a cluster. In case of a highly over-complete codebook, we experimentally select the threshold (e.g., 10-percentile) such that the overall reconstruction quality of the unlabeled dataset does not degrade significantly. The basis vectors, which are left after the selection process, constitute the optimized codebook $\boldsymbol{\mathcal{B}}^*$ in our activity recognition framework. Figure 4.6(a) shows examples of a subset of basis vectors learned from accelerometer measurement frames obtained after taking the magnitude of acceleration components. Figure 4.6(b) shows the time shifting property observed within a cluster of basis vectors.

In addition to the reduction in the structural redundancies, a more significant benefit of the codebook selection method is the improved running time to extract the activation vectors with a smaller codebook. Small running time to compute activation vectors is essential for achieving real-time recognition abilities on the mobile platform. In the following we describe the activation or feature vector extraction method in detail.

4.2.3 Feature Representation and Classifier Training

Once the codebook is optimized, it can be used to accurately represent sensor measurement vectors. The sparse activation vector needed to reconstruct a measurement vector $\boldsymbol{x}$ can be computed by solving the following optimization equation:

$$\boldsymbol{a} = \arg\min_{\boldsymbol{a}} ||\boldsymbol{x} - \sum_{j=1}^{S^*} a_j \boldsymbol{\beta}_j||_2^2 + \alpha ||\boldsymbol{a}||_1, \tag{4.10}$$



(a) A subset of the 512 basis vectors.

(b) Time shifting property.

Figure 4.6: (a) Example of basis vectors learned from accelerometer measurements (magnitude). (b) Time shifting property observed within a cluster of basis vectors.

where S^* is the cardinality of the optimized codebook $\boldsymbol{\mathcal{B}}^*$ and $\boldsymbol{\beta}_j \in \boldsymbol{\mathcal{B}}^*, \forall j$. The sparse activation vector $\boldsymbol{a}$ is the code for vector $\boldsymbol{x}$ generated using codebook $\boldsymbol{\mathcal{B}}^*$ and also the feature vector for continuous sensor measurements within our framework. Figure 4.7 depicts examples of the sparse feature vectors extracted from accelerometer measurements obtained on a user's personal mobile phone (Samsung Galaxy S III) for different transportation activities performed by the user, e.g., 'being idle', 'walking', traveling by 'bus', 'train', 'metro' and 'tram'.

To learn a mapping from the feature space to a set of target activity classes, we require a small amount of ground truth annotations along with the sensor measurement vectors. Let $\boldsymbol{X}' = \{\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_M\}$, where $\boldsymbol{x}'_i \in \mathcal{R}^n$, is the set of measurement frames for which ground truth an-

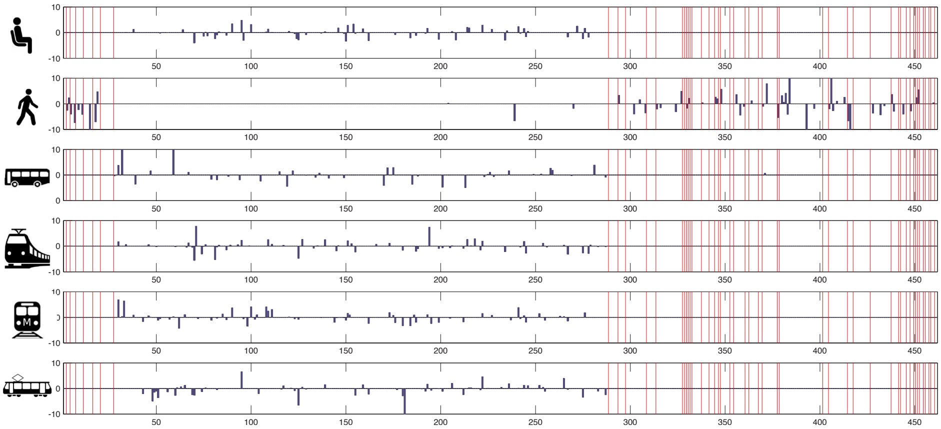


Figure 4.7: Example of sparse feature vectors derived from accelerometer measurement frames obtained during different transportation modalities. The vertical red lines separate activations belonging to different clusters of basis vectors identified during the codebook selection phase.

notations $y = \{y_1, \ldots, y_M\}$ are available. To train a classifier, e.g., a $C4.5$ decision tree [126] or a support vector machine (SVM) [29], we first construct the feature vector set $\boldsymbol{a} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_M\}$ by solving Equation 4.10 for all measurement vectors in $\boldsymbol{X}'$. The classifier is then learned using a standard supervised learning technique using the dataset $\{(\boldsymbol{a}_i, y_i)\}_{i=1}^{M}$. Once the classifier is trained, it can be transferred to the mobile phone, along with the optimized codebook, to perform online activity recognition (see Section 4.4 for details).

4.3 Benefits of the Sparse-coding Approach

To study the benefits of the sparse-coding-based approach, we apply the proposed activity recognition framework to the problem of accelerometer-based transportation-mode detection. We also demonstrate the easy generalizability of the framework in case of gesture recognition using wearable sensors, e.g., accelerometer and gyroscope, by running experiments on the publicly available *Opportunity dataset* [87, 116]. The details of the datasets and the pre-processing steps are given in Article IV.

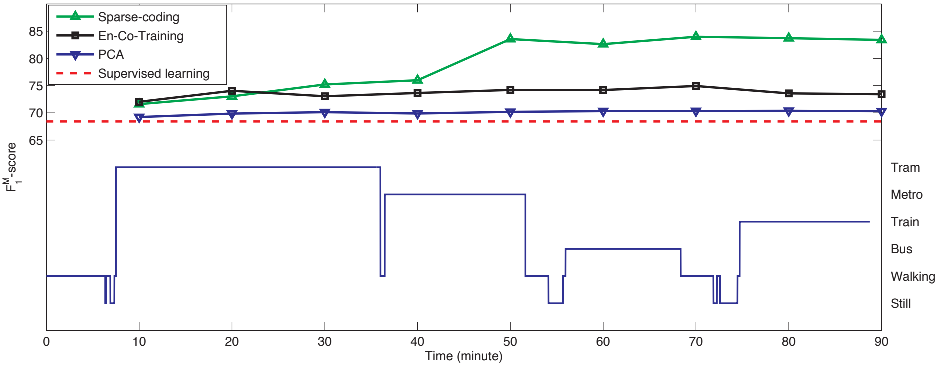


Figure 4.8: Classification accuracies for varying amounts of unlabeled data while keeping the labeled dataset and the test dataset constant.

Accelerometer-based Transportation-mode Detection

To show that even a small amount of additional unlabeled data can be effectively used to significantly improve the performance of a sparse-coding-based activity classifier we perform a set of experiments with SVM, where the size of the unlabeled dataset is increased systematically (as they are obtained), while keeping the labeled and the test dataset fixed. The experimental set-up corresponds to the envisioned application scenario, where users carry the sensing device in their everyday life without worrying about the phone, their activities, and ground truth annotations; see Article IV for details.

Figure 4.8 summarizes the results of the experiments, where we study the performances of the sparse-coding-based approach, the state-of-the-art feature-engineering-based transportation-mode detection approach by Wang et al. [139], a principal component analysis (PCA)-based feature learning approach proposed by Plötz et al. [107], which employs the ECDF normalization [47], and a semi-supervised learning algorithm En-Co-Training proposed by Guan et al. [44]. As the performance metric we use the weighted F$_1$-measure or the F$_1^M$-score [22, 118] to mitigate the effect of non-uniform class distributions present within the test dataset. The F$_1^M$-score is computed as:

$$F_1^M\text{-score} = \frac{\sum_{i=1}^{c} w_i \cdot F_1^i\text{-score}}{\sum_{i=1}^{c} w_i}, \tag{4.11}$$

where F_1^i-score represents the F_1-score of the i^{th} class (out of c different classes within the test dataset) and w_i corresponds to the number of samples

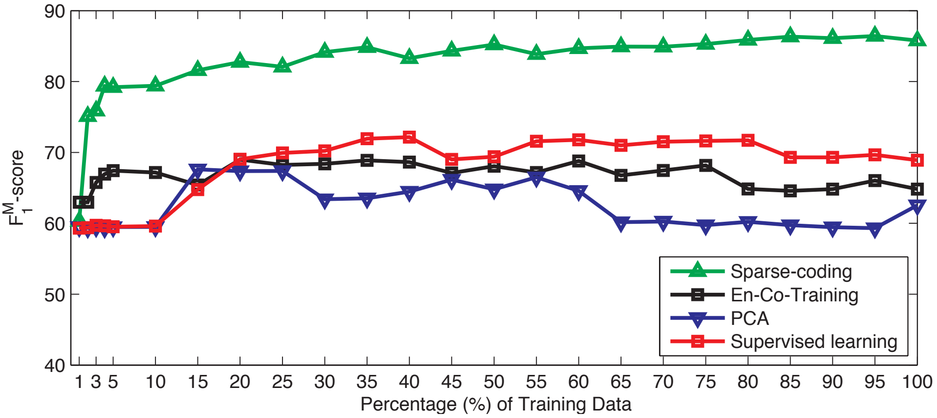


Figure 4.9: Classification accuracies for varying amounts of labeled data, while unlabeled and test datasets are kept fixed.

belonging to the i^{th} class.

As illustrated in Figure 4.8, the performance of the sparse-coding approach improves significantly, from a F_1^M-score of 71.8% to 84.6%, as more unlabeled data is available for learning the codebook. The sparse-coding based approach exhibits the best overall recognition performance among all the approaches studied. Neither the PCA-based feature learning approach nor the En-Co-Training algorithm can utilize the unlabeled dataset well and show minor variations (effect of noise) in overall classification performances with increasing size of the unlabeled data. The purely supervised learning approach, on the other hand, cannot utilize the unlabeled dataset and therefore shows constant performance level with a F_1^M-score of 68.2%. In addition to the recognition performances, Figure 4.8 also includes the ground truth labels of the unlabeled dataset to show the type of transportation activities carried out by the user. However, we do not use the annotations of the unlabeled dataset during model construction.

The superiority of the sparse-coding approach is also observed in case of a *six-fold* cross validation experiment using SVM across three users. Table 4.1 summarizes the results of the experiments for all the algorithms mentioned above. The sparse-coding approach achieves the best F_1^M-score of 79.9%, which is significantly better than all other algorithms, i.e., $p < 0.01$ in McNemar's χ^2 test with Yates' correction [93].

Even when a small amount of training data is available, the sparse-coding algorithm demonstrates a much superior recognition performance. In Figure 4.9 we show the performances of all the algorithms under varying

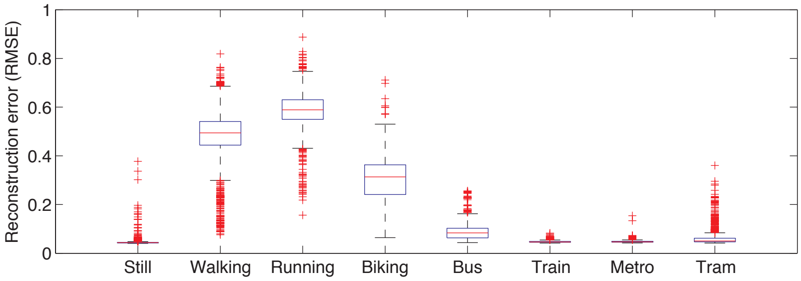


Figure 4.10: Box plot of the reconstruction errors for previously seen and extraneous (running and biking) activities.

	F_1-score						F_1^M-score
Algorithms	Still	Walk	Bus	Train	Metro	Tram	
Sparse-coding	**90.4**	**98.6**	**68.6**	**26.2**	**38.4**	**44.5**	**79.9**
En-Co-Train.	84.0	97.8	55.1	2.5	12.0	13.8	69.6
Wang et al.	81.5	96.3	51.3	2.5	10.2	17.3	67.9
PCA	83.9	91.0	39.7	0.2	3.7	6.6	65.5

Table 4.1: Classification performance of sparse-coding and other algorithms.

percentages of the labeled data used to train the SVM classifier. When only 2% of the labeled data is used, the sparse-coding algorithm achieves a F_1^M-score of 75.1%, significantly better ($p < 0.05$) than other baseline algorithms. As more training data is added, the performance of the sparse-coding algorithm improves and the algorithm achieves the highest F_1^M-score of 86.3% when 95% of the labeled data is used. Significant improvement in the overall recognition performance for small training data sizes (e.g., between 1% and 5%) and the overall stability of the performance indicate superior capability of the proposed sparse-coding-based approach.

The learned codebook can also be used to reconstruct *extraneous* activities, i.e., activities present within the labeled and/or the test dataset, but completely absent in the unlabeled dataset from which the codebook is learned. For illustration, Figure 4.10 shows the box plot for the distribution of reconstruction error of different transportation activities, along with two extraneous activities: running and biking. The box plots indicate that the derived codebook can reconstruct extraneous activities well with reconstruction errors comparable to the activities seen during the codebook learning. Further details of the evaluation results can be found in Article IV.

	F_1^M-score		
	Acc.	Gyro.	Acc. + Gyro.
Sparse-coding	**65.9**	**67.2**	**66.6**
Plötz et al. [108]	65.0	66.0	64.9
PCA	63.7	65.3	63.3

Table 4.2: Classification performance of sparse-coding and other algorithms on the Opportunity challenge dataset using accelerometer (acc.) and gyroscope (gyro.).

Gesture Recognition Using Wearable Sensors

Next we show the generalizability of the proposed sparse-coding framework beyond the accelerometer-based sensing and apply our proposed framework to identify activities of daily living as they are recorded in a sensor-rich environment as part of the Opportunity challenge dataset [87, 116]. More specifically, we focus on identifying gestures performed with the right arm from an unsegmented data stream of an inertial measuring unit (IMU) attached to the right lower arms of users. We apply sparse-coding on measurements obtained from the gyroscope and the accelerometer on the IMU. Additionally, we consider combining features extracted from both the sensors in case of gesture recognition. Table 4.2 summarizes the average results of a six-fold cross validation experiment for the sparse-coding framework, a feature-engineering based supervised learning approach proposed by Plötz et al. [108], and a PCA-based feature learning approach with ECDF normalization. The results show that the sparse-coding based approach outperforms all other state-of-the-art approaches for identifying activities of daily living. The sparse-coding based approach achieves F_1^M-scores of 65.9%, 67.2%, and 66.6% while extracting features from the accelerometer, gyroscope, and both accelerometer and gyroscope respectively. For all sensor configurations, the McNemar test indicates that the performance of the sparse-coding is statistically significant, i.e., $p \ll 0.01$ in all three sensor configurations.

4.4 Discussion

The most time-consuming process in our proposed sparse-coding framework is the task of codebook learning from the unlabeled dataset. The time required to extract the basis vectors depends mainly on (i) the size of the unlabeled dataset, (ii) the number of basis vectors to be learned, (iii) the sparsity requirement, which is controlled by the parameter α in Equation 4.5, and (iv) the dimensionality of the data, i.e., the measurement vector length n. The second most time-consuming step is the training of a supervised classifier using the labeled dataset. The requirement of a small labeled dataset also favors small training time. Both of the time-consuming steps can be done offline. Online recognition of activities can be performed on the mobile platform, as long as the mobile device has access to the codebook and the classifier, which can be easily transferred, e.g., from a desktop computer. The only time-consuming task that has to be performed on the mobile platform is the mapping of measurement vectors to the feature vectors by optimizing Equation 4.10. With moderately large codebook (e.g., 350 basis vectors) and measurement vectors, a recognition rate of around 5 Hz can be achieved using contemporary smartphones (e.g., Samsung Galaxy S III). Moreover, the codebook selection step further helps to improve the running time of the optimization equation. A plot, showing the dependency of the codebook size on the running time, is given in Article IV (Figure 10).

Chapter 5

Conclusions

In this thesis we have studied and proposed new methodologies that contribute to the research directions in ubiquitous computing and mobile sensing, in particular to those which are related to developing and deploying highly usable, real-world, and continuous context sensing applications on mobile platforms. The central research focus of this thesis has been to develop novel algorithms for localization, energy-efficient position and trajectory tracking, and activity recognition on mobile platforms with limited battery power. As localization and activity recognition are the two most popular and frequently performed context inference tasks on mobile platforms, the technical contributions made in this thesis directly benefit a large variety of mobile sensing applications that operate continuously in everyday situations.

The results presented in the first part of the thesis provided solutions to challenges regarding: (i) the optimization of GSM radio map storage, radio map maintenance and synchronization, and overall localization accuracy, (ii) the identification of the optimal duty cycling period for on-device GPS receivers, utilizing instantaneous motion contexts of a user, and (iii) the trade-off between the robustness in tracking and overall energy consumption. The second part of the thesis addressed challenges in activity recognition, another important and highly popular context recognition task within ubiquitous computing. The proposed framework (i) significantly reduces the effort needed for bootstrapping activity recognizers, and (ii) generalizes well across various application domains and sensor modalities.

The work presented in this thesis has opened up interesting new future research directions. For instance in the GSM localization work (Chapter 2), the time needed to estimate a user's location depends mainly on the number of particles and the radio map size matching the current BTS. The estimation time can be improved by employing adaptive particle filtering [37, 51]. In future we plan on using on-device motion sensors and high-level contexts, such as the current transportation mode of the user, to deploy a better particle motion model. Another interesting research direction is to study the deployment of a hierarchical grid structure to support positioning at various granularities, e.g., block or city level. Moreover, grid cell dimensions can be adapted, based on several factors including the calibration data density and the number of visible BTSs. This adaptation can be beneficial in case of trading-off radio map storage size with localization accuracy. Our proposed GSM localization algorithm can also be used by the EnTracked$_{RT}$ system, described in Chapter 3, as an alternative and less power demanding localization technique, especially when the requested error bound is large. Lastly, the parameter synchronization process is susceptible to erroneous and inflated reporting by miscreants. The lack of protection against such malicious attacks calls for new research endeavors on secure radio map synchronization.

The current implementation of the EnTracked$_{RT}$ system (Chapter 3) rely only on mobile phone sensors. We believe that the performance of the system can be improved by utilizing additional information such as the local road network, and current activity and traveling pattern of a user. Thus, our proposed activity recognition framework (Chapter 4) can further augment the EnTracked$_{RT}$ system by providing accurate information about a user's current activity or transportation mode. Knowledge of user's current activity will allow the sensor manager of the EnTracked$_{RT}$ system to switch confidently among on-device sensors. In addition to the optimised sensing, the performance of the EnTracked$_{RT}$ system can be further improved by learning and recognizing users' movement patterns on the device. For example, a predictive motion model can be learned from a user's motion history on the phone to further suppress unnecessary GPS polling, thereby developing a near *zero energy tracking system* for mobile devices on previously known routes. The integration of a suitable predictive model with EnTracked$_{RT}$ is part of our future endeavor.

The codebook selection process in our proposed activity recognition framework (Chapter 4) follows a simple cross-correlation-based clustering approach. In the future we would like to integrate a correlation-based penalty term into the optimization equation during the codebook learning. There are still many challenges in activity recognition research that call for further research. For example, there is no consensus on the optimal frame width for activity recognizers, which often varies depending on the target set of activities. The activity recognition framework outlined in this thesis is highly generalizable and can be easily adapted to work with any sensors generating a continuous data stream. Thus, our proposed methodology provides a theoretically well-founded starting point for exploring other sensing domains, including but not limited to, audio, image/video, physiological measurements, and other time-varying signals. In future we plan on taking open problems in mobile sensing and try to provide practical solutions based on solid theoretical background.

The rapid technological growth in sensing abilities and miniaturization of sensors are making our surrounding environment intelligent. While going about our everyday lives, people are leaving behind digital traces, which can be analyzed to study and understand complex societal dynamics never imagined possible before. Thus, increasing capabilities of continuous and sustained sensing, combined with the computing power of the cloud infrastructures, bring the unique opportunity to amalgamate techniques from several fields to find solutions to real-world problems. While new technological developments in pervasive sensing and inference methods have joined the new revolution called 'Big Data', new privacy-related research challenges are emerging.

The technological solutions presented in this thesis complement each other to provide a rich context model on mobile platforms under limited battery budget. The thesis also identified new and interesting research directions to overcome emerging challenges in sustainable context inference on mobile and wearable platforms.

References

[1] FCC adopts rules to implement enhanced 911 for wireless services. FCC news, CC docket no. 94-102, 12 June 1996.

[2] A. Ahtinen, M. Isomursu, Y. Huhtala, J. Kaasinen, J. Salminen, and J. Häkkilä. Tracking outdoor sports – user experience perspective. In *Proceedings of the European Conference on Ambient Intelligence*, pages 192–209, 2008.

[3] H. Alemdar, T. L. M. v. Kasteren, and C. Ersoy. Using active learning to allow activity recognition on a large scale. In *Ambient Intelligence*. Springer Berlin Heidelberg, 2011.

[4] R. A. Amar, D. R. Dooly, S. A. Goldman, and Q. Zhang. Multiple-instance learning of real-valued data. In *Proceedings of International Conference on Machine Learning (ICML)*, 2001.

[5] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS)*, 2003.

[6] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.

[7] L. Atallah and G.-Z. Yang. The use of pervasive sensing for behaviour profiling - a survey. *Pervasive and Mobile Computing*, 5(5):447–464, 2009.

[8] P. Bahl and V. Padmanabhan. Radar: an in-building RF-based user location and tracking system. In *Proceedings of 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2000.

[9] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Proceedings of the 2nd International Conference on Pervasive Computing (Pervasive)*, pages 1–17, 2004.

[10] P. Berkhin. Survey of clustering data mining techniques. In J. Kogan, C. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data*, pages 25–71. Springer, 2006.

[11] S. Bhattacharya. Place identification: A comparative study. Master's thesis, Department of Computer Science, University of Helsinki, 2009.

[12] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[13] U. Blanke, B. Schiele, M. Kreil, P. Lukowicz, B. Sick, and T. Gruber. All for one or one for all? Combining heterogeneous features for activity spotting. In *Proceeding of 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2010.

[14] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, pages 92–100, 1998.

[15] H. Blunck, M. B. Kjærgaard, and T. S. Toftegaard. Sensing and classifying impairments of GPS reception on mobile devices. In *Proceedings of the 9th International Conference on Pervasive Computing (Pervasive)*, 2011.

[16] A. Bulling, U. Blanke, and B. Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33:1–33:33, 2014.

[17] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *The VLDB Journal*, 15(3):211–228, 2006.

[18] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[19] M. J. Caruso and L. S. Withanawasam. Vehicle detection and compass applications using AMR magnetic sensors. In *Sensors Expo Proceedings*, pages 477–489, 1999.

[20] T. F. Chan, G. H. Golub, and R. J. Leveque. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242–247, 1983.

[21] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning.* MIT Press, 2010.

[22] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. del R. Millán, and D. Roggen. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013.

[23] M. Chen, T. Sohn, D. Chmelev, D. Haehnel, J. Hightower, J. Hughes, A. LaMarca, F. Potter, I. Smith, and A. Varshavsky. Practical metropolitan-scale positioning for GSM phones. In *Proceedings of the 8th international conference on Ubiquitous computing (UbiComp)*, pages 225–242, 2006.

[24] Y. Chon, E. Talipov, H. Shin, and H. Cha. Mobility prediction-based smartphone energy optimization for everyday location monitoring. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2011.

[25] Y. Chon, E. Talipov, H. Shin, and H. Cha. SmartDC: Mobility prediction-based adaptive duty cycling for everyday location monitoring. *IEEE Transactions on Mobile Computing*, 13(3):512–525, 2014.

[26] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceeding of the International Conference on Artificial Intelligence and Statistics (AISTAT)*, 2011.

[27] S. Consolvo, D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, I. Smith, and J. A. Landay. Activity sensing in the wild: a field trial of ubifit garden. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2008.

[28] I. Constandache, R. R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *IEEE INFOCOM*, pages 1–9, 2010.

[29] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2003.

[30] J. Douceur. The sybil attack. In *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin Heidelberg, 2002.

[31] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[32] C. Drane, M. Macnaughtan, and C. Scott. Positioning GSM telephones. *IEEE Communications Magazine*, 36(4):46–54, 59, 1998.

[33] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. BikeNet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):1–39, Jan 2009.

[34] T. Farrell, R. Cheng, and K. Rothermel. Energy-efficient monitoring of mobile objects with uncertainty-aware tolerances. In *Proceedings of the 11th International Database Engineering and Applications Symposium (IDEAS)*, 2007.

[35] B. Ferris, D. Hähnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *Proceeding of Robotics Science and Systems*, 2006.

[36] D. Figo, P. Diniz, D. Ferreira, and J. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14-7:645–662, 2010.

[37] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research*, 22(12), 2003.

[38] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, 2003.

[39] J. Frank, S. Mannor, and D. Precup. Activity and gait recognition with time-delay embeddings. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[40] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. Micro-Blog: sharing and querying content through mobile phones and social participation. In *Proceeding of the 6th international conference on*

Mobile systems, applications, and services (MobiSys), pages 174–186, 2008.

[41] M. G. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2002.

[42] J. geun Park, D. Curtis, S. Teller, and J. Ledlie. Implications of device diversity for organic localization. In *Proceedings of IEEE Infocom*, pages 3182–3190, 2011.

[43] R. Grosse, R. Raina, H. Kwong, and A. Y. Ng. Shift-invariance sparse coding for audio classification. In *Proceeding of International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[44] D. Guan, W. Y. Lee, Y.-K. Lee, A. Gavrilov., and S. Lee. Activity recognition based on semi-supervised learning. In *Proceeding of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2007.

[45] R. H. Güting and M. Schneider. *Moving Objects Databases.* Morgan Kaufmann, 2005.

[46] 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 the 10th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2004.

[47] N. Hammerla, R. Kirkham, P. Andras, and T. Plötz. On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In *Proceeding of International Symposium on Wearable Computers (ISWC)*, 2013.

[48] M. Hazas, J. Scott, and J. Krumm. Location-aware computing comes of age. *Computer*, 37(2):95–97, 2004.

[49] S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2013.

[50] J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. In *Proceedings of 5th International Symposium on Spatial Data Handling*, pages 134–143, 1992.

[51] J. Hightower and G. Borriello. Particle filters for location estimation in ubiquitous computing: A case study. In *Proceedings of the 6th international conference on Ubiquitous computing (UbiComp)*, pages 88–106, 2004.

[52] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[53] J. Hoey, T. Plötz, D. Jackson, A. Monk, C. Pham, and P. Olivier. Rapid specification and automated generation of prompting systems to assist people with dementia. *Pervasive and Mobile Computing*, 7(3):299–318, 2011.

[54] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services (MobiSys)*, pages 15–28, 2008.

[55] P. O. Hoyer. Non-negative sparse coding. In *Proceeding of IEEE Workshop on Neural Networks for Signal Processing*, 2002.

[56] D. H. Hu, V. W. Zheng, and Q. Yang. Cross-domain activity recognition via transfer learning. *Pervasive and Mobile Computing*, 7(3):344–358, 2011.

[57] T. Huynh, U. Blanke, and B. Schiele. Scalable recognition of daily activities with wearable sensors. In *Proceedings of Location and Context-Awareness (LoCA)*, pages 50–67, 2007.

[58] T. Huynh, M. Fritz, and B. Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing (UbiComp)*, pages 10–19, 2008.

[59] T. Huynh and B. Schiele. Analyzing features for activity recognition. In *Proceeding of Joint Conference on Smart objects and Ambient Intelligence (sOc-EUSAI)*, 2005.

[60] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[61] E. D. Kaplan and C. J. Hegarty. *Understanding GPS: Principles and Applications.* Artech House, Inc., second edition, 2006.

[62] T. Kasteren, G. Englebienne, and B. J. A. Kröse. Transferring knowledge of activity recognition across sensor networks. In *Proceeding of the 8th Internation Conference on Pervasive Computing (Pervasive)*, 2010.

[63] M. B. Kjærgaard. Location-based services on mobile phones: Minimizing power consumption. *IEEE Pervasive Computing*, 11(1):67–73, 2012.

[64] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk. Indoor positioning using GPS revisited. In *Proceedings of the 8th International Conference on Pervasive Computing (Pervasive)*, pages 38–56, 2010.

[65] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær. Entracked: energy-efficient robust position tracking for mobile devices. In *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys)*, pages 221–234, 2009.

[66] V. Könönen, J. Mäntyjärvi, H. Similä, J. Pärkkä, and M. Ermes. Automatic feature selection for context recognition in mobile devices. *Pervasive and Mobile Computing*, 6(2):181–197, 2010.

[67] M. D. Kristensen, M. B. Kjærgaard, T. Toftkjær, S. Bhattacharya, and P. Nurmi. Improving pervasive positioning through three-tier cyber foraging. In *Workshop on Pervasive Communities and Service Clouds (PerCoSC)*, 2011.

[68] J. Krumm, editor. *Ubiquitous Computing Fundamentals.* Chapman and Hall/CRC, 2010.

[69] J. Krumm, R. Caruana, and S. Counts. Learning likely locations. In *Proceeding of 21st Conference on User Modeling, Adaptation and Personalization (UMAP)*, 2013.

[70] J. Krumm and E. Horvitz. LOCADIO: Inferring motion and location from wi-fi signal strengths. In *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous)*, pages 4–14, 2004.

[71] K. Kunze, M. Barry, E. A. Heinz, P. Lukowicz, D. Majoe, and J. Gutknecht. Towards recognizing Ti Chi – an initial experiment using wearable sensors. In *Proceeding of 3rd International Forum on Applied Wearable Computing (IFAWC)*, 2006.

[72] A. Küpper. *Location-Based Services : Fundamentals and Operation.* Wiley, 2005.

[73] C. Ladha, N. Y. Hammerla, P. Olivier, and T. Plötz. Climbax: Skill assessment for climbing enthusiasts. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013.

[74] H. Laitinen, J. Lähteenmäki, and T. Nordström. Database correlation method for GSM location. In *IEEE Vehicular Technology Conference (VTC), VTS 53rd*, volume 4, pages 2504–2508, 2001.

[75] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place Lab: Device positioning using radio beacons in the wild. In *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive)*, pages 116–133, 2005.

[76] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, Sept. 2010.

[77] N. D. Lane, Y. Xu, H. Lu, S. Hu, C. T., A. T. Campbell, and F. Zhao. Enabling large-scale human activity inference on smartphones using community similarity networks (CSN). In *Proceeding of the 13th International Conference on Ubiquitous Computing (UbiComp)*, 2011.

[78] R. Lange, T. Farrell, F. Dürr, and K. Rothermel. Remote real-time trajectory simplification. In *Pervasive Computing and Communications (PerCom)*, 2009.

[79] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Neural Information Processing Systems (NIPS)*, 2007.

[80] J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. In *Proceedings of the 4th International Conference on Pervasive Computing (Pervasive)*, 2006.

[81] J. Levett. Mobile location based services: Applications, forecasts and opportunities 2009-2014. Press relesease: Juniper Research, 2010.

[82] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of*

the 8th international conference on Mobile systems, applications, and services (MobiSys), pages 285–298, 2010.

[83] R. Liu, T. Chen, and L. Huang. Research on human activity recognition based on active learning. In *International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 285–290, 2010.

[84] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *Proceedings of the 10th international conference on Ubiquitous computing (UbiComp)*, 2007.

[85] H. Lu, J. Yang, Z. Liu, N. D. Lane, C. T., and C. A. The jigsaw continuous sensing engine for mobile phone applications. In *Proceeding of ACM Conference on Embedded Networked Sensor Systems*, 2010.

[86] P. J. Ludford, D. Frankowski, K. Reily, K. Wilms, and L. Terveen. Because I carry my cell phone anyway: functional location-based reminder applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 889–898, 2006.

[87] P. Lukowicz, G. Pirkl, D. Bannach, F. Wagner, A. Calatroni, K. Förster, T. Holleczek, M. Rossi, D. Roggen, G. Tröster, J. Doppler, C. Holzmann, A. Riener, A. Ferscha, and R. Chavarriaga. Recording a complex, multi modal activity data set for context recognition. In *ARCS Workshops*, pages 161–166, 2010.

[88] S. Mallat. *A Wavelet tour of signal processing.* Academic Press, 1999.

[89] A. Malm. GPS and mobile handsets, http://www.berginsight.com/reportpdf/productsheet/bi-gps4-ps.pdf.

[90] J. Mäntyjärvi, J. Himber, and T. Seppänen. Recognizing human motion with multiple acceleration sensors. In *Proceeding of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2001.

[91] S. Matyas, C. Matyas, C. Schlieder, P. Kiefer, H. Mitarai, and M. Kamata. Designing location-based mobile games with a purpose: collecting geospatial data with cityexplorer. In *Proceeding of the International Conference on Advances in Computer Entertainment Technology*, pages 244–247, 2008.

[92] I. Maurtua, P. T. Kirisci, T. Stiefmeier, M. L. Sbodio, and H. Witt. A wearable computing prototype for supporting training activities in automotive production. In *proceeding of 4th International Forum on Applied Wearable Computing (IFAWC)*, 2007.

[93] Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157, 1947.

[94] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*, pages 337–350, 2008.

[95] D. Minnen, T. Starner, I. Essa, and C. Isbell. Discovering characteristic actions from on-body sensor data. In *Proceeding of International Symposium on Wearable Computers (ISWC)*, 2006.

[96] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 55–68, 2009.

[97] I. Muslea, S. Minton, and C. A. Knoblock. Selective sampling with redundant views. In *Proceeding of Innovative Applications of Artificial Intelligence Conference (IAAI)*, 2000.

[98] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning – Special issue on information retrieval*, 39(2–3):103–134, 2000.

[99] P. Nurmi and S. Bhattacharya. Identifying meaningful places: The non-parametric way. In *Proceedings of the 6th International Conference on Pervasive Computing (Pervasive)*, volume 5013 of *Lecture Notes in Computer Science*, pages 111–127. Springer, 2008.

[100] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1. *Vision Research*, 37:3311–3325, 1997.

[101] V. Otsason, A. Varshavsky, A. LaMarca, and E. Lara. Accurate GSM indoor localization. In *Proceedings of the 7th international conference on Ubiquitous computing (UbiComp)*, 2005.

[102] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive GPS-based positioning for smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 299–314, 2010.

[103] J. Paek, K.-H. Kim, J. P. Singh, and R. Govindan. Energy-efficient positioning for smartphones using cell-id sequence matching. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 293–306, 2011.

[104] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[105] J. Pärkkä, M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen. Activity classification using realistic data from wearable sensors. *Biomedicine*, 10(1):119–128, 2006.

[106] C. Pham and P. Olivier. Slice&dice: Recognizing food preparation activities using embedded accelerometers. In *Proceeding of International Conference on Ambient Intelligent (AmI)*, 2009.

[107] T. Plötz, N. Y. Hammerla, and P. Olivier. Feature learning for activity recognition in ubiquitous computing. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[108] T. Plötz, P. Moynihan, C. Pham, and P. Olivier. Activity Recognition and Healthier Food Preparation. In *Activity Recognition in Pervasive Intelligent Environments*. Atlantis Press, 2010.

[109] M. Rabbi, S. Ali, T. Choudhury, and E. Berke. Passive and in-situ assessment of mental and physical well-being using mobile sensors. In *Proceedings of the 13th international conference on Ubiquitous computing (UbiComp)*, 2011.

[110] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceeding of the International Conference on Machine Learning (ICML)*, 2007.

[111] H. S. Ramos, T. Zhang, J. Liu, N. B. Priyantha, and A. Kansal. Leap: a low energy assisted GPS for trajectory-based services. In *Proceedings of the 13th international conference on Ubiquitous computing (UbiComp)*, pages 335–344, 2011.

[112] O. Rashid, I. Mullins, P. Coulton, and R. Edwards. Extending cyberspace: location based games using cellular phones. *Computers in Entertainment (CIE) - Theoretical and Practical Computer Applications in Entertainment*, 4(1), 2006.

[113] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th International Conference on Innovative Applications for Artificial Intelligence (IAAI)*, 2005.

[114] S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. In *Proceedings of the 8th International Conference on Pervasive Computing (Pervasive)*, pages 138–155, 2010.

[115] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):13:1–13:27, 2010.

[116] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. del R. Millán. Collecting complex activity datasets in highly rich networked sensor environments. In *Seventh International Conference on Networked Sensing Systems (INSS)*, pages 233 –240, 2010.

[117] T. Roos, P. Myllymäki, and H. Tirri. A statistical modeling approach to location estimation. *IEEE Transactions on Mobile Computing*, 1(1):59–69, 2002.

[118] H. Sagha, S. Digumarti, J. del R Millan, R. Chavarriaga, A. Calatroni, D. Roggen, and G. Tröster. Benchmarking classification techniques using the Opportunity human activity dataset. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2011.

[119] A. Schmidt, M. Beigl, and H.-W. Gellersen. There is more to context than location. *Computers And Graphics*, 23(6):893–901, 1999.

[120] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann. GPPS: A gaussian process positioning system for cellular networks. In *NIPS*, 2003.

[121] V. Y. Skvortzov, H.-K. Lee, S. Bang, and Y. Lee. Application of electronic compass for mobile robot in an indoor environment. In *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2963–2970, 2007.

[122] T. Stiefmeier, D. Roggen, G. Tröster, G. Ogris, and P. Lukowicz. Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing*, 7(2):42–50, 2008.

[123] M. Stikic, D. Larlus, S. Ebert, and B. Schiele. Weakly supervised recognition of daily life activities with wearable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2521–2537, 2011.

[124] M. Stikic and B. Schiele. Activity recognition from sparsely labeled data using multi-instance learning. In *Location and Context Awareness (LoCA)*, 2009.

[125] M. Stikic and K. B. Schiele. Exploring semi-supervised and active learning for activity recognition. In *Proceeding of International Symposium on Wearable Computers (ISWC)*, 2008.

[126] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[127] E. M. Tapia, S. S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Proceeding of the 2nd Internation Conference on Pervasive Computing (Pervasive)*, 2004.

[128] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 85–98, 2009.

[129] A. Thiagarajan, L. S. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. Accurate, low-energy trajectory mapping for mobile devices. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2011.

[130] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* MIT Press, Cambridge, MA, September 2005.

[131] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99–141, 2001.

[132] E. Trevisani and A. Vitaletti. Cell-ID location technique, limits and benefits: an experimental study. In *6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2004.

[133] H. T. T. Truong, E. Lagerspetz, P. Nurmi, A. J. Oliner, S. Tarkoma, N. Asokan, and S. Bhattacharya. The company you keep: Mobile malware infection rates and inexpensive risk indicators. In *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, pages 39–50, 2014.

[134] K. Van Laerhoven, D. Kilian, and B. Schiele. Using rhythm awareness in long-term activity recognition. In *Proceeding of 12th IEEE International Symposium on Wearable Computers (ISWC)*, 2008.

[135] A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, and V. Otsason. GSM indoor localization. *Pervasive and Mobile Computing*, 3(6):698–720, 2007.

[136] A. Varshavsky, A. LaMarca, J. Hightower, and E. de Lara. The SkyLoc floor localization system. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 125–134, 2007.

[137] U. Varshney. Pervasive healthcare and wireless health monitoring. *Mobile Networks and Applications*, 12(2–3):113–127, 2007.

[138] T. Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, 23(176):88–93, April 1975.

[139] S. Wang, C. Chen, and J. Ma. Accelerometer based transportation mode recognition on mobile phones. In *Proceeding of Asia-Pacific Conference on Wearable Computing Systems*, 2010.

[140] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the*

7th International Conference on Mobile Systems, Applications, and Services (MobiSys), pages 179–192, 2009.

[141] J. Williamson, S. Robinson, C. Stewart, R. Murray-Smith, M. Jones, and S. Brewster. Social gravity: a virtual elastic tether for casual, privacy-preserving pedestrian rendezvous. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1485–1494, 2010.

[142] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and imprecision in modeling the position of moving objects. In *Proceedings of 14th International Conference on Data Engineering*, 1998.

[143] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases - Special issue on mobile data management and applications*, 7(3):257–387, 1999.

[144] Z.-L. Wu, C.-H. Li, J.-Y. Ng, and K. Leung. Location estimation via support vector regression. *IEEE Transactions on Mobile Computing*, 6(3):311–321, 2007.

[145] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114, 2010.

[146] Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma. Geolife2.0: A location-based social networking service. In *Proceeding of 10th International Conference on Mobile Data Management: Systems, Services and Middleware (MDM)*, pages 357–358, 2009.

[147] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[148] Z. Zhuang, K.-H. Kim, and J. P. Singh. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys)*, 2010.