Energy-efficient Continuous Context Sensing on Mobile Phones

THÈSE N^{O} 6761 (2015)

PRÉSENTÉE LE 9 OCTOBRE 2015
À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE SYSTÈMES D'INFORMATION RÉPARTIS
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Julien EBERLE

acceptée sur proposition du jury:

Dr O. Lévêque, président du jury
Prof. K. Aberer, directeur de thèse
Prof. O. Abou Khaled, rapporteur
Dr G. P. Perrucci, rapporteur
Prof. D. Gatica-Perez, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

Acknowledgements

First and foremost, I would like to thank my thesis director Professor Karl Aberer for his guidance all along my research. I still remember one of my first meetings with him, when he told me I will need to be somehow autonomous. I took it as a challenge.

But being autonomous doesn't mean always working alone, and in this regard, I would like to thank also Zhixian Yan, Sofiane Sarni, Matteo Vasirani and Jean-Paul Calbimonte for the collaboration and very valuable advices that helped me find my way in the jungle of publications and conferences, and understanding the nuts and bolts of the academic world.

I will also never forget how it all started at Nokia Research Center Lausanne and the fantastic atmosphere and the team spirit I experienced there. Thanks a lot to the former NRC team!

After my first year and the shutdown of NRC Lausanne, moving back to EPFL was a difficult choice, the non-academic world presenting tempting alternatives. But the warm welcome I received from the LSIR members convinced me more deeply that I made the good decision. Therefore, I want to thank all of the current and former LSIR members, including the ICSIL staff, and more specially Tri, Alexandra, Michele, Berker, Chantal and Jean-Eudes.

Working all the time on the thesis is not the most efficient way to get through it, one needs to take a step back sometimes and change his perspective. For that, I would like to thank my family and friends who supported me in my moments of doubts and re-motivated me. A big thank you for my parents, Joëlle and her parents too! Thank you also to the PolyLAN team for giving me the occasions to look at my work more like a GAME!

I would also like to thank the whole OpenSense and OpenSense2 teams for the enriching collaborations and more particularly Adrian for all the time spent to explain to me everything about air pollution and how to sense it.

A big thank also to Steven for his time proof-reading this thesis and all the others that have helped me during those four years.

This thesis was funded by Nokia Research Center, Nano-Tera OpenSense and OpenSense2 projects, and the OSPER project.

Lausanne, 11 September 2015 J. E.

Abstract

With the ever increasing adoption of smartphones worldwide, researchers have found the perfect sensor platform to perform context-based research and to prepare for context-based services to be also deployed for the end-users. However, continuous context sensing imposes a considerable challenge in balancing the energy consumption of the sensors, the accuracy of the recognized context and its latency. After outlining the common characteristics of continuous sensing systems, we present a detailed overview of the state of the art, from sensors sub-systems to context inference algorithms. Then, we present the three main contribution of this thesis.

The first approach we present is based on the use of local communications to exchange sensing information with neighboring devices. As proximity, location and environmental information can be obtained from nearby smartphones, we design a protocol for synchronizing the exchanges and fairly distribute the sensing tasks. We show both theoretically and experimentally the reduction in energy needed when the devices can collaborate.

The second approach focuses on the way to schedule mobile sensors, optimizing for both the accuracy and energy needs. We formulate the optimal sensing problem as a decision problem and propose a two-tier framework for approximating its solution. The first tier is responsible for segmenting the sensor measurement time series, by fitting various models. The second tier takes care of estimating the optimal sampling, selecting the measurements that contributes the most to the model accuracy. We provide near-optimal heuristics for both tiers and evaluate their performances using environmental sensor data.

In the third approach we propose an online algorithm that identifies repeated patterns in time series and produces a compressed symbolic stream. The first symbolic transformation is based on clustering with the raw sensor data. Whereas the next iterations encode repetitive sequences of symbols into new symbols. We define also a metric to evaluate the symbolization methods with regard to their capacity at preserving the systems' states. We also show that the output of symbols can be used directly for various data mining tasks, such as classification or forecasting, without impacting much the accuracy, but greatly reducing the complexity and running time.

In addition, we also present an example of application, assessing the user's exposure to air pollutants, which demonstrates the many opportunities to enhance contextual information when fusing sensor data from different sources. On one side we gather fine grained air quality information from mobile sensor deployments and aggregate them with an interpolation model.

And, on the other side, we continuously capture the user's context, including location, activity and surrounding air quality. We also present the various models used for fusing all these information in order to produce the exposure estimation.

Key words: mobile sensing, time series, collaborative sensing, machine learning, sensor scheduling, energy-efficiency, symbolic representation, stream processing

Résumé

Avec l'adoption mondiale sans cesse croissante des smartphones, les chercheurs ont trouvé la plateforme idéale pour mener leurs recherches sur le contexte et préparer le terrain pour que les services basés sur le contexte soient déployés jusqu'aux utilisateurs finaux. Néanmoins, la perception en continue du contexte pose un défi considérable pour trouver un compromis entre la consommation énergétique des capteurs, l'exactitude de la reconnaissance du contexte et sa latence.

Après la présentation des caractéristiques communes des systèmes de mesures en continue, nous présentons un aperçu détaillé de l'état de l'art, partant des sous-systèmes de capteurs, jusqu'aux algorithmes d'inférence de contexte. Ensuite, nous décrivons les trois contributions principales de cette thèse.

La première approche que nous présentons est basée sur l'utilisation de communications locales pour échanger les informations collectées avec les appareils voisins. Comme les informations environnementales, de proximité et de position peuvent être obtenues d'autres smartphones proches, nous avons conçu un protocole pour synchroniser les échanges et répartir de manière équitable les tâches. Nous démontrons théoriquement et expérimentalement la réduction d'énergie nécessaire dans le cas où les appareils peuvent collaborer.

La deuxième approche se concentre sur la façon de planifier les capteurs mobiles, optimisant l'exactitude et les besoins énergiques. Nous formulons le problème de planification optimale, en tant que problème de décision et nous proposons un système à deux étapes pour approximer sa solution. La première étape segmente la série temporelle des mesures du capteur en ajustant divers modèles. La deuxième étape défini l'échantillonnage optimal, sélectionnant les mesures qui contribuent le plus à l'exactitude du modèle. Nous fournissons des heuristiques quasi-optimales pour chacune des étapes et évaluons leurs performances en utilisant des données de capteurs environnementaux.

Pour la troisième approche nous proposons un algorithme d'apprentissage incrémental qui identifie les motifs répétitifs dans les séries temporelles et produit un flot de symboles compressé. La première transformation symbolique partitionne les données initiales. Puis les itérations suivantes encodent les séquences répétitives de symboles en d'autres symboles. Nous présentons une métrique pour évaluer les transformations par rapport à leur capacité à préserve les informations d'état du système. Nous montrons également que les symboles en sortie peuvent être directement utilisés pour diverse tâches d'exploration de données, tels que la classification ou prévision, sans impacter de manière notable leur exactitude, mais en

réduisant fortement leur complexité et temps de calcul.
De plus, nous présentons un exemple d'application, estimant l'exposition aux polluants atmosphérique, qui démontre les nombreuses opportunités pour enrichir les informations contextuelles lors de la fusion de différentes sources de données. D'un côté, nous collectons des informations détaillées sur la qualité de l'air à partir d'un déploiement de capteur mobiles et nous les agrégeons au moyen d'un modèle d'interpolation. Et, d'un autre côté, nous capturons le contexte de l'utilisateur en continue, incluant sa position, son activité et la qualité de l'air environnant. Nous présentons aussi les divers modèles utilisés pour fusionner ces données afin de produire l'estimation d'exposition.

Mots clefs : collecte mobile, séries temporelles, collecte collaborative, apprentissage automatique, planification de mesures, efficacité énergétique, représentation symbolique, traitement de flux

Contents

Acknowledgements **i**

Abstract (English/Français) **iii**

List of figures **xi**

List of tables **xv**

Introduction **1**

1 Background and Motivation **3**
- 1.1 Background 3
- 1.2 Applications 4
- 1.3 Challenges 5
- 1.4 Generic architecture 6
- 1.5 Contributions of this thesis 8

2 State of the art **11**
- 2.1 Sensing platforms 11
  - 2.1.1 Dedicated Hardware 11
  - 2.1.2 Smartphones 12
- 2.2 Sensor abstraction 13
  - 2.2.1 Sensor scheduling 16
- 2.3 Context inference 18
  - 2.3.1 Activity recognition 19
  - 2.3.2 Location 21
  - 2.3.3 Environment 23
  - 2.3.4 Proximity 25
  - 2.3.5 Health 28
- 2.4 Sensor fusion 31

3 Collaborative sensing **33**
- 3.1 Introduction 33
- 3.2 Related Work 35
  - 3.2.1 Centralized coordination 35

3.2.2 Opportunistic collaboration 35
3.2.3 Ad-hoc coordination 36
3.2.4 Fairness 36
3.3 Problem Formulation 37
3.3.1 General definitions 37
3.3.2 Specific instance 38
3.4 Opportunistic Strategy 40
3.4.1 Base-strategy 40
3.4.2 Synchronization 40
3.4.3 Time-window 41
3.4.4 Parameters selection 43
3.5 Theoretical Optimum 43
3.5.1 Definitions 45
3.5.2 Competitive analysis 47
3.6 Evaluation 48
3.6.1 Dataset descriptions 48
3.6.2 Results 51
3.7 Conclusion 55

4 Optimal Sampling **57**
4.1 Introduction 57
4.2 Related Work 58
4.2.1 Sensor Placement in WSN 58
4.2.2 Sensing from Mobile Sensors 58
4.2.3 Time series segmentation 59
4.3 Two-Tier Optimal Mobile Sensing 59
4.3.1 Problem Statement 60
4.3.2 Near-Optimal Segmentation 60
4.3.3 Near-Optimal Sampling 63
4.4 Experimental Evaluations 66
4.4.1 Segmentation Results 67
4.4.2 Sampling Results 68
4.4.3 Near-Optimal Combination 71
4.5 Conclusion 71

5 State identification **73**
5.1 Introduction 73
5.1.1 Solution overview 75
5.2 Related Work 76
5.2.1 Time series segmentation 76
5.2.2 Online frequent pattern mining 77
5.2.3 Time series subsequences clustering 77
5.2.4 Time series features extraction 78

5.3 State-Preserving Symbolic Representation 79
5.3.1 State Preservation Index 79
5.3.2 The Spclust Algorithm 81
5.3.3 Run Length Encoding 82
5.4 The StateFinder Algorithm 82
5.4.1 Overview 82
5.4.2 Segmentation 83
5.4.3 Clustering 86
5.4.4 Improving segmentation accuracy 87
5.4.5 Reverting States 88
5.5 Applications 89
5.5.1 State Anomaly Detection 89
5.5.2 State Forecasting 90
5.5.3 Data Compression 90
5.6 Experiments 91
5.6.1 State Preservation Index 91
5.6.2 State Recognition 93
5.6.3 State Forecasting 94
5.6.4 State Anomaly Detection 96
5.6.5 Running Time 96
5.7 Conclusion 97

6 Application example: Air pollution exposure estimation for personalized health 99
6.1 Introduction 99
6.2 Related Work 100
6.2.1 Assessing pollution exposure 100
6.2.2 Personal monitoring devices 101
6.3 System architecture 102
6.4 Personal Sensing Device 103
6.4.1 Hardware 103
6.4.2 Software 104
6.5 Aggregation Server 105
6.6 Sensor Fusion 107
6.6.1 Activity and breathing 107
6.6.2 Location 108
6.6.3 Pollution concentration 108
6.6.4 Pollution exposure 109
6.7 Experiments 109
6.7.1 GSN server 109
6.7.2 tinyGSN 110
6.8 Discussion 111

7 Conclusion **113**
7.1 Future directions 114

Bibliography **149**

Curriculum Vitae **151**

List of Figures

1.1 The generic architecture for continuous context sensing 7

2.1 Evolution of m-Health applications (source [146, p. 229]) 28
2.2 The context processing stages and their corresponding fusion methods. 31

3.1 The error is growing linearly between the measurements and the total error is the total area of the triangles. 39
3.2 Optimal Number of samples for different values of δ for an experiment of one day, without collaboration. 41
3.3 Timelines showing the samples taken and exchanged by two users. 42
3.4 Worst scenario when two nodes try to collaborate. 47
3.5 Percentage of time with another of the 166 user in the BT vicinity 49
3.6 Distribution of the intervals between two encounters. 49
3.7 Distribution of the duration of the encounters. Removing the intervals smaller than 4 minutes reduces by one order of magnitude the total number of encounters. 50
3.8 Timeline from the Paléo dataset showing when GPS (Green and Yellow) and Bluetooth encounters (Red, Blue and Yellow) are available. Red means that a GPS location is included in the BT beacon, whereas Blue is when there is no location. (for B&W printing, from darker to lighter: red, blue, green, yellow) . . 51
3.9 Histogram of the time between two consecutive Bluetooth encounters and encounters duration over all the users. 52
3.10 Distribution of energy and total cost saving for all users using the optimal solution. 52
3.11 Error reduction induced by collaboration and measurement exchanges, showing the offline optimal solution and our online algorithm. 53
3.12 Average number of duplicated measurements per node over one year for different budgets, the optimal strategy always being at 0. 54
3.13 Heatmap of the total cost with $\delta = 0.2$ according to the R and W parameters . . 54

4.1 OptiMoS's two-tier optimal sensing framework 59
4.2 Results using SVM-inferred segments from the training set applied on the testing set 68
4.3 Regression in segments ($Heuristic^{+}$, $K = 5$) 68
4.4 RSS Ratio by different sampling methods (SVM) 69
4.5 RSS Ratio by different sampling methods (Linear) 70

4.6 RSS Ratio convergence using Mutual-Information sampling 70
4.7 RSS Ratio for different combinations of segmentation and sampling (SVM) . . 71

5.1 An overview of our solution. We start with the initial symbolization process (Spclust, see Section 5.3). Next, we repeatedly compress the representation using run-length encoding (RLE) and recognize states using StateFinder (see Section 5.4). The output is then used in various applications (see Section 5.5 and 5.6). 75
5.2 Illustration: several consecutive occurrences of a pattern indicate that the system is in a certain state. 83
5.3 We apply the Spclust algorithm to the raw data (sensor measurements) to produce the symbolic time series. After applying RLE to the symbolic time series (not visible in this figure), we apply StateFinder to find the states. The StateFinder algorithm can be applied several times to build a multi-level state representation. The raw data is displayed using the horizon graph [106]. 83
5.4 Illustration on how segments are formed, using the data from the bird call dataset in [235]. The dashed line shows the (three) symbols at level 0, plotted as values 0.0, 0.1, and 0.2, the thin solid line is the prediction error over time, and the thick solid line is the threshold parameter $p = 0.9$. The gray areas show the segments. 86
5.5 Complete microwave dataset, showing one month of usage (top) and one hour of microwave oven usage (bottom), showing different power levels. 91
5.6 Evaluation of symbolic representation of the microwave dataset. In (a) the highest eval value is achieved using grid size=20 and threshold=100, 200, or 300. Next, for (b) and (c), we set Spclust grid size to 20 and threshold to 200. The alphabet size of uniform, median, and distinct median is set to 3. For KMeans, we set k=3. 92
5.7 Prototypes of the states, generated from the transition matrices of the microwave time series in Figure 5.5a. 94
5.8 Gyroscope value on x-axis, recorded on a smartphone while the user was performing different activities. Below is the symbolized version using 3 symbols and the color indicates the state detected by the StateFinder. 95
5.9 Comparison of the 1-hour forecasting accuracy using both data representations, symbols (level 0) and states. 95
5.10 Timing for the pattern-based forecasting algorithm, on top of different data, and the processing steps for generating those data. 96
5.11 Anomaly detection on states and symbols generated from the microwave dataset. The top plot shows the raw data including the anomaly around hour 30. The bottom plot shows the anomaly probability computed from the the StateFinder output symbols. 96
5.12 Processing time for running StateFinder on different datasets and dataset sizes. 97

6.1 System Block Diagram 102
6.2 Air Quality extension for smartphones 103

6.3 Internal and external communication, based on ØMQ 106
6.4 comparison of elements produced and consumed per second for the 3 scenarios: using remote wrapper, using zeromq wrapper with xml serialization or Kryo serialization 110
6.5 The dots on the map represent the GPS points collected, colored by day, whereas the timeline at the bottom shows the various states of the system on the first day. (blue: lost, orange: GPS moving, light gray: stationary by accelerometer, dark gray: stationary by Wi-Fi) 111

List of Tables

2.1 Non-exhaustive list of available sensors on mobile platforms and how they can contribute to the usual context dimensions. D: directly, M: mapping through a model, C: combined to other sensors. The grayed cells indicate the most commonly used sensors for those contexts. *see the sections below for a longer list of related works using these sensors. 19

3.1 Summary of the notation 38
3.2 Performance comparison between the baseline (regularly sampling) with and without communication, and our collaborative strategy. 54

6.1 Confusion matrix for the Random Forest activity classifier 110

Introduction

With 1.2 billion units sold in 2014, smartphone sales represented more than 60% of the overall mobile phone sales, according to Gartner [92]. This trend is expected to continue growing, especially in the developing countries, where smartphones are increasingly available at affordable prices.

In the near future, the ever increasing capabilities of smartphones to record and sense will raise the promise of more personalized and contextually relevant services. Given the numerous built-in sensors in today's smartphones (accelerometer, GPS, microphone), and their access to their owner's calendar, to-do list, SMS/call history, etc., it is not hard to imagine that in the near future they will be capable of completely analyzing our daily schedule, and giving specifically tailored and highly personalized recommendations. These could include what clothes to wear (depending on predicted activities and weather forecast), which route to take to avoid traffic jams, and which items to buy at the supermarket (depending on food preferences or whether one plans to have guests). In addition, the role of sensor networks is expected to become pervasive. Smart home and smart city concepts require sensor deployment in almost every corner of our living space, to make sure that homes and cities are "smart enough" to assist and support the needs of their inhabitants [17].

All the information collected by these mobile sensors, following us everywhere, enables the devices to understand our context and, day after day, allows them to know us better. As Jain and Jalali wrote in their vision article [120], humans are driven by curiosity, and understanding themselves has always been of great interest. For this reason, humans started writing diaries and chronicles, recording events of their life, also taking pictures and videos. More recently, the notion of *quantified self* related to the more scientific approach of recording all digital traces left by our actions. Finally, with the pervasiveness of sensors, Jain and Jalali [120] proposed a new concept, called *objective self*, where the sensors actively and objectively sense all the aspects of our context, without the bias of the user driven sensing.

However, to make this future a reality, some challenges have to be addressed, related to the limitation on the resources on mobile devices, like for example the battery or the local storage. Going in this direction, this thesis presents contributions that leverage mathematical models, smart use of sensors and communication and the combination of sensors.

First we propose to share the output of energy hungry sensors between nearby devices, and we develop a protocol to schedule the measurements and exchanges, distributing the load evenly among the devices. Then, we design a method in two steps to approximate the optimal sampling in the case of a mobile sensor, relying on various models to interpolate the measurements. Finally, we propose an efficient representation of sensor data to reduce the costs of storage and data mining, while identifying and preserving the states' information present in the data stream.

The chapters 3, 4 and 5 of this thesis are based on the following publications:

- J. Eberle, Z. Yan and K. Aberer. Energy-Efficient Opportunistic Collaborative Sensing. *IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, Hangzhou, China, October 14-16, 2013.
- Z. Yan, J. Eberle and K. Aberer. OptiMoS: Optimal Sensing for Mobile Sensors. *13th International Conference on Mobile Data Management (MDM), Bengaluru*, India, July 23-26, 2012.
- J. Eberle, T. K. Wijaya and K. Aberer. Online Unsupervised State Recognition in Sensor Data. *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, St. Louis, Missouri, USA, March 23-27, 2015.

, and the chapter 6 is partially based on these ones:

- Y. Kim, J. Eberle, R. Hanninen, E. C. Un and K. Aberer. Mobile observatory: an exploratory study of mobile air quality monitoring application. *ACM conference on Pervasive and ubiquitous computing - UbiComp '13*, Zurich, Switzerland, 08-12 09 2013.
- E. C. Un, J. Eberle, Y. Kim and K. Aberer. A model-based back-end for air quality data management. *ACM conference on Pervasive and ubiquitous computing - UbiComp '13*, Zurich, Switzerland, 08-12 09 2013.
- B. Predic, Z. Yan, J. Eberle, D. Stojanovic and K. Aberer. ExposureSense: Integrating Daily Activities with Air Quality using Mobile Participatory Sensing. *IEEE International Conference on Pervasive Computing and Communication*, San Diego, California, USA, March 18-22, 2013.

1 Background and Motivation

1.1 Background

In this thesis we define the context as a multidimensional space representing the user's situation. This is very similar to the well known definition from A. Dey:

> *Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.* [70]

However, in our case we will focus on the personal context, and all the sensing modalities that are related. We can already divide the contextual dimensions into two types, namely environmental (exteroception), and personal (proprioception). The exteroception relates to the perception of external signals such as the proximity to other people or objects, the environmental conditions, including the weather, sound or light, and the positioning in an external coordinate system. Whereas the proprioception relates to the perception of internal signals, in our case the human body, such as the user's activity (at different levels), her vital signs, emotions, and other health related parameters. Then, the combinations of these dimensions can be mapped to a higher level description of the user's situation, such as "working in front of a computer in my office", "walking home", or "shopping with my parents".

Mobile computing has greatly evolved in the last few years, since the survey on context-aware mobile computing from Chen and Kotz [46] and the update by Hong et al. [112]. Some of their research topics became actual products on the market and some new hardware and enablers opened new research directions. But most importantly, smartphones have taken a prominent place in our society, giving the researchers the perfect tool to build sensing platforms.

First, the smartphones encapsulate a wide range of sensors, long-range as well as short range communication, and mechanisms to rapidly develop and deploy applications at large scale. In this way, they are perfectly fitting the need as a context sensing device. The sensing modalities present on today's smartphones allow them to measure some of the contextual dimensions directly through the embedded sensors, such as the GPS or the light sensor. And for the other dimensions, they have to be translated from indirect measurements, as for example the accelerometer, which can provide information about the activity of the user, or the front camera which can be used for tracking our emotions [317].

Secondly, the smartphones are following their owners very closely, thus matching their measurements to the actual situation of the user in a non-intrusive way. In 2006, Patel et al. [214] conducted an experiment showing that users were having their phones at arms reach only 58% of the time and in the same room 78% of the time. But more recently, Dey et al. [71] conducted a similar experiment with smartphones and even if the close proximity didn't change much, the smartphones were located in the same room almost 90% of the time. The authors also provided a model for predicting the proximity of the phone, allowing context sensing application to adapt themselves to the situation.

1.2 Applications

This section introduce various use cases and applications that are or could be benefiting from a continuous context-aware mobile system.

Sensing and collecting information about the context of users can be, by itself, a research goal. But more interestingly the main driver for such research is to be found in the potential applications that can be enabled on top of a context-aware mobile platform. In their survey, Lockhart et al. [174] give an overview of the various applications that can be built on top of an activity recognition platform. For example, they mention the fitness tracking application recording daily performances and calories burnt, health monitoring applications and fall detection, home automation, but also third party applications such as targeted advertisement or corporate management and accounting for employee time.

The first thing one may want to do with contextual data is to store them and develop various visualizations to quickly grasp the life patterns and events, like in the Lifelogging application [241] or more recently in Memo-it [3]. Once the current context and some historical life patterns are identified, there is also a possibility to develop context-based recommender systems [170]. For example, knowing the user's wardrobe, her mood, clothes preferences, the weather condition of the day, the user's agenda and the groups of people she will likely meet during the day, one could develop an application for recommending what to wear today.

Health related applications have also drawn a lot of attention because of their potential impact on human beings' primary needs. They can be split in two categories, the ones for well-being, like sports tracking, and personal training, usually based on the gamification concept, and the ones for medical usage, like monitoring and alerting, usually targeting specific groups of people, like elderly people, the ones with chronic diseases and high-risk groups. Some example of such applications have already been implemented as prototypes, like Mobilize! [39], a mobile application for helping depressive people follow their therapy or LaCasa [109], an application for alerting when a patient with dementia is having an abnormal behavior.

In a more general way, if we do not restrict ourselves to a single application field, like clothing, health, places, movies, a recommender system could become a personal assistant [100], able to react to the user's emotions and provide useful information when queried. When coupled with a natural language interface, such as Siri[1] or Cortana[2], it can become a very personalized assistant. According to their website[3] and an article in the news[4], Viv labs are currently developing "an intelligent, conversational interface". They would be able to mine all available information about the user (calendar, contacts, ...) and from the Internet to answer pretty complex queries such as "finding the store on the way to your brother's that has the cheapest wine that goes well with lasagna". Going further, one could imagine the endless possibilities if the reasoning back-end could be empowered by engines such as IBM Watson[5].

Finally and most importantly, the contexts information should be open and made accessible to third party applications, with the user's permission of course. This would let anybody express their imagination to define new possible usage and services around contextual information. It can be implemented in the operating system API or as an extension to the HTML and JavaScript standard library, similarly to the W3C Sensor API[6], thus making the web context aware.

1.3 Challenges

As for every embedded and mobile device, the available computational and energy resources are critical. Especially the battery lifetime is an important factor for people's daily use of their smartphones. The limitations of the mobile platform must be taken into account at each level of design. First, the power consumption of the sensors themselves are spread over a wide range. Previous studies have shown for example that GPS energy consumption is one order of magnitude higher than normal mobile phone energy consumption in idle mode [79]. Therefore their sampling rate and duty cycles have to be carefully defined to enable them to sleep for some time and save energy, while providing enough meaningful measurements when they are awake.

This holds also for the algorithms that will process the data. Even if modern smartphones now possess similar computing capacities to desktop computers, allowing them to run very complex algorithms in real-time on the sensor data flow, it does not come for free. Indeed, in most recent smartphones, the power consumption of the CPU can be adapted to its load,

---

[1] https://www.apple.com/ios/siri/

[2] http://www.windowsphone.com/en-us/how-to/wp8/cortana/meet-cortana

[3] http://viv.ai/

[4] http://www.wired.com/2014/08/viv/

[5] http://www.ibm.com/smarterplanet/us/en/ibmwatson/

[6] http://www.w3.org/2009/dap/

by adjusting its working frequency. Therefore, reducing the computational needs or making the algorithms more efficient will also improve the battery duration. The marginal cost of improving accuracy of the models and algorithms must also take into account the increase in computational complexity. Moreover a good usage of sleep time and synchronization between repetitive processes can also take advantage of these adaptive CPUs.

Therefore, when dealing with energy-hungry sensors, we need to come up with a good sensing strategy that considers the "trade-off" of three dimensions, i.e., *latency*, *accuracy* and *energy consumption*.

First, as we aim at having a system that is continuously aware of the contexts to support real-time context-based applications, we need to minimize the detection delay of a state change. This does not necessarily imply continuous sensing on all sensors, but rather knowing the right time to turn them on. From the sensor perspective this is also an issue as some of them, like the GPS, are not available everywhere, raising the need for having other sensor alternatives. On the algorithmic side, the latency constraint can have different meanings: it can be in terms of performance, where fast and efficient algorithms will be able to detect instantly a context change or on the algorithm needs, like for example requesting data that is not yet available because of the network/sensors being slow or the sliding window needing future data.

The second constraint, accuracy, can be adapted by changing the sampling rate of the sensors, increasing the uncertainty between the measurements, or by sensor selection, with a more or less accurate sensor or group of sensors. At the algorithm level, different models can be used, showing various performances in terms of accuracy and complexity.

Finally, energy consumption, as explained before, can be impacted in various way, either by energy-hungry sensors or by running intensive computation and communication tasks. Minimizing the power consumption, while optimizing the accuracy of the measurements for reconstructing the measured field has recently been a subject of intense research [58, 83, 209] and Chapter 2 gives a broad overview of the various methods used in the literature to improve in any of these dimensions.

Mobile sensing also raises other challenges, more specific to the domain in which it is used. These includes, for example, connectivity, privacy, trust, reputation, user attention, incentives, sensor invasiveness and acceptance, and sensor calibration. However, they have been subjects of orthogonal researches and will not be further considered in this thesis.

1.4 Generic architecture

For this thesis we will consider a generic architecture that corresponds to the typical data-flow in mobile phone based context sensing applications [115], but focusing more on the lower layers and adding our contributions. The Figure 1.1 shows the three different layers of this

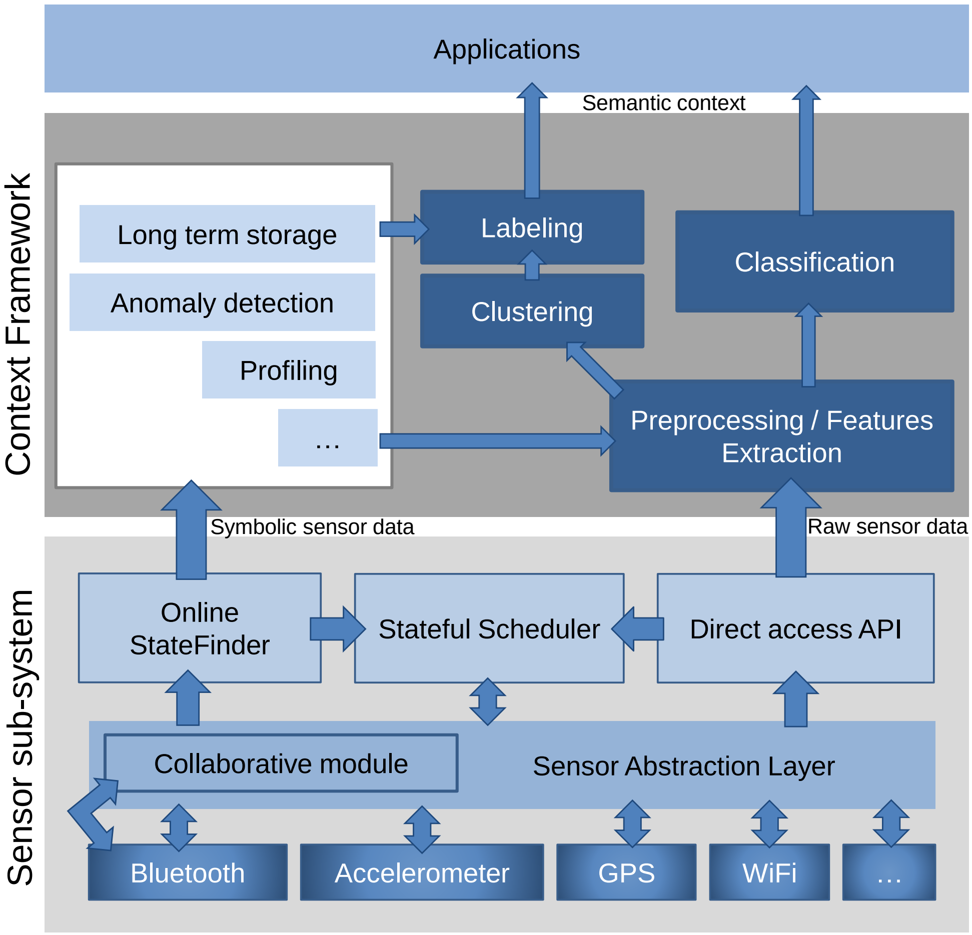


Figure 1.1: The generic architecture for continuous context sensing

architecture.

First, at the bottom, the sensor subsystem takes care of efficiently gathering data from the physical sensors. An abstraction layer simplifies the access to internal as well as external sensors with a common API. It includes a collaborative module that takes care of sharing measurements with neighboring devices. Its protocol, using Bluetooth communication, is described in detail in Chapter 3. A scheduler takes care of optimizing the sensor sampling in such a way that the requests can be satisfied while reducing the energy consumption of the sensors (see Section 2.2.1). An offline version of this algorithm is presented in Chapter 4. To be able to access the sensors from the operating system, the *direct access API* serves as a proxy, forwarding the requests to the scheduler and reading the sensor values from the abstraction layer. An alternative access to the sensor data is provided by the StateFinder module, presented in Chapter 5. It outputs a symbolic representation of the data, that is more efficient to process, while keeping most of the information from the sensors. This module can also generate queries for the scheduler, if it needs more recent data or if it runs in continuous data flow.

The whole sensor subsystem is supposed to be running continuously in the background, and thus has very hard energy efficiency constraints. Therefore, in most of its implementations in present day smartphones, it runs on a separate hardware processor (see Section 2.1).

Then, the context framework implements the typical pipeline for context recognition, either in a supervised way, through classification, or an unsupervised one through clustering (see Section 2.3). Starting from raw data, the first step is to extract the information and usually to represent it as a feature vector. The classification step uses pre-computed models, that were trained on user annotated data, to classify and put semantic labels on the feature vectors or to groups of feature vectors. On the other hand, the clustering process is not provided with any model and looks for similarity in the data to group them in clusters. To provide sense to those clusters, a labeling step is then needed to attach semantic labels to them. It can be done through inference rules based on expert knowledge or classification. As we will show in the Chapter 5, the symbolic representation of sensor data can be used as an alternative representation that can be directly input to a labeling process, in order to generate a semantic context, or fed to the more traditional preprocessing and feature extraction path. Moreover, different tasks can also be preformed on top of those symbols, such as multi-resolution long-term storage, anomaly detection, user profiling, etc.

Finally, at the top, the context-based applications can directly subscribe to the streams of semantic data output provided by the Context Framework. As we mostly focus on the data flow from the sensors up to the applications, the schema does not show the feedback channel from the application down to the API, but in a real implementation this could allow the lower layers to be aware of the applications needs and further adapt their behavior.

This architecture of sensor subsystems is not only designed for smartphones, but also for any kind of sensing devices such as smart-watches, wireless weather stations, mobile air quality sensor nodes, etc. They could benefit from the optimizations described further in this thesis. For this reason and to show the versatility of our approach, we do not restrict our experiments to data acquired from smartphones, but also use the ones from other kinds of mobile and static sensor systems.

1.5 Contributions of this thesis

As highlighted in the architecture description, our contributions to enable energy-efficient continuous context sensing devices can be summarized along three directions. First we explored the collaboration opportunities among smartphones to reduce the overall sensing load. Then, we optimized the scheduling of mobile sensors, based on different models of the sensors outputs. Finally, we designed an algorithm to encode the sensor data, making its processing more efficient while having a low impact on accuracy.

- **A collaborative sensing protocol**
  - We formulate the online/distributed collaborative sensing as a non-linear optimization problem, in order to reduce useless redundancy among the sensing devices via opportunistic collaborative sensing.
  - We design two technical solutions for our online/distributed collaborative sensing, including a basic strategy and synchronization methods.
  - We provide extensive experimental studies using two real world datasets, showing the impact of collaboration in terms of sensing error reduction.
  - Moreover, we provide a theoretical analysis of the online strategy compared to the offline optimal solution.
- **A Model-based optimal sampling algorithm**
  - We formulate the mobile sensor sampling problem as an optimization problem and provide optimal solutions, applicable for small instances.
  - We develop a two-tier framework, based on segmentation and sampling, as a general approach for the above problem.
  - We design and evaluate near-optimal heuristics for both segmentation and sampling in the case of a mobile sensor.
- **Finding states in sensor time series**
  - We propose a novel *state preservation index* which quantifies how good a cluster/symbolic configuration is (given the measurements) in preserving system's states.
  - We propose a novel *StateFinder* algorithm which combines the effectiveness of some well-known techniques, such as Markov chain and subsequences clustering.The algorithm is able to identify the states along a data stream in an online and unsupervised way, using a limited memory and computational footprint.
  - We carry out experiments using real-world datasets from different domains, e.g., human energy consumption and activity recognition, and present different application scenarios, e.g., state recognition, forecasting, and anomaly detection, to show the versatility of our approach.

In addition, to showcase the various concrete developments made in this thesis, we present in Chapter 6 a complete ecosystem for assessing the daily air pollution exposure, by fusing data collected on a smartphone and, at a larger scale, by mobile sensors deployed on buses.

2 State of the art

This chapter describes the different approaches that were researched to address the challenges from Section 1.3, namely trading off latency, accuracy and energy consumption. This research being technology driven, we start by presenting the hardware platforms that enabled context sensing. Then we present the upper layers, namely the sensors abstraction and the context inference.

Previous mobile sensing surveys [115, 149, 180] have taken different points of view, for example detailing more the inference pipelines and features extracted from the sensors or focusing on low-level sensing and implementations. To complement these perspectives, we present more recent work that emphasize the *energy efficient continuous context sensing on mobile phones*.

Before going further, it is also worth noting that the identified context by itself can be a very valuable piece of information that can in its turn be used for reducing the energy consumption of the mobile device as explained in the survey of Vallina-Rodriguez and Crowcroft [286].

2.1 Sensing platforms

2.1.1 Dedicated Hardware

Before smartphones were available with all their sensors, or to have a better control on sensor placement or the complete data management pipeline, researchers have built their own sensing platforms. In the continuity of wireless sensor networks, they designed autonomous sensor boards able to communicate locally or store the collected data. Being mounted on the belt or wrist, they were part of the large category of wearable computers. We present in this section the main generic context sensing hardware, but more specific ones can be found in Section 2.3, whether they are related to activity recognition, health monitoring or environmental monitoring.

The TEA awareness module from Gellersen et al. [90] is a small sensor board with light sensors, microphones, accelerometer, skin conductance sensor and a thermometer built-in. The microcontroller extracts cues from each sensor feature individually, abstracting the sensor from the context inference module. The board was designed to complement a mobile phone and provide it with context awareness. Whereas the Mediacup, from the same authors, is a standalone device, attached at the bottom of a coffee cup. It is also able to recognize its own context with accelerometers and a temperature sensor.

Also complementing a mobile phone without sensors, the SenSay [266] platform connects to an external sensor box, including an accelerometer, a microphone, a light sensor and a thermometer, all of them managed by a microcontroller. The latter samples the sensors at the maximum rate and keeps the last values in a cache, whose latest value is returned to the phone upon request. A decision module keeps track of the user state (normal, idle, uninterruptible or high activity) and triggers the corresponding action on the mobile phone. The wired serial connection between the board and the phone passes through a laptop for sensor data processing, but it was planned to remove this intermediary in a later version.

Similarly, the Mobile Sensing Platform (MSP) [56] is a small wearable device implementing activity recognition algorithms using data from the integrated sensors, like Gyroscope, barometer, BT, accelerometer, microphone, etc. In a first version, it was attached to a PDA for data processing and storage and in its second version a larger memory allowed all the data to be kept on the MSP device. By removing the need for a secondary device and some other optimizations, the batterie life was substantially increased, almost reaching a full day. The context inference algorithms were also run directly on the microcontroller of the second version.

Developed by Yamabe and Nakajima [308], in collaboration with Nokia Research Center Tokyo, the Muffin device holds a large number of sensors (thermometer, hygrometer, barometer, pulse and skin resistance sensor, compass, gyroscope, accelerometer, GPS, camera, microphone, etc.), behind a touch screen and managed by the Linux operating system. It has the same form factor as a smartphone, just a little bit thicker. No specific management of the sensors is discussed in the paper, the main focus being on fusing the various contexts from all the sensors (see Section 2.4).

2.1.2 Smartphones

The smartphones themselves having a large panel of available sensors directly on-board, such as the inertial sensors, positioning, or proximity sensors, they have become the default platform for context sensing. In his work, Liu [169] gives a detailed review of these sensors and how they were typically used in various applications.

However, the smartphones were not initially designed for continuous sensing, which triggered some research to overcome this issue. For example, Microsoft Research launched the SensoryPhone project to provide an energy efficient hardware and software platform, supporting continuous awareness of the user's context. They added a secondary processor, in reality composed itself of two processors, named Little-Rock, to directly interface the sensors and allow certain tasks to be offloaded from the main CPU [227]. Then they developed the SpeakerSense [177] application, relying on the secondary processor to detect if someone is speaking or not, and, only in the first case, to wakes up the CPU to identify the person. But no further details were given on the other sensor management policies.

The idea of using dedicated hardware to do the continuous processing of sensor data was integrated later on in commercial devices such as the iPhone. Apple presented in September 2013 its sensor co-processor (named M7 [1]), equipping the iPhone 5S. It runs continuously and processes accelerometer data for coarse grained activity recognition. Since then, the next generation of co-processor, named M8, has integrated some other sensors, such as a barometer for computing variation of altitude. More sensors can also be integrated by using the iOS accessory mode, like the digital multi-meter demonstrated by Brateris et al. [32].

The Android ecosystem, even if it is more often used for building demonstrators, does not however impose many constraints on the hardware requirements. It provides a unified view of the various underlying implementations of sensor hardware and allows the developers to query the sensors by specifying some high level requirements, such as the location accuracy (high, medium, low) for location or the sampling rate (fastest, game, normal, UI) for the accelerometer. Most of these APIs are asynchronous to help the operating system manage better the CPU sleeping cycles. Regarding the support of co-processors, it is introduced in the form of soft sensors, for example the Z series of Sony Xperia provides a step counter sensor[2]. The X8 chipset from Motorola[3] also provides built-in dedicated cores for always-on sensor processing, supporting the voice commands.

Future developments seem to be oriented towards more dedicated co-processors and system-on-a-chip (SoC), that are highly optimized for specific tasks.[4, 5]

2.2 Sensor abstraction

When dealing with various platforms and sensors, one may want to have an abstraction layer that provides a common and intuitive interface to access all of them transparently, whether they are on-board sensors or external ones. The current mobile operating systems already provide some abstractions, but they are limited to location sensors and activity in Android. Therefore, some research groups developed middlewares that allow other researchers to quickly prototype mobile sensing applications. In addition to providing the sensor readings, they also proposed different ways to describe the context one wants to identify, and gave access to the context inference results to the applications, while taking care in the background of using the appropriate sensors. Running all the context recognition pipelines on the phone

---

[1] http://www.apple.com/iphone-5s/specs/

[2] http://developer.sonymobile.com/2014/03/24/lower-your-xperias-power-consumption-with-the-new-sensor-co-processor-video-code-example/

[3] https://www.motorola.com/us/X8-Mobile-Computing-System/x8-mobile-computing-system.html

[4] http://www.pnicorp.com/products/sentral-sensor-fusion/

[5] http://www.audience.com/multisensory-processors

poses some challenges, as we have seen in the previous chapter. Moreover, when collecting data from a large group of participants, like in the participatory sensing scenario, there is a need for aggregating all the data in one place. For these reasons, many of the sensor abstraction middlewares also include a way to have their collected data sent to a server, either just to store it or to run more complex applications. Two approaches have been considered. The first one consists of having the sensing tasks defined on the smartphone application, either at programming time or through the user interface and the second one is pushing the tasks from the server, sometimes also selecting which device is more suitable for performing it. Finally, inspired by sensor networks and cloud computing, other abstractions based on service oriented architectures, such as Sensing as a Service (SaaS)[68, 74, 264], have been proposed to make the mobile sensing platform available from anywhere through the Internet, similar to the growing Internet of Things [282].

For accessing built-in as well as external sensors, through any network interface, the mSense middleware [144] defines two layers. The *Sensor* layer is hosting the hardware sensors' entry points, the network sensors' providers and simulated sensors, when they are not available in the hardware. Then, the *Data* layer is taking care of storing and fusing the data streams, for example in a dead reckoning module. Finally, the applications will subscribe to one of the available modules. The Dandelion [165] framework introduces the *senselet* abstraction for programming external sensors, independently of its processing hardware. Senselets allow to push some of the pre-processing closer to their source, thus improving the communication efficiency. Whereas, the Dandelion runtime, installed on the sensors and the smartphone is taking care of the coordination and communication and provides the application with the sensor data.

To simplify further the development of context based applications, the context inference algorithms were integrated directly in the sensor abstraction layer. For example, Seemon [126] is a framework for context monitoring defining context changes with a set of changes in the features extracted from the sensors. The context queries from the applications are translated using predefined thresholds and rules, and indexed for fast access when the sensor data arrives. To reduce the energy required, sensors are activated according to the ESS (Essential Sensor Set), which is re-computed from the current running queries at regular intervals. To be able to dynamically adapt to new context definitions and inference pipelines, the Dynamix [44] context framework allows new plug-ins to be loaded automatically at runtime, uploaded by the developer community. The plug-ins and their privacy policies can be configured by the user through the application interface. Other applications can access the resulting contexts directly from the Dynamix service, through a firewall also configured by the user. Similarly, the Mobile Sensing Framework [42] provides support to run personalized inference pipelines, but also reacts to other applications and manages sensor access conflicts, like for example releasing the microphone if an incoming call is detected. Flexibility is also needed in terms of available sensors, especially when those are wearables and can be easily added and removed by the user. For this purpose, Seeger et al. [255] proposed an event-based middleware managing sensors connected to a Body Sensor Network (BSN). A central event reasoner, on the smartphone, is taking care of identifying situations and creating derived events. All the events are then made available to the applications.

One of the typical uses of context-based applications by the research community being data collection, there was also a need to aggregate the data centrally by pushing them to a server. The first example is the Funf Open Sensing Framework [8], from the MIT Media Lab. This open source extensible framework for mobile devices provides a reusable set of functionalities for the remote collection of many data types. It is currently implemented as an application for Android and server-side scripts for uploading, parsing and visualizing of the collected data. Later, the Open Data Kit (ODK) [35] was developed as a set of tools for a non-technical audience aiming at recording any sensor available on mobile phones, including surveys for the users. In its second iteration, namely ODK 2.0 [36], it includes a module for uploading the collected data, either to a cloud storage provider or to personal machines running their own Aggregate server. Similarly, the AIRS [283] middleware enables the exploitation of the various sensing modalities available, in a highly configurable manner, also including users' input and annotations, while providing a way to collect the data centrally on a server. Following the design principles of the Global Sensor Networks (GSN), namely virtual sensors as stream processing classes and wrappers as data source abstractions, the MOSDEN [122] mobile application was designed to support crowdsensing scenarios. The plug-ins abstracting the mobile phone sensors allow accessing internal as well as external ones. Regarding cloud data aggregation, it uses the GSN servers as backend, relying on its push and pull restful API for communication.

Pushing the collected data on the server allowed the researchers to have at their disposal a much larger computational power and the possibility to run the actual application in the cloud. For example, the MobiSens [304] mobile application only implements lightweight activity recognition, while the server runs the heavy-weight processing and hosts a Lifelogger application. The server is also responsible for pushing the sensing policy and some incentives to the participants for providing annotations. Similarly, the architecture of HealthOS [163] is also based on lightweight adapters on mobile devices, whereas most of the computation happens on the server side. This system aims at abstracting commercial health applications and devices for fusing their data through various configurable pipelines before providing it back to the smartphone.

To address the need for a new application, or reconfiguring them all, each time the researchers want to conduct a different study, some mobile sensing applications were designed with the opposite paradigm. The server contains the description of the modalities needed and pushes to the participants the sensing tasks. This also allows to select only a subset of the participants and to perform optimizations centrally. For example, the PRISM [63] platform automatically deploys binary applications to the registered mobile devices satisfying predefined criteria. Special care has been taken for mitigating the privacy risks of the participants when running the code pushed by the server. Similarly, the Pogo [34] middleware allows researchers to submit applications in the form of JavaScript files and the Pogo administrators are taking care of assigning the Pogo participants. To avoid draining too much energy on the participant smartphone, piggybacking is used to synchronize the communication and sensing with other processes on the device. Also using javascript as a task description language, the Code In The Air [240] system proposes in addition a visual interface for end users to define their tasks without coding. It supports conditions on activities as well as places and automatically divides the client and server side of the tasks for deployment. Finally, the MECA [314] middleware exposes a declarative interface for cloud applications to describe the phenomenon they want to monitor. The backend server then translates the request into data needs and selects the devices and which of their sensors to enable.

2.2.1 Sensor scheduling

Having an abstraction layer in between the applications and the sensors allows a fine grained management of the available resources, optimizing the sensor duty cycles (ON/OFF cycles) and sampling rates to minimize the energy consumption while satisfying the applications' needs. The first approaches that were studied on smartphones were to find the best uniform sampling rate, according to the three dimensions presented earlier (energy, delay and accuracy). The CenceMe [190] application uses various duty cycles for the GPS, Bluetooth and data upload to reduce the power consumption. The authors show in their experiment the impact of duty cycle on accuracy and inference delay, and conclude that it should be adapted to the user or application need and to the context. This strategy is used for example in the EmotionSense [229] application, designed for recognizing speakers and their emotions, where the sampling rate of the GPS, accelerometer and microphone are adapted based on a set of rules. The A3R [310] application for recognizing activity also adapts the sampling rate of the accelerometer, depending on the current activity. The optimal rates and minimum set of features to extract are defined for each activity based on actual energy measurements.

However, as the context used for setting the uniform sampling rates is actually based on those same sensors outputs, the low sampling states will tend to absorb short activity changes and increase the detection delay. Therefore, a continuous adaptation of the sampling rate and defining when the sensor should be sampled next, based on the current context knowledge, is an optimization problem which attracted many researchers. For example, Wang et al. [296] formulated the problem as a constrained Markov decision process, where the model decides when to perform the next sample according to the user's states. They also show the improvements over the uniform sampling. Similarly, Lu et al. [176] also implemented a Markov decision process for duty cycling the GPS in their sensing application, Jigsaw. But for the other sensors, microphone and accelerometer, the duty cycling is continuously adapted, increasing the sleeping time when no activity is detected and decreasing it otherwise. Rachuri et al. [228] conducted an extensive study, for various sensors, on possible back-off and advance functions for varying the sampling rate. They show that no single solution works for all sensors in all contexts and discuss a dynamic algorithm for selecting the best one in each situation. The same authors also implemented SociableSense [230], a software for studying interactions and sociability among users. To improve the energy efficiency, linear reward-inaction is used to learn a sampling policy that minimizes the number of missed events. They also show that it outperforms the other back-off and advance functions (linear, quadratic and exponential).

Regarding participatory sensing platforms, more specific optimizations can be applied to schedule the environmental sensors as well as the location ones. For example, in the USense [4] middleware, participants can specify their own constraints on the percentage of energy dedicated for sensing and an adaptive sampling scheme is defined accordingly. Whereas, in PSense [14], the mobile application schedules the GPS according to the distance to the available nearby queries from the server. Similarly, the geo-fencing application from Man and Ngai [185] selects the sensing granularity according to the distance to the nearest region of interest.

More recent approaches leverage the numerous sensors available on the mobile platform. As Schirmer and Höpfner [252] proposed in their work, two methods can be applied, namely sensor substitution and sensor triggering. The first one aims at using a cheaper sensor, when available and providing enough accuracy, instead of using an energy-hungry one, both sensors contributing to the same contextual dimension (see also Table 2.1).

For example, the EnTracked$_T$ [136] framework is using the compass and accelerometer to estimate when the users are changing their trajectory and then scheduling a GPS measurement. Also for location tracking, the FreeTrack [55] application learns from the user's habits and records the cell-towers as an input for triggering the Wi-Fi based positioning or the GPS. At a higher level the ACE [200] middleware provides the application with semantic contexts, such as "is driving" or "is alone". By using some rules mined from the past observations, it can infer certain contexts from others and uses them to find cheaper conditional sensing scenarios. The Senergy [129] context API allows the programmers to specify their priority among latency, accuracy and energy use and then decides which sensors and which method will be used to provide the context information. Finally the phone sensors can also be substituted by external sensors, as presented in the METIS [231] sensing platform. It opportunistically leverages the available sensors in the infrastructure of a smart building for reducing its own sensing cost, selecting the sensors in the same room as the mobile device.

The second method, namely triggering, consists of using other contextual dimensions to trigger an expensive sensor, based on some rules, either learned or explicitly given. For example, the Energy Efficient Mobile Sensing System (EEMSS) [295] allows the programmers to describe the situations and which sensors need to be used for detecting a possible transition to another situation in the configuration files, relying on some expert knowledge. In the RAPS [209] application from Paek et al., the GPS is triggered based on the user activity, obtained from the accelerometer, but also considering user's mobility history. It also keeps tracks of the cell-tower's received signal strength each time the GPS fails to obtain a fix to build a black-list, to prevent turning the GPS on if it has a low probability of success. Similarly, the LifeMap [54] context provider uses the accelerometer to identify if the user is moving or not, and then selects the appropriate minimum sensor set. More recently, Pendao et al. [217] also implemented a motion detection algorithm, based on the accelerometer. They studied in detail the impact on the battery life of various sensing policy, based on periodic sampling and motion detection. But those method rely heavily on the developers and researchers defining the appropriate set of rules for triggering energy-hungry sensors. Therefore, Li et al. [160] built an inference model of context stability that takes care of triggering the heavy-duty sensors when the probability of a context change is high enough, given the data collected by the light-duty sensors.

It should also be noted that these two methods are not mutually exclusive and can be combined, like for example in the middleware developed by Zhuang et al. [322] which aggregates the location sensing needs from the applications and merges them to reduce the GPS usage. For this purpose they implemented four methods, namely substitution, as defined above, suppression and adaptation, which are similar to the triggering approach above, and piggybacking which improves the synchronization of the application requests. Similarly, the SenseMe [25] system performs continuous context identification, applying the same scheduling optimizations to be able to provide a context API to third party applications, including location, activity recognition, social and environmental contexts.

2.3 Context inference

The context is a multidimensional concept that can be built from many different sensor inputs. But the mapping from sensors to the dimension they can contribute to is not always straightforward. Table 2.1 lists the commonly found sensors on the various mobile sensing platforms and their contributions to the usual context dimensions.

Direct (D) contributions are obtained directly through the sensor output, like the GPS which gives the latitude and longitude. Some sensors' output however needs to be processed and, using some learned model, mapped (M) to contextual outputs. For example, the accelerometer values have to be processed and a classifier would then be able to tell the user's activities. Finally, some contributions are obtained by combining (C) the sensor data with some other dynamic data source, being either other sensors on the device or external sensors. For example combining the accelerometer for counting the steps and the compass for the heading makes it possible to track the motion of the subject, this is called dead reckoning. It is important to note here that we consider in this list the *sensing* modality only. For example Wi-Fi is only considered for scanning the neighboring access points, not for communication, and similarly for Bluetooth.

We also limit our survey to first level contexts, such as location and activity. Higher level contexts, such as places and routines, that can be further inferred, are out of the scope of this survey.

	Activity	**Location**	**Environment**	**Proximity**	**Health**
GPS	M [161]	D	M [210, 307]		C [179, 229]
Wi-Fi	M [268]	M*		D [97, 188]	
GSM cell tower		M [48]		C [159]	
Bluetooth		M[318]		D*	
NFC		M[107, 202]		D*	
light sensor	C [189, 213]	M[12, 147]	D [203, 321]		C [179]
camera	M*	M[299]	M [166]	M*	M [254]
microphone	C [189, 213]	M [12]	D *	M*	M*
accelerometer	M*	C*	D [82]	C [110]	M [179, 184]
gyroscope	M*	C*		C [188]	
compass	C [213]	C*			
barometer	C [213, 248]	D	D [203]		
thermometer	C [189, 213]		D [203, 208]		D
hygrometer	C [213]		D		
gas sensor			D*		
ECG	M [153, 213]				M*
breath rate sensor	M [153, 213]				M*

Table 2.1: Non-exhaustive list of available sensors on mobile platforms and how they can contribute to the usual context dimensions. D: directly, M: mapping through a model, C: combined to other sensors. The grayed cells indicate the most commonly used sensors for those contexts. *see the sections below for a longer list of related works using these sensors.

2.3.1 Activity recognition

During the last 15 years of research, human activity recognition has been studied under many aspects. In addition to the challenges mentioned earlier, this research had to address a number of issues related to machine learning, such as feature selection or building realistic and representative training sets. In their survey, Lara and Labrador [152] provided a large overview of the human activity recognition systems using wearable sensors, including the smartphone based approaches. After describing the general design of such a system, namely data collection - feature extraction - model training - classification, they gave an overview of the feature extraction and classification methods used. Most of the surveyed methods used a direct mapping from a feature vector to an activity or probability of an activity. The feature extraction is performed either through statistical methods, summarizing a window of sensor data into representative values such as its mean, variance, energy, etc., or through structural detectors trying to match predefined or learned patterns. The learning part itself makes use of the whole set of available classifiers, from decision trees to neural networks and Markov models, including also support vectors machines and many others. The comparison of their performance varies from one work to another, making it difficult to draw a clear winner. This is also linked to the difficulty of making quantitative comparisons of the methods themselves, mostly due to the lack of standard procedure and datasets. From the evaluation they show in the end of their work, we can already conclude that adding more sensors and especially well placed ones increases greatly the accuracy and generality of the methods. This is also shown in the experiments of Gao et al. [89]. A typical example is the system developed by Parkka et al. [213], consisting of 22 different sensors, such as GPS, light sensor, microphone, accelerometer, thermometer, heart rate and breath rate sensors, etc.

In addition to those accelerometer based approaches, Liao et al. [161] proposed to perform significant places identification at the same time as activity recognition from the GPS output only. Using hierarchical conditional random fields, they were able to model the links between activities and the location where they happened, thus recognizing activities like working, walking, taking the bus, driving, etc. In their *Telepathic Phone* experiment Sigg et al. [268] demonstrated that it is possible to recognize some gestures, walking speed and presence of the user just by using Wi-Fi signal strength information.

Among the latest advances on human activity recognition and the ongoing research, we can find two main directions. The first one is trying to apply more advanced or recently invented machine learning methods to increase the recognition performance. For example, Extreme Learning Machines, a kind of artificial neural network, has been used by Wang et al. [293] on top of a bag of linear dynamical systems as features. Another example is the semi-supervised method from Tran et al. [281]. It is based on partially hidden discriminative models, as opposed to the generative Hidden Markov Model generally used for representing transitions between activities. In this case they show that their discriminative model outperforms the generative counterpart.

The second approach is trying to improve on the generality of the methods, and thus their applicability in real-life scenarios. Most of the current solutions are designed specifically for a given problem, in a given environment and sensor setup, and with a limited population, making their comparison and usage in widely deployed applications more difficult. Therefore researchers have been trying to generalize the solutions. For example, by including a more diverse population, Capela et al. [41] were able to build a more robust classifier from a single smartphone accelerometer signal, identifying the relevant features for each group (able bodied, senior, stroke) and for the whole population. An automated way of grouping similarly behaving people was also proposed by Lane et al. [151], by creating community similarity networks from the sensor data and personal information and training personalized classifiers. By creating virtual physical environments, Poshtkar et al. [224] were able to generate automatically large labeled datasets, allowing them to test various solutions quickly. Reducing the supervision of the classifier, like in [281] and [26], is also a way to get to an online version that would be acceptable in terms of user interruption and input needs, as there is currently no unsupervised (or semi-supervised) online method.

Among the efforts made in reducing the energy consumption of the mobile devices used for context inference, most of them tried to adapt the sensor scheduling and sampling rate to the current activity. Some others have explored the trade-off of off-loading the classification and computation to dedicated servers. And finally some of them have tried to simplify the processing, taking into account its cost in energy. For example, Kobe [57] is an implementation of a feature classification system using the cloud to compute the most effective parameters according to predefined configurations (connectivity, CPU load, phone model). And then, at runtime, it switches between the parameters and it is able to offload part of the classification to the cloud as long as it respects the constraints of the user (energy budget and latency). Sensor parameters like scanning rate or measurement precision are also precomputed in the cloud and changed in real-time according to the current configuration of the phone.

Regarding human activity recognition, we should also mention the huge amount of work using external sensors, such as video cameras [223], presence sensors [175], floor accelerometers [181] that can be deployed at the house or building level. Most recent developments in this field were made thanks to the large availability of 3D camera, that include depth [7]. Moreover, the usage of newer kind of sensors opened the door to more fine grained and less intrusive monitoring. For example Sigg et al. [269] were able to use the variation in the signal of a local FM radio station to identify the motion activity of a user in a room. Home electrical appliances can also provide useful information regarding the activity of the inhabitants, through their usage and monitored via their electricity consumption [22], or even their noise [215].

2.3.2 Location

As the Global Navigation Satellite System (GNSS) is providing direct information about location, it is nowadays typically used in most implementations of location-based services. But its high power consumption, long time to first fix and unavailability indoors has pushed the researchers to find alternatives. First, the most represented one for the mobile platform is using radio fingerprints, especially from Wi-Fi networks [81], but also from the GSM network [48]. The principle of fingerprinting is to associate a place with a recognizable and hopefully unique pattern of sensor data. In their survey on indoor positioning systems Liu et al. [167] and Gu et al. [94] presented their very first implementation, namely RADAR [13] and a few years later, improving the fingerprints, Horus [315]. They work in two phases, both having strong limitations to their wide adoption. In the first phase they need to learn a complete mapping of the fingerprints. This is sometimes called war-driving as it started with people driving cars with Wi-Fi antennas. This is costly for large areas and even worse when the Wi-Fi infrastructure is likely to change over time. The second phase is the online query of the learned mapping, which is made difficult due to the high variability of the Wi-Fi signal, influenced by the layout of people and furniture in the neighborhood.

In recent years, several solutions have been proposed to overcome these limitations. First, Simultaneous Localization And Mapping (SLAM) systems have been developed [84, 116] to avoid the need for a training phase. To this purpose, the EZ localization algorithm [52], based on a genetic algorithm, is merging all the reports of fingerprints, some of them having absolute coordinates attached, and centrally reconstructs the map by constraining the distances between users and access points. Another approach by Dousse et al. [76] is to directly apply clustering on the fingerprints and associate the clusters with a semantic tag, as a place of importance. But by using only Wi-Fi signal strength as input these methods have a limited accuracy that can be improved by using more information from the Wi-Fi physical layer. In PinLoc [257] and its continuation, CUPID [258], the channel frequency responses are directly clustered, and the system is able to estimate the angle of arrival and the distance to the access point, dealing with the multi-path effect. But these methods need special hardware unlike the one from Martin et al. [186], who first proposed an implementation working on a smartphone, having a precision of up to 1.5 meters.

To improve the accuracy of the fingerprint maps, some researchers have also combined it with other sensor inputs. For example, [168] and [134] used acoustic ranging to compute precisely distances between peer devices. In their LiFS system, Yang et al. [311] used the accelerometer as a pedometer to help mapping in 2D the Wi-Fi fingerprint space. In their continuation work [50], they included the concept of virtual rooms, which are k-means clusters of fingerprints. Werner et al. [299] combined a Wi-Fi based localization with an image recognition algorithm, able to recognize a place from the camera of a mobile device.

Finally, once the map is built, new fingerprints can be recognized and associated to a location. But depending on the features and distance measures used, the results can vary greatly [81]. Therefore Sharma et al. [260] proposed KARMA, an online method for recalibrating the maps by taking into account the dynamic changes of the environment, affecting the Wi-Fi signal. Yu et al. [316] also showed those variations of the fingerprints, and compared the performances of 2.4 GHz and 5GHz Wi-Fi channels, the latter being more accurate for indoor localization.

Fingerprints are not limited to radio signals, they also include more exotic modalities, such as sensing the ambiance to recognize a place, based on the sound, light, and accelerometer fingerprints [12].

The uniqueness of fingerprints being sometimes an issue, for places without Wi-Fi coverage, or to get more accurate localization, researchers also considered artificially enhancing the fingerprints by spreading beacons around. These are usually based on the Bluetooth communication channel, which is more accurate than Wi-Fi as shown by Zhao et al. [318], or RFID like the SpotON [107] and LANDMARK [202] systems. But they can also rely on light, as demonstrated by Kuo et al. [147].

Another alternative to the GPS, often used to complement it, is *dead reckoning*. The principle is to use inertial information from the device to compute its displacement, for example by knowing the speed and bearing, it is possible to rebuild the trajectory. In contrary to the methods shown above, this method gives only a relative positioning, that can be made absolute by knowing some reference points. For example, in the CompAcc application developed by Constandache et al. [61], the assisted-GPS is triggered when it observes a too large dissimilarity between the reported path and walking paths from a map. As it does not rely on external input, it has some advantages for tracking motion and users trips in places without recognizable fingerprints. But it relies greatly on the accuracy of the measurements taken on the heading and the estimated distance covered, and drifts can lead to very large errors. In his survey Harle [102] presents the various methods used for detecting steps and heading, using the accelerometer, gyroscope, compass or foot pressure. However, the compass is easily influenced by its close environment, leading to large errors on the position estimation. Therefore Roy et al. [247] developed the WalkCompass, by integrating the motion patterns of walking with the identification of the phone placement on the body. They showed an improvement in the estimation of the user's walking direction, directly benefiting dead reckoning systems.

To take advantage of the fingerprinting and dead reckoning complementary strength, many works have been conducted on fusing their outputs. For example, Rai et al. [234] and Hong et al. [111] used particle filtering for estimating the location of the user on a map and, this way, being able to geolocate the Wi-Fi fingerprint more easily. Seitz et al. [256] used a Hidden Markov Model whereas Liu et al. [173] applied a Markov Chain Monte Carlo to integrate the fingerprint observations. In their UnLoc system, Wang et al. [290] extended the fingerprints concept to landmarks, by including other sensor recognizable output, similar to the work of Chen et al. [49], based on a Kalman filter for the data fusion. For example a turn in a corridor or an elevator can be recognized from the gyroscope, respectively with the barometer. In addition, the Travi-Navi application from Zheng et al. [319] integrates the smartphone camera to help in recognizing places and detecting when the user is deviating from a defined path.

2.3.3 Environment

The environment represents all the external influences that can affect the user. We distinguish it from proximity sensing, described in the next section, which represents the other surrounding entities, which can be other people or physical objects. Environment and location are very often correlated as the measured phenomenon are space- and time-dependent. Therefore all the previous methods used to get the user's location can be indirectly used here if one has a model of the field of interest. Since the recent rise of participatory sensing, measuring environmental parameters from mobile devices gained a lot of traction. They can indeed be used for collaboratively building the model and maps from a very large number of geo-located measurements, thus allowing users without those sensors to also obtain information about these parameters at their location.

Most environmental parameters can be sensed directly from the available sensors, such as temperature, humidity, air pressure, noise and light. But also by extending the sensing capabilities with external devices such as air quality sensors. Therefore most of the research has been focused in gathering and aggregating the measurements, with the idea of alleviating the poor quality of the reports by their huge quantity. In addition to air quality, which is surveyed in Section 6.2.2, the most present topic in this literature is urban noise monitoring, made easy to deploy by the large number of microphone equipped devices, namely all mobile phones.

In his evaluation of the NoiseSpy mobile application, Kanjo [127] presents the challenges of deploying such a smartphone based distributed monitoring system, namely the incentives of the participants, accuracy of the measurements, the need for managing the local phone resources and the privacy of the participants. Rana et al. [237] developed a complete end-to-end mobile phone based noise mapping system, named Ear-Phone. They particularly focused on the reconstruction of the map, using compressive sensing, from incomplete measurements reported by the participants. Santini et al. [250] made an extensive study on the actual capabilities of off-the-shelf smartphones, showing that calibration can be an issue, depending on the hardware abstraction and direct sensor data accessibility from the API, as the noise canceling filters cannot always be bypassed. They also raised some concerns about real deployments and the impact of the phone placement (pocket, bag, hands), as opposed to controlled studies, mostly used because of their more predictable results. Maisonneuve et al. [182] proposed NoiseTube, an application used to measure the personal and collective noise exposure, including manual tagging of the place and noise cause. In their continuation works [183], they expanded the semantic tagging and made it partially automated, from the time and location of the participant and, more recently [73], they conducted a larger study to validate their system.

Complementing physical sensor output with human reports was also one of the focuses in the work of Demirbas et al. [66]. They used Twitter as a communication channel for sending measurement queries and responses about the noise level and weather conditions, mixing automated sensor Tweets with manual reports in a predefined format. In the Atmos application, Niforatos et al. [203] also used the temperature, air pressure and light sensors for assessing the weather conditions, which can be used for complementing current weather forecasting infrastructure at the very local level. The idea of using the internal battery thermometer for measuring the air temperature was also explored by Overeem et al. [208]. They showed in their study that a correlation exists, but many other parameters have to be taken in consideration, such as the phone usage, position and whether the user is indoors or outdoors.

Some other interesting natural phenomena have been, or are currently being monitored, through the use of mobile phones. For example, Faulkner et al. [82] presented an earthquake detector based on the smartphone accelerometer and Pankratius et al. [210] are designing a system for measuring the ionosphere by monitoring the slight variations of the GPS signal.

In general the main issue of environmental sensors is their huge dependency on the user's context, and this has to be taken into consideration for obtaining better quality data, whether it be for participatory sensing or direct local usage.

One typical context dimension that lies between location and environment is the indoor versus outdoor identification. Some algorithms introduced before the era of smartphones were already able to identify if a picture from a digital camera was taken indoors or outdoors, aggregating low-level features like in the work of Szummer and Picard [274] or using the straightness of edges like in the work of Payne and Singh [216]. More recently some algorithms were developed to run on the smartphone's camera. For example, Lipowezky and Vol [166] used the white balancing parameters and boosting to classify the pictures directly on the device.

Using a dedicated temperature sensor, John Krumm [123] was able to correctly classify the position of the device 81% of the time based on generally available online outdoor temperature, increasing to 91% when comparing to an actual outside thermometer.

Since the location can be used for determining if the the device is indoors or outdoors, many of the methods seen in the previous section can be used for this classification. But recently some researchers studied algorithms for this specific task. Xu et al. [307] used the GPS signal noise, in addition to the light intensity, to infer if the device is indoors or outdoors. Canovas et al. [40] built a boosting binary classifier based on Wi-Fi fingerprints. But the GPS, and Wi-Fi scanning to a lesser extent, being known to be power hungry, one may want to use cheaper sensors and avoid the need for war-driving as in the second method. For this purpose, the IODetector from Zhou et al. [321] uses a combination of detectors based on light intensity, cellular signal strength and magnetic field variation to classify the location of the mobile phone as indoor, semi/outdoor or outdoor. In their experiment they successfully used the detector to switch between indoor and outdoor positioning methods, namely GPS and Wi-Fi fingerprints.

2.3.4 Proximity

We consider in this section proximity information as the detection, identification, and sometimes the precise relative position of other entities in the vicinity of the user. These entities can be other people or more generally physical objects. Two distinct approaches were used for this purpose, either the other entity can be directly seen from the signal it emits or its response to a signal, or both entities infer their collocation by observing a similar environment or even directly by knowing their own location.

The first and primary task that can be achieved by proximity sensors is presence detection and, more interestingly counting the number of other devices. For example Naini et al. [198] equipped a few volunteers with smartphones logging the Bluetooth devices they detect to estimate the population size at a music festival, with similar methods as biologists use for estimating animal populations. Based on audio tones, Kannan et al. [128] had a completely different approach where the devices communicate with their neighbors, in a distributed fashion, to converge to their actual count. Also using the microphone, the Crowd++ application [306] is able to count the number of people talking in a certain place, in a completely unsupervised way.

The unique identifiers used in the detection process can actually be used to map the signals to entities. This has been first used for identifying objects, by equipping them with RFID tags that can be read one by one. Vogt [288] proposed a method for optimizing the tag reading process to enable it to identify multiple objects with passive RFID tags. In their RFIG lamps, Raskar et al. [239] augmented the tags by making them sensitive to light and adding geometry information from the reader, composed of a camera, a projector and a RFID reader. The information from the tags allowed them to project images and track the deformations of the objects. Augmented reality games also took advantage of this technology as in the Pac Man version of Cheok et al. [51], where the real world interaction of the user with tagged objects was reflected in the game.

As smartphones started being equipped with a camera, other kinds of tags appeared, like the visual ones. In their study Toye et al. [280] evaluated the viability of printed visual codes for users interacting with their environment. They showed that the participants learned quickly how to scan the codes and received positive feedback. Among other works with visual tags, Rohs and Zweifel [246] demonstrated a way to combine them with motion recognition, to create new kind of interaction with bus stops or vending machines, for example by rotating the phone after reading the tag to get the arrival time of the next bus. Further advances in smartphone capabilities enabled them to directly use the camera to recognize the objects [91], without the need to transform them to make them recognizable.

When detecting nearby people, the identifiers they carry, mostly their Bluetooth enabled smartphone, allow us to link them to their online social profiles, such as Facebook or LinkedIn, revealing another dimension of context, namely the social context. In their survey Schuster et al. [253] identified the different dimensions of the "Pervasive Social Context", based on their own taxonomy: STiPI, for Space, Time, People and Information source. The intersection with this survey is to be found in S1 (small scope), T1 (short-term activity), P1 (individuals) and I1-3 (From pervasive sensors, potentially integrated with social networks), which mostly contain Bluetooth and RFID proximity detection.

For example, the WhozThat social application from Beach et al. [20] collects neighboring people identifiers and fetches their online profile for finding common hobbies or musical preferences. Ben Mokhtar and Capra [24] used physical and social proximity to assign tasks to co-located friends, developing the concept of social computing. In a more pragmatic application, Rafael et al. [232] used the detection of nearby people to help blind people engage in social interactions. The link between physical proximity patterns and social interaction has also been studied by Do and Gatica-Perez [75]. They presented a method to infer the social groups from Bluetooth scans and the temporal context, and predict interactions.

Due to its limited spatial propagation, sound, and especially the voice, has also been used to identify nearby people, like in SpeakerSense [177]. Nakakura et al. [199] and Wyatt et al. [305] went further by identifying the conversation groups and who is talking to whom.

After detecting and identifying the other entities, the researchers designed methods to compute their relative positions. On one hand some of them used the distance and the angle of arrival as in the external device developed by Hazas et al. [105], where a radio signal and an ultrasound allowed them to obtain relative positioning with other devices within an error of less than 10 cm for the 90 percentile. On the other hand some of them used the distance between nodes to build a graph and then projected this graph onto a 2D plane, like the Virtual Compass [18], which is based on Bluetooth or the BeepBeep [218] protocol, able to compute distances between devices by emitting sound beacons.

On the social side, the relative positioning of two individuals enables the detection of face-to-face interaction. For example, Van den Broeck et al. [287] deployed active RFID badges at two conferences, allowing the participants to explore their social graph, combining real-life encounters and social network inputs. Instead of relying on external hardware, Matic et al. [188] used the smartphone's integrated gyroscope and Wi-Fi signal strength to measure the orientation, respectively the distance, between two persons, inferring if they are facing each other or not.

For the cases where the needed proximity sensors are not available, an alternative to direct detection of neighboring devices was also envisioned by some research groups. Namely, they assumed that if two devices are sensing similar environments, or locations, they have a high probability of being collocated. One of the first attempts in this direction, by Holmquist et al. [110] was to use the accelerometers and if the shaking patterns on the two devices matched, they will get connected. The NearMe system [145] compares the Wi-Fi fingerprints sensed from several devices to figure out if they are nearby, without needing an absolute location system. Instead of giving a fine-grained estimation of the distance, Carlotto et al. [43] proposed a Gaussian Mixture Model classifier for three proximity levels, namely high (same room, < 5m), medium (same floor, 5m-15m) , low (same building, > 15m). At a larger scale, in their PeopleTones application Li et al. [159] used the GSM fingerprint to infer collocation and informing the user with a specific pattern of vibration of the phone, depending on the identity of the other user. Whereas, Gupta et al. [97] used a Wi-Fi based localization technique to match mobility traces of users and determine their affinity. They also cluster co-presence at a place to infer informal social groups.

Somehow related to proximity, researchers have recently explored the case of phone placement, allowing them to detect if they are in a pocket, lying on a desk or in a bag. In their Phoneprioception app, Wiese et al. [300] relied on the accelerometer and a light sensor, augmented with the capacitive screen input and a multi-spectral sensor to infer the phone placement with 85% (pocket/bag/hand) to 100% (enclosed) accuracy. They also conducted a large user study, asking participant where they keep their phone in various contexts. However the Phoneprioception runs on some dedicated hardware developed specifically for the study, as some sensors were not available on commercial off-the-shelf smartphones. On the contrary, the Sherlock [312] platform continuously recognize the micro-environment of the phone (placement, backing material, interaction), using only the on-board sensors, such as the accelerometer, microphone, camera, gyroscope and light. A hierarchical detection algorithm is used to trigger only the needed sensors, thus reducing the long term consumption of the system.

2.3.5 Health

We regroup in this section all the contextual information about the user's physical and mental state that can be gathered through the sensors. This includes the directly observable vital signs, such as the heart rate, breath rate, blood pressure, or oxygen concentration in blood and the indirect parameters like sleep quality, mood, happiness or stress. As defined by Istepanian et al. [118] 10 years ago, this field of study has been called m-Health, standing for "mobile computing, medical sensor, and communication technologies for health care." This definition depicts the main research topics and challenges that were addressed in the field, namely building small, less intrusive and accurate sensors, enabling a reliable, efficient and versatile communication for the on-body wireless sensors, and developing an end-to-end system that makes the collected data exploitable. Even if most of these challenges have been addressed by researchers, m-Health has not yet shown its full potential. Recently, the U.S. National Institutes of Health gathered m-Health specialists for discussing the need for rigorous empirical and theoretical foundation in this field [146]. Indeed, many studies and developments have been made independently and with various goals, showing how large the topic is, but without collecting enough evidences on the efficacy of m-Health. As shown on Figure 2.1, they also outlined the trend of m-Health, going from measurements (getting the data) to diagnosis (understanding the data), treatment/prevention (using the data) and a global service (scaling up).

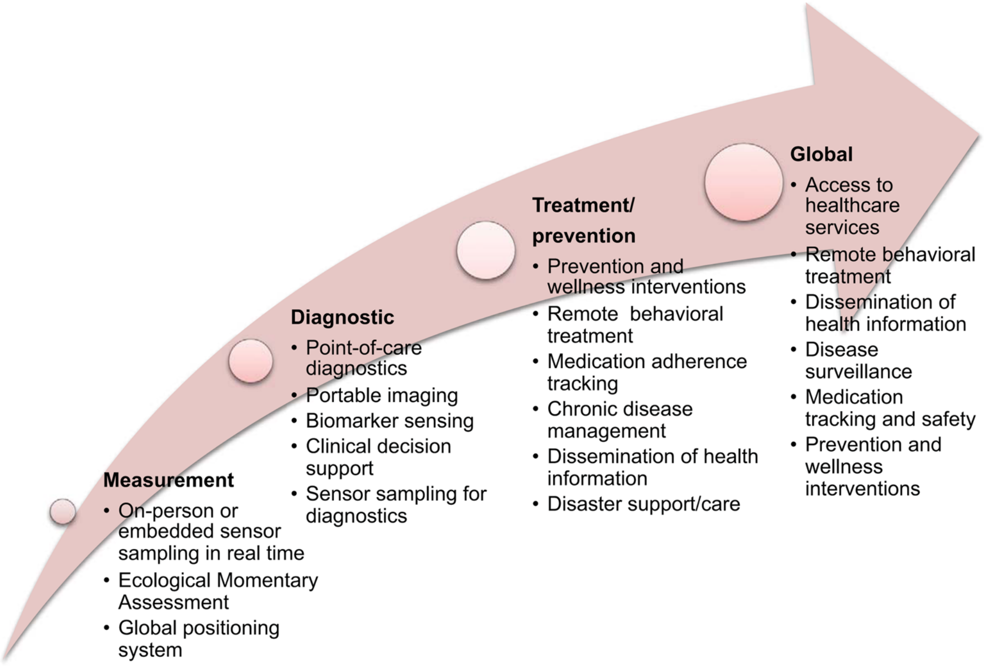

Measurement
• On-person or embedded sensor sampling in real time
• Ecological Momentary Assessment
• Global positioning system
Diagnostic
• Point-of-care diagnostics
• Portable imaging
• Biomarker sensing
• Clinical decision support
• Sensor sampling for diagnostics
Treatment/ prevention
• Prevention and wellness interventions
• Remote behavioral treatment
• Medication adherence tracking
• Chronic disease management
• Dissemination of health information
• Disaster support/care
Global
• Access to healthcare services
• Remote behavioral treatment
• Dissemination of health information
• Disease surveillance
• Medication tracking and safety
• Prevention and wellness interventions

Figure 2.1: Evolution of m-Health applications (source [146, p. 229])

Regarding the first step, namely the data acquisition, Pantelopoulos and Bourbakis [211] surveyed the wearable health-monitoring systems, including the mote, smart textile and smartphone-based systems, evaluating their maturity with respect to their full potential. Among the "mature" smartphone-based systems combined with wearable sensors, the Personnal Health Monitor [157] allows the user to perform a heart attack self-test, by answering questions and using the ECG sensor. As for the HeartToGo system [207], it shows the user an in-depth analysis of its ECG in real-time, identifying arrhythmia with the help of a neural network classifier.

More recently, Postolache et al. [225] proposed a smart bracelet with a photoplethysmographic sensor (light based) and an accelerometer, that preprocesses the data before sending them to an Android phone via Bluetooth. The smart bracelet is able to extract the hearth rate and blood oxygen levels as well as computing some simple features, while being non-invasive and easy to carry compared to chest-bands and electrodes. These sensing platforms can also be used for supporting medical and psychological experiments, like the PsychLog platform [87] which is used for experience sampling. This method captures the participants behavioral, emotional and physical state in their natural environment, by recording their vital signs, and regularly (or randomly) prompting for filling a survey. To save energy the sensor data is recorded only in a time window centered on survey time.

Part of the acquisition of the data, the wireless communication challenges with the phone and the external sensors has also been addressed, preferring Bluetooth communication to Zigbee, as the later is not natively available on smartphones. For example, Zhong et al. [320] designed a wireless body sensor network, optimizing the Bluetooth communication for energy efficiency and pushing most of the computation onto the phone. In their work, Istepanian et al. [119] successfully implemented a sensor network based on the Internet of Things communication protocols and based on IPv6.

It is worth noting that some researchers investigated physiological parameter sensing in rather unusual ways. The Septimu [204] modified earbuds include a small microphone that is able to listen to the heartbeat and infer the heart rate. It also includes an accelerometer and gyroscope for activity level recognition, the processing being done on the phone. Scully et al. [254] studied the feasibility of using the mobile phone camera and flash to infer the heart rate, breathing rate and oxygen saturation. The users were instructed to put a finger directly on the phone's camera, with the flash light being on. They were then able to extract the physiological parameters from color change signal from the camera.

Some works based their system on the microphone only, like Larson et al. [154] who extracted specific features for cough detection that prevented reconstructing the user's voice. The BodyScope [313] wearable device is maintaining a microphone on the throat to classify activities such as eating, drinking, speaking or coughing. Similarly, BodyBeat [233] identifies non-speech body sounds from a neck device.

The second step, namely understanding the data for a diagnosis, includes many kinds of inferences. Sleep quality has been studied with the HealthGear system [206] of Bluetooth connected sensors, more precisely with a blood oximeter and two methods for detecting sleep apnea, namely based on thresholds or based on a spectral analysis of the oximeter signal. Detection of insomnia was also performed within the OPTIMI [184] project through small heart rate sensors and accelerometers carried like a necklace. The time and frequency features are extracted directly by the sensors as well as the computation of the activity level and heart rate variability signals. Using only the phone's microphone, Hao et al. [101] implemented iSleep, a mobile application which classified the sounds during the night into either noise or sleep events, like moving, coughing or snoring. Then a score of sleep efficiency was computed and compared to a questionnaire, showing a very good match.

On the psychological side, Ma et al. [179] proposed a framework for mood analysis, called Mood Miner. They collected data from the accelerometer, microphone, GPS and light sensor as input for a classifier based on a factor graph, assuming mood has a Markovian property. As for LiKamWa et al. [162] they used the user interactions with the phone, including SMS, calls, applications usage as well as the location for inferring mood in their MoodScope application. The application then provides an API enabling other services to access the user's mood state. In EmotionSense, Rachuri et al. [229] implemented a speaker recognition subsystem, augmented with a location, activity and proximity modules as input for a Gaussian Mixture Model classifier for emotions. Other works, more specifically focused on happiness, can also be found in the survey of Muaremi et al. [195].

Another important aspect of one's mental state is the stress level. The AutoSense wearable system, measuring the heart rate and breathing rate, was used by Plarre et al. [221] to provide sensor input to a personalized classifier for perceived stress in the natural environment. However, the need for wearing additional sensors making it less practical for daily usage, researchers focused their efforts on the data already available from the phone. Based on the mobility traces, call logs and the social context from Bluetooth scans, Bauer and Lukowicz [19] were able to show a significant change in participant's behavior between stressful and non stressful situations. Furthermore, Sano and Picard [249] combined these data with a small wrist sensor for skin conductance. However they showed that the wrist sensor's data did not improve much the classification results. Using the microphone, Lu et al. [178]proposed StressSense, an application for recognizing stress level from the voice features, based on an adaptive GMM classifier using Maximum A Posteriori estimates. The classification pipeline was also designed for saving energy, only running on-demand the heavy computation parts and taking advantage of the phones' multi-core CPUs. Bogomolov et al. [31] also showed in their experiment that weather conditions are also an important factor influencing the participants' stress level.

Finally, the third step, namely making use of the data and closing the loop, was studied in numerous intervention studies as surveyed by Klasnja and Pratt [137]. However, most of those works were based on SMS communication, the patient reporting his health status. Among the works that use sensor and instant feedback, Morris and Guilak [193] estimated the stress level of the user from the heart rate and the mobile application provided advices to manage the stressful situations as soon as they were detected. Another large category of applications that provide feedback are the wellness and sport coaching applications, mostly because they are less risky to experiment with than the ones concerned with health issues. For example MPTrain [205] is a sport training application monitoring the heart rate through a chest-band and counting steps from the accelerometer in order to calibrate the exercise intensity by changing the music played. Ubifit Garden, developped by Consolvo et al. [60], provides a glanceable display of a garden, where the achievement of a training goal is shown as a flower. The authors conducted several studies [138] showing the usefulness of this kind of constant reminder for initiating a real behavioral change of the user in the long term.

2.4 Sensor fusion

The principle of sensor fusion is to combine data from various sensors to improve the accuracy or extract more information, over what would be available from each sensor individually. As we have seen in the previous sections, when several sensors can be used for the same contextual dimension, they can either be used alternatively, choosing the most accurate or energy efficient one, or in a complementary way. In this section, we present the various methods that have been used to fuse sensor data at all the processing stages, whether they be for the same contextual dimension or for different ones.

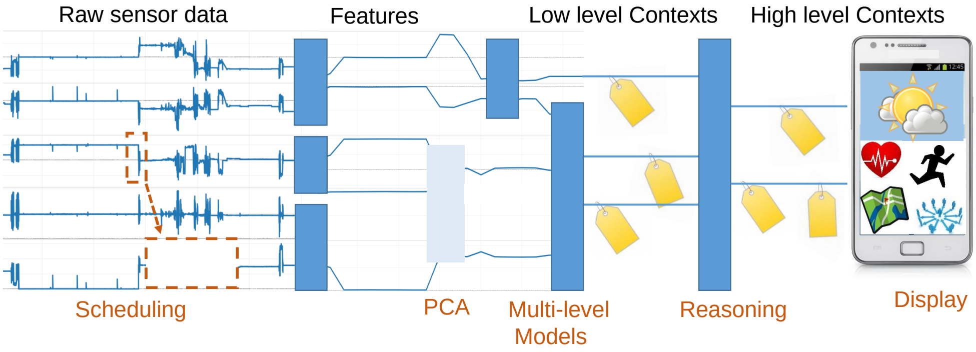


Figure 2.2: The context processing stages and their corresponding fusion methods.

First, while manging the sensor schedule, their complementarity can be used to trigger or disable other sensors, as in the state-machine proposed by Shimizu et al. [265], based on the accelerometer, microphone and Bluetooth output. Some other similar approaches have been also mentionned in Section 2.2.1. Then, when processing the raw data from the various sensors, Guiry et al. [95] suggested to apply a principal component analysis dimensionality reduction over the feature vectors from all sensors, thus building new feature vectors that summarize them.

Most of the sensor fusion approaches are applied at the classifier construction level. The feature vectors including data from all sensors are directly input to build the various kind of classifiers, with or without the feature selection step. For example, Ganti et al. [88] used a hierarchy of Hidden Markov Model to fuse acceleration and audio samples for activity recognition. Whereas Liu et al. [171] merged the breath rate and acceleration with an SVM classifier for activity and SVR for estimating the energy expenditure. Regarding location and transportation modes, Reddy et al. [242] integrated features from the GPS and accelerometer into their two stage classifier composed of a decision tree and a discrete HMM.

Based on the Muffin sensing device, Yamabe and Nakajima [308] developed the citron framework, following the blackboard architecture. The idea is to have a single database of resources, consisting of sensor data and processed contextual inferences on which simple modules analyze context and send back their inference results to the database. This allows them to abstract multi-level inferences and sensor fusion, and making them adaptive to the available resources. Similarly, Dunkel et al. [77] represented the sensor data as events and built an event processing system, where successive modules transformed the sensor events into behavior events and situation events. Another way of fusing inferences inside a multi-level model was demonstrated by Altun and Barshan [10], who used the activity inferences to recalibrate their dead reckoning method. At each activity change, the location is re-evaluated by mapping the possible action and their location, before recomputing backward the last trajectory and improving its accuracy.

Next, the fusion can also be applied after the context information is extracted, as in the SocialFusion system [21] where pattern discovery is applied after a first stage of inference on individual context dimensions. More precisely, user specific and group patterns are mined by computing the frequent sequences and frequent itemsets across modalities to provide recommendations. If the previous inference stage is able to produce a symbolic context, using for example a predefined ontology, then reasoning can be applied to fuse the data and produce more complex situation information. This principle was implemented in COSAR [244] where the semantic location was merged with the low level activities to produce high level activity. As for Bicocchi et al. [27], they proposed an approach for dealing with uncertainties of inferences while fusing them by computing their semantic proximity on the ConceptNet semantic network, using commonsense reasoning.

Finally, the fusion can be left to the last stage, namely while communicating the context information back to the user. For example by using the huge amount of possible display dimensions as in the BeWell [150] mobile application where the activity, social and health dimensions are represented by the behavior of various underwater animals, evolving on the smartphone wallpaper.

3 Collaborative sensing

3.1 Introduction

The strategies presented in Chapter 2.2.1 have been designed to optimize the use of each sensor present on the device. Single sensors' duty-cycles are computed to maximize the information gain, while reducing the sensing cost and taking the applications' needs into account. Some strategies [185, 217, 322] also take advantage of the various possibilities to get the same information from different sensors, for example the location can be obtained from the GPS or by triangulation with the visible Wi-Fi access points. But they are only focused on a single device or sensing node, determining when or where to turn them on or off.

In this chapter, we add another dimension to the problem, namely the sensing nodes are able to establish both long-range and short-range communications. Most, if not all, smartphones can connect to the Internet either through the GSM network or Wi-Fi access points and they have the capability to communicate to neighboring devices through Bluetooth. Those other devices can be external sensors, such as a wristband monitoring the heart rate or shoes counting the steps [273], but also other smartphones, thus creating a new kind of opportunistic sensor network.

Typically, sensor scheduling and data transfer has been widely studied in static sensor networks, but in contrast, we investigate sensing amongst multiple mobile users. The mobility of these devices can be turned into an advantage over static sensing nodes as they can cover much larger areas and comparatively reduce energy consumption as shown in the work of Wang et al. [294]. Similarly, the OpenSense project [2] applies this principle by mounting air quality monitoring sensors on buses and trams. Beside individual sensor optimization, a more global scheduling and placement problem is defined by Saukh et al. [251]. Their solution is to select the tram lines and the sampling points that provide the best coverage of a city. But these sensors' mobility is predefined with constraints. When we consider unconstrained mobility, for example using smartphones to gather air quality information [103], the problem becomes more challenging. We need to dynamically provide real-time sensing strategies that enables the collaboration between nearby devices, in an opportunistic way.

In terms of collaboration, mobile devices may reduce their individual expenses by sharing sensing information between the mobile nodes thereby reducing the total number of measurements needed. This collaboration can be done in two different ways:

- *Centralized Strategy* – There is a central server monitoring all nodes, and it is provided in real-time with their mobility information. Based on the locations, the centralized server can select which node should sense at which time to reduce redundancy. As it has a complete knowledge of the nodes and potentially also the values they are measuring, it can also perform more complex tasks such as faulty sensor detection or trust management [243].
- *Distributed Strategy* – Instead of using central-control, the distributed strategy focuses on exploiting proximity to share the nearby/similar measurement. Some mobile nodes may then get the measurements from their neighbors instead of sensing by themselves. Such a distributed strategy requires no central-control, and only short-range communication is allowed.

The contributions in this chapter are focused on the distributed version of collaborative sensing by studying the opportunities of mobile proximity to reduce the energy consumption of each sensing node. As an example, we consider reducing the number of needed GPS and Ozone level measurements by sharing them through Bluetooth. Indeed, with an average power consumption of less than 20mW [15], Bluetooth is a very good candidate for replacing the pluggable Ozone sensors [226] (more than 400mW according to our measurements) if another device can share it. Designing an energy-efficient sensing protocol for such distributed mobile sensing scenarios is non-trivial and we have the following challenges:

- *No-central control* – The first one is quite obvious as the opposite of the centralized strategy. There is no master server with the knowledge of real-time mobility. Even with a range of mobility prediction methods, it is still not guaranteed to have a complete view about who is where at what time. The distributed strategy should be able to deal with the uncertainty of mobile nodes' mobility in real-time.
- *Heterogeneous nodes* – For real-life opportunistic sensing, each mobile node can have a different energy-budget and sensing preference. For example, some nodes are super users with massive energy consumption compared to others; whilst some care more about own-sensing quality rather than approximation of the measurement from nearby nodes. An optimal sensing strategy should be able to satisfy various types of node requirements.
- *NP-Hard* – Like many other works studying the problem of sensing protocol [143, 309], we face a NP-hard problem to get an optimal sensing. Even in many cases, near-optimal solutions are computationally expensive. Real-life applications should take these into account.

To address these challenges, in this chapter, we design a protocol allowing the nearby devices to synchronize and reduce duplicate measurements. In addition to empirical studies using real-life rich datasets (the Nokia Data Collection Campaign dataset [135] and the Paleo Festival dataset from Naini et al. [198]), we also compute an optimal upper-bound on the gain of collaboration.

After reviewing the previous work on collaborative sensing, Section 3.3 presents the formal definitions and the instance of the problem studied in this paper. A new strategy is incrementally built in Section 3.4, detailing the construction steps. In Section 3.5, we formulate the scheduling problem as an optimization under constraint that we use in Section 3.6 for comparison with the performances of our strategy on a real world dataset.

3.2 Related Work

3.2.1 Centralized coordination

Two distinct approaches have been studied for coordinating mobile sensing nodes with unconstrained mobility, namely centralized and distributed. The first one allows to reach better results in terms of coverage and minimal redundancy, but it needs to track continuously the nodes and for them to be able to communicate at any moment. For example, Ngai and Xiong [201] propose a strategy where the central server keeps a constantly updated view of the quality of the knowledge in each region and asks mobile nodes for more measurements if this quality is too low or if an event is suspected. Such centralized information can also be used for learning mobility patterns, as in Bigwoodt et al. [28]. They used the pattern in history to predict the encounters of a mobile node, allowing a better coordination.

Riahi et al. [243] proposed an efficient heuristics to select the sensing nodes based on the mix of queries received by the central server, taking into account different aspects such as data reliability, users privacy and resources constraints.

But central coordination also has a lot of drawbacks and challenges to solve, such as long-range (and costly) communication, privacy and big data processing, therefore we preferred using an ad-hoc peer-to-peer coordination between the nodes.

In this scheme, the mobile nodes only exchange information locally using short range communication. This information enables them to select the best sensing policy.

3.2.2 Opportunistic collaboration

Information exchange happens opportunistically as soon as two nodes are within communication range.

In their work, Wang et al. [291] studied uncoordinated mobile sensors that can exchanges their knowledge about the field they are sensing if they meet each other. The protocol is based on the fact that a node will ask the neighboring nodes for information if it doesn't have enough confidence in its own measurements. A simulation was run to evaluate the performances on a mobility model.

In RAPS [209], a system for tracking the location of a mobile phone, the authors included the option for sharing GPS location through Bluetooth. They also made some extensive energy measurements for Bluetooth and GPS among other sensors. In their protocol, the communication is initiated when a node has a fresh GPS position and it sends it to all neighbors, updating their location according to their uncertainty. The results showed more than a 10% energy saving using this process on their dataset, but very often the client failed to connect as several other clients were trying to initiate the connection all at the same time.

3.2.3 Ad-hoc coordination

But even in ad-hoc fashion, coordination is needed to reduce duplicated measurements and overall energy-consumption.

Weinschrott et al. [298] presented the notion of virtual sensors which represents a static view of all the mobile sensors. When in a certain region, mobile sensors provide data to the corresponding virtual sensor, responsible for this region. Using this paradigm, the authors presented a centralized and a distributed version of their coordinating algorithm, both relying on route prediction. This algorithm optimizes the selection of mobile nodes and the scheduling of measurement to minimize overlapping and reduce the overall energy consumption. It also supports re-assignment of the task if the sensor cannot fulfill the request. The Aquiba protocol [277], developed by Thepvilojanapong et al., sets the sensing rate of each sensor according to the need of the query received from the central server and to the number of peers that are visible in the neighborhood. A variation of this protocol supports the selection of representatives that will actually perform the sensing task. Their evaluation is based on simulations only on mobility models.

3.2.4 Fairness

When optimizing for a global objective function, there is a risk that a few nodes actually take all the load and therefore one should also factor fairness into the optimization. In their offline algorithm for solving optimally the centralized scheduling of sensors problem, Sheng and Tang [263] proposed a min-max approach. On their side, Tang and Zhang [276] presented a centralized algorithm that is able to compute efficiently the optimal solution to the scheduling problem, defined using the virtual sensors abstraction, while seeking a min-max fair sensing schedule.

Unfortunately this issue was to our knowledge never taken into account for online opportunistic coordination.

Our approach differs from these previous works in the fact that we implemented an ad-hoc distributed strategy for coordination of the measurements themselves and we also provide an optimal offline one for comparison. The evaluation, by simulating over two real-world datasets instead of a mobility model reflects better the issues that may arise when going for a real implementation.

3.3 Problem Formulation

In this section, we first define the general sensing problem, using a similar notation to Wang et al. [291]. Then, the specific instance of this problem that we develop further and evaluate, is presented.

3.3.1 General definitions

We consider a set of N mobile nodes $\{n_1..n_i..n_N\}$, having each a limited budget B_i and as a task to take measurements $m_s^n = [\rho_s^n, t_s^n, \mu_s^n]$ with a sensor s of a field F with noise: $\mu_s^n = F(\rho_s^n, t_s^n) + \omega$. We define ρ as the location, t as the time and μ as the value of the measurement m. The goal is to achieve a sufficient coverage for modeling with precision a certain spacio-temporal field $\mathscr{M}^n(\rho, t) = F(\rho, t) + \varepsilon$. The definition of the specific problem or field to sense contains information on the sensing coverage requirement and the way measurements are merged to build the model. Each measurement has a cost depending of the sensor used $c(m_s^n)$.

The collaboration between users happens when one user sends her current measurement to another user. Sending and receiving operations also have a cost $c(\text{send}_s^n)$ and $c(\text{receive}_s^n)$

For our evaluation we define the total error as

$$\mathscr{E} = \sum_{i=1}^{N} \sum_{t \in T} |F(\rho, t) - \mathscr{M}^{n_i}(\rho, t)|, \tag{3.1}$$

where ρ is the location of the user at time t. In practice, we use the last known location $\rho_s^{n_i}$ taken from the last measurement before time t.

To evaluate how evenly shared is the load amongst the sensing nodes, we also define an approximation of the fairness. For this purpose, we compute the average gain ratio of each node, expressed by the number of received measurements, divided by the number of sent or receives measurements:

$$\mathscr{G}_s^n = \frac{|\text{receive}_s^{n_i}|}{|\text{receive}_s^{n_i}| + |\text{send}_s^{n_i}|} \tag{3.2}$$

Then Jain's Fairness Index [121] is used to assess the distribution of the gains among the nodes:

$$\mathscr{J}_s = \frac{\left(\sum_{i=1}^{N} \mathscr{G}_s^{n_i}\right)^2}{N \cdot \sum_{i=1}^{N} \mathscr{G}_s^{n_i 2}} \tag{3.3}$$

This is an approximation because it assumes that on average every node will encounter the same number of other nodes and thus a fairness of 1 is not always possible. However, for a given experiment with predefined paths and encounters, it can still serve for easily comparing different protocols' fairness.

The Table 3.1 summarizes all the symbols defined so far.

symbol	definition
N	number of mobile nodes
n_i	a mobile node
B_i	the budget of node i
m_s^n	a measurement with sensor s at node n
ρ	the location of the corresponding measurement
t	the time of the corresponding measurement
μ	the value of the corresponding measurement
F	the field to sense
ω	the noise of measurements
$\mathscr{M}^n$	the model of the field at node n
ε	the error of the model
$c(x)$	the cost of the operation x
$\mathscr{E}$	the total error
$\mathscr{J}_s$	the fairness

Table 3.1: Summary of the notation

3.3.2 Specific instance

As we have two variables to minimize, namely the cost and the error, we will consider a generic optimization problem, including both. To this purpose, we define δ as the cost of the error, quantifying the relative importance of modeling error with regard to the sensing cost. Therefore our cost function, also called *total cost*, will be as follow:

$$\mathscr{C} = \sum_{i=1}^{N} \sum_{t \in T} c(m_s^n) + \delta \cdot \mathscr{E}. \tag{3.4}$$

Without limiting the generality of the foregoing, we will use an abstract field F sensed by the nodes with an uncertainty growing linearly with time. Meaning that the error of the last measurement is directly proportional to its age. A real-world example could be measuring the location of somebody moving at constant speed. In a discrete time, the total error between samples is growing as the well-known sequence of triangular numbers $(1,3,6,10,15,...)$, but to use it in a continuous time as we will be doing in some further computation, we express it as a function of time between samples Δt:

$$\mathscr{E}(\Delta t) = \frac{\Delta t \cdot (\Delta t + 1)}{2}. \tag{3.5}$$

Then the total error is the sum of all these triangles as shown on Figure 3.1.

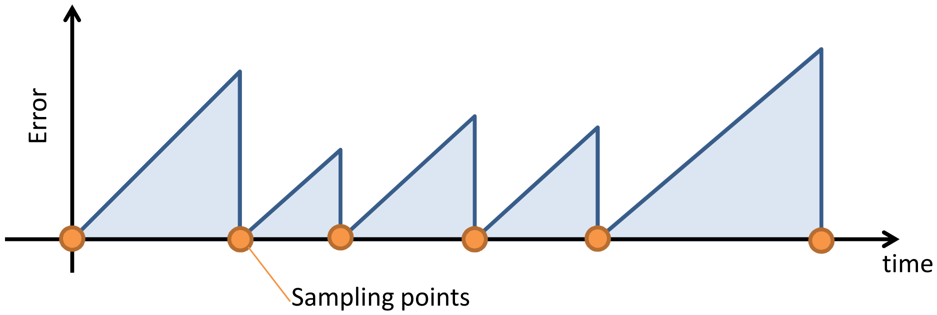


Figure 3.1: The error is growing linearly between the measurements and the total error is the total area of the triangles.

For the simulation and evaluation we will restrict ourselves to a concrete problem, but of course the algorithms presented here can easily be adapted to other collaborative sensing scenarios.

The mobile nodes are making measurements with a certain sensor, m^n_{Sensor} and can communicate using Bluetooth with the neighboring nodes. The field F to sense is the same as for the theoretic part and evolves with time. The model $\mathscr{M}^n$ simply returns the last sensed value. As the datasets we use for the evaluation have a granularity of 1 minute, we will also use this step-size for our simulation, which makes 1440 steps per day. A budget B_i is computed at the beginning of the experience for each node i and represents the number of measurements that would minimize the equation 3.4 for a given δ, see next section for the formula. Then, we will try to further minimize the costs by using as much collaboration as possible . According to our measurements and the ones reported by Paek et al. [209], certain sensors such as GPS or Ozone sensors have a much higher energy consumption than Bluetooth, thus making collaboration desirable. Moreover, the new version of Bluetooth low energy (BLE) suggests even more incentives to use short range communication instead of using the GPS or other costly sensors.

3.4 Opportunistic Strategy

This section shows the sensing strategy developed to maximize the collaboration, thus minimizing the total power consumption. We will build our strategy incrementally, adding new behavior at each step.

3.4.1 Base-strategy

We start with the very simple case where the mobile nodes do not know when they will meet each other, but they are aware of the duration of the experiment and therefore can plan when measurements have to be taken. Moreover, if all measurements are equally useful for the model, the optimal distribution is to sense *uniformly at regular intervals*. In our simulation, we defined the error as being linear in time, therefore we will sample uniformly. Given the ratio δ and an experiment of one day (1440 steps) it is possible to optimize beforehand the number of measurements using the error definition from Equation 3.5. This will be called the initial budget, and for a node i, it is given by:

$$B_i = argmin_b \left(b + \delta \cdot \frac{1440/b \cdot (1440/b + 1)}{2} \cdot b \right) \tag{3.6}$$

Figure 3.2, shows the initial Budget for different values of δ.

This first scenario is illustrated on Figure 3.3-A, the user 1 having a larger δ and thus a higher sampling rate.

At anytime, the nodes are broadcasting their *latest measurement* and at each time step they scan the neighborhood and collect information about the other devices within communication range. With this communication we can see on Figure 3.3-B that during encounters the nodes have more measurements available, reducing their sensing error, but a lot of duplicated measurements appear. A measurement is called a duplicate if it was performed in the same step as another measurement and the two nodes where within communication range and could have shared their measurements.

To avoid the communication collision while using Bluetooth, we suggest using the discovery beacon for piggybacking the broadcast information in the *extended inquiry response* packet's payload (see the Bluetooth LE specifications [267] for more details). An example of a real implementation of such a connectionless scheme was realized by Bin and Kim [29].

3.4.2 Synchronization

Collaboration so far is limited to passively listening to other nodes and no specific action is taken for avoiding *duplicate measurements*. In order to achieve that, the nodes would need to

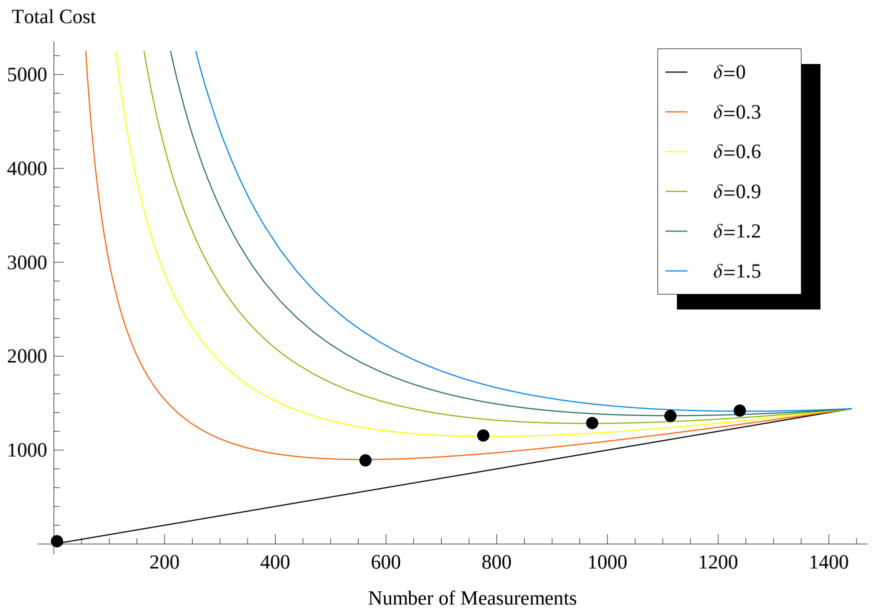


Figure 3.2: Optimal Number of samples for different values of δ for an experiment of one day, without collaboration.

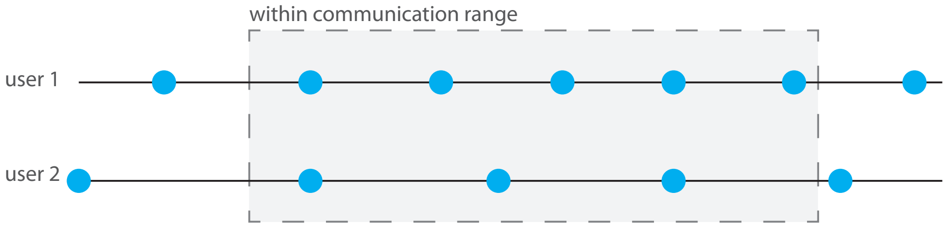
decide which one is in charge of sensing.

To deterministically decide, the nodes also exchange some other information, like their remaining budget, age of last measurement and δ, and the node with the largest remaining budget will adapt its next sensing time to the other node's need. More formally, it defines its next sensing time as the *minimum* between its own next sensing time and the planned next sensing time of the other node, based on the sampling rate computed from Formula 3.6.

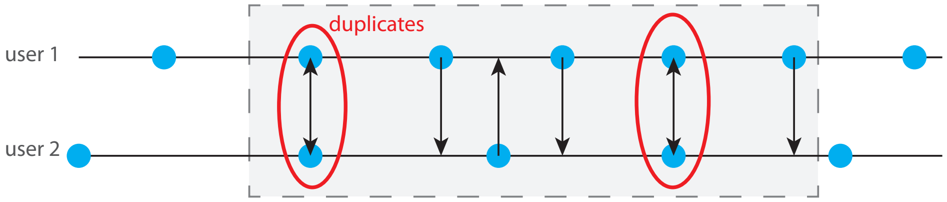
To illustrate this, we show on Figure 3.3-C that user 1 (with a larger δ and thus also a larger budget) will continue taking measurements while user 2 will just rely on the ones he receives. After some time, the budget of user 2 is at the same level as user 1, thanks to the saved measurements and therefore user 2 is selected to take a sample, while user 1 in his turn relies on it. This selection process ensures that, with long encounters, the two nodes will contribute parts of their budgets that are proportional to their actual needs, and ensure some kind of fairness.

3.4.3 Time-window

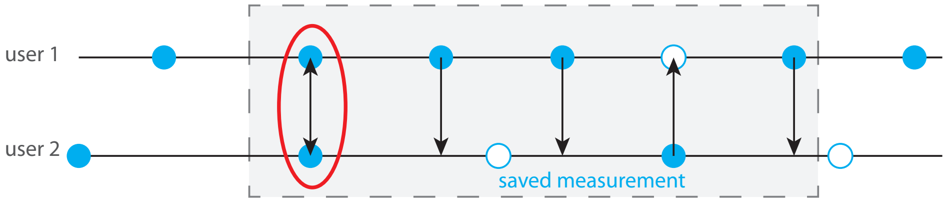
Such collaboration works well when the nodes spend a lot of time together, but it may lead to bad scheduling if the encounters are very short. For example a node meets another one with a



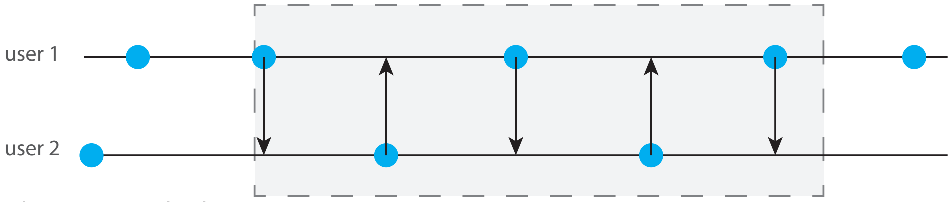
A. No collaboration, each node sense at its own rate.



B. Nodes can exchange measurements when in range, but they don't change their sensing rate.



C. Nodes can exchange measurements when in range and choose their sensing time to benefit more from collaboration.



D. The optimal scheduling

Figure 3.3: Timelines showing the samples taken and exchanged by two users.

larger remaining budget and decides that it will rely on the other node to take a measurement at the next step instead of scheduling a measurement itself, but then the other node leaves and our first node will have its error increased, by not taking a new measurement.

Therefore, to determine if two nodes can start collaborating, we need to know with a certain probability that they will still be in range at the following step. The distribution of the length of encounters shows a lot of very short ones (see more details in Section 3.6). Those are either due to actual short encounters, people passing by or to missing beacons because of the short communication range. Thus we define a *time window* of size W and a ratio R. If the number of encounters per step is greater than R within the last W steps, then the node will assume that there is enough probability to establish a contact at the next step and will try to synchronize with the other nodes.

Of course this strategy is also applicable with more than two devices, the one with the largest budget taking the measurements. Algorithm 1 summarizes the final strategy using the parameters previously defined.

3.4.4 Parameters selection

The above presented strategy can be tuned using a few parameters. These can be set globally for a given application or individually for each device, according to users' preferences.

The first and most important parameter is δ; it represents the trade-off between accuracy and energy saving. More formally it is the cost of an error of one unit. Therefore the effect of this parameter is strongly linked to the choice of the error function of the model. In the case of a linearly growing error as we chose and from the Figure 3.2, we can see that a value greater than 0.5 would lead to sample more than once every two steps and a δ of 0 implies that no measurements are needed. A typical value that we used for the evaluation is 0.2, giving a sampling period of approximatively 4 steps.

The two other parameters W and R need some more knowledge about the distribution of encounters length and interval. Typically if the duration between encounters is short, for example when the devices are around the limit of communication range, a larger W can help to keep the devices collaborating. Conversely, if encounters are separated by large amount of time you may want to reduce the W to react faster at the next encounter.

The ratio R when set to 100% will restrict collaboration to encounters that last at least W+1 steps. This can be used if the distribution of encounters contains too many short encounters that would trigger the collaboration mechanism, but not last long enough to be of benefit to the nodes. Smaller values of R could be used to overcome gaps in encounters, but would also increase the delay for ending collaboration at the end of an encounter.

3.5 Theoretical Optimum

In this section, we look at how an offline algorithm would perform in scheduling the sensing nodes to minimize the objective function given by Formula 3.4, as shown on Figure 3.3-D. We use the field $\mathscr{F}$ and model $\mathscr{M}$ defined in Section 3.3.2 and formulate an optimization problem under constraints.

According to the definition of our problem, each node will have to satisfy the tradeoff between modeling error and measurement cost. In the offline case, the nodes know exactly the starting and ending time of the experiment and can therefore plan all their measurements.

In the simple case where the nodes never meet and therefore cannot exchange any information, the optimal solution we gave in Section 3.4.1 can be used and the node would sample uniformly over time. In the opposite case where for example two nodes spend all the time together, the

Algorithm 1: collaborative sensing strategy

```
/* initialization */
age ← infinity                                   //how old is the last value
measurement ← null                               //last value
lastNode ← null                                  //who was the last encounter
b ← $B_i$                                        //budget from Equation 3.6
timeWindow ← [0... 0]                            //FIFO queue of size $W$
baseRate ← T / b
/* main loop */
while remainingTime > 0 do
    temp ← currentMeasurements                   //get sensor data
    if temp then
        age ← 0                                  //update last value
        measurement ← temp
    setBroadcastingInfo(measurement,age,b)       //to be sent
    others ← received info                       //get other nodes' data
    timeWindow.push(|others|)
    timeWindow.pop()
    if others is not empty then
        lastSeen ← 0
        lastNode ← max(others,b)                 //keep the node with max budget
    if any (other.age < age) in others then
        (measurement,age) ← (other.measurement,other.age)   //if newer, update our data
    if lastNode and sum(timeWindow) > $R \cdot W$ then
        if b >= lastNode.b then
            sampling ← (age >= baseRate)
            or (lastNode.age >= lastNode.baseRate)
        else
            sampling ← (age >= baseRate)
    else
        sampling ← (age >= baseRate)
    if sampling then
        querySensors()
        b -= 1
    age += 1
```

optimal solution is for them to alternatively sample and send their result to the other node, reducing this way their own expense in terms of measurements. The reduction grows then proportionally with the number of nodes.

But if we consider a more complex schedule of encounters between the nodes, the solution is no longer trivial. Indeed, if two nodes have overall a different total time spent with other nodes, they will be able to save different amount of sample-budget and distribute it over the time they are alone. But if two nodes that have a different overall target sampling rate meet, they should agree on a collaborative sampling rate that can be anything between their respective targets. To compute formally the total amount each node will use from its budget for each period, we first need to define some variables.

3.5.1 Definitions

For a certain node or group of nodes, an encounter is a period of time during which the environment of the node is stable, meaning that it has the same neighbors. As our dataset is sparse enough, and to simplify the computation, we will consider complete graphs for modeling the encounters on this dataset. In other words, if node A sees node B then node B sees node A and the same for larger groups by applying transitivity. This is a reasonable assumption as the detection distance is pretty small (a few meters) compared to the field of experiment (tens of kilometers) and so there is a very low chance of creating a long chain[1].

Considering that, we build the matrix E of all encounters and the vector L for the durations, defined as:

$$E_{j,i} = \begin{cases} 1 & \text{if } n_i \text{ is in encounter } j \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

$$L_j = \text{duration of encounter } j \tag{3.8}$$

Further we define the matrix A, which is similar to E but contains the contribution (in terms of number of measurements shared) of node i during the encounter j. Finally let O_e and O_n be vectors filled with 1's with a length of the number of encounters and respectively the number of nodes. The total error from Formula 3.1 using the continuous definition from Formula 3.5

---

[1] This was tested on our dataset, but is not always true

can now be expressed with these matrices as

$$\mathscr{E} = \left(\frac{L}{2} * \left(\frac{L}{A.O_n} + O_e\right)\right)^{\top} .E.O_n \tag{3.9}$$

and the total cost in Formula 3.1 is written

$$\mathscr{C} = O_e^{\top}.A.O_n + \delta \cdot \left(\frac{L}{2} * \left(\frac{L}{A.O_n} + O_e\right)\right)^{\top} .E.O_n \tag{3.10}$$

where the divisions and the multiplication $(*)$ are done element-wise on the vectors.

This defines a non-linear minimization under constraints problem:

$$\min \mathscr{C}(A) \tag{3.11}$$

subject to

$$\{A_{j,i}\} \begin{cases} = 0 & \text{if } E_{j,i} = 0 \\ >= 0 & \text{otherwise} \end{cases} \tag{3.12}$$

As devices usually have a limited total energy available, for example when using batteries, it also makes sense to rewrite the optimization by setting the budget to a fixed value B and minimizing only the error.

$$\min \mathscr{E}(A) \tag{3.13}$$

subject to

$$(O_e^{\top}.A)^{\top} - B = 0 \tag{3.14}$$

$$\{A_{j,i}\} \begin{cases} = 0 & \text{if } E_{j,i} = 0 \\ >= 0 & \text{otherwise} \end{cases} \tag{3.15}$$

Both these problems are considered for our evaluation.

3.5.2 Competitive analysis

In this section, we evaluate the performance of our algorithm when an opponent with the goal of maximizing the error could decide the next input, i.e. the encounters of the nodes. The competitive analysis shows the ratio in a worst case scenario between the considered algorithm and an optimal (usually offline) one, which would have complete knowledge of future encounters. For this analysis, all the sensing nodes will run our protocol. We do not consider misbehaving nodes here.

First we consider the case when the nodes meet less than the threshold R (as defined in Section 3.4.3) in any possible window of length W, both nodes will take measurements at a regular rate of B/T, where T is the total time of the experiment and B the initial budget. The total error is then $(n/2)\cdot T\cdot(T/B+1)$ for n nodes. The best that can be achieved by an offline algorithm when a maximum (still below the threshold) of encounters are properly distributed is to alternatively sample and share the resulting measurements. If R is smaller than B/T, the remaining measurement are also distributed evenly for both nodes at the same time, the sampling rate being doubled compared to the shared part. The resulting error is:

$$\frac{n}{2}\cdot T\cdot\left(\frac{T}{\left(B+(n-1)\cdot min\left(B,\frac{R\cdot T}{n}\right)\right)}+1\right). \tag{3.16}$$

The $min(B,(R\cdot T)/n)$ represents the number of measurements that can be shared. For this case, the ratio of online algorithm to the offline algorithm is upper-bounded by n when n nodes are meeting.

Then we consider the case when the nodes have enough encounters, the W and R parameters playing an important role. Indeed as long as the number of encounters in the last W steps is at least $R\cdot W$, a node with a lower budget will rely on the last encountered node to provide it with the measurements. Therefore the strategy of the opponent will be to make the node with the lowest budget wait for a measurement from another node for as long as possible, but having enough encounters to be above the threshold. This scenario is shown on Figure 3.4, where R_T is the remaining time and R_B is the remaining budget.

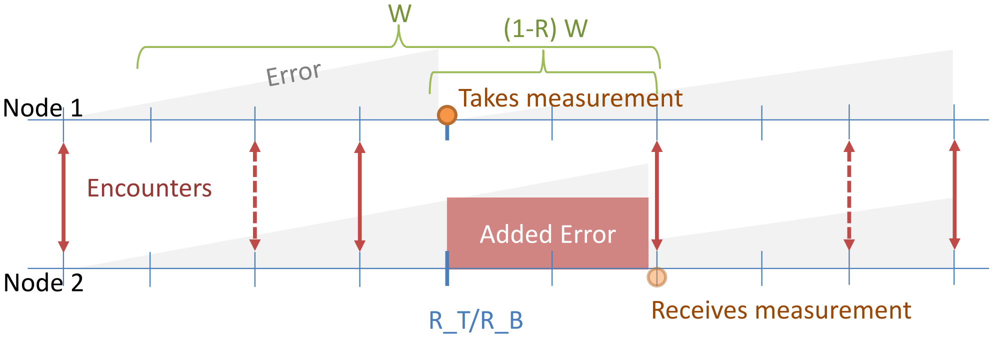


Figure 3.4: Worst scenario when two nodes try to collaborate.

This waiting time is defined by $(1-R)\cdot W$ and the added error compared to the case where the sharing of the measurement would have happened immediately is marked in red on the Figure. Node 2, with a lower budget was planning to take a measurement at time (R_T/R_B), but as it met Node 1 right before, with a larger budget, it waited (at most $(1-R)\cdot W$ steps) to receive the measurement from it. This can then be repeated with Node 1 and 2 inverted, as Node 1 may have decreased its budget now. To match the necessary number of encounters to reach the ratio R in the window, some encounters can be added before the measurement time, like the dashed arrows on the Figure. In this scenario, by shifting the sampling times slightly earlier, the optimal algorithm would be able to reach at least the same error as in the previous case, the "added error" being the lower bound on the difference between both algorithms. From the Figure 3.4 we can see that we have at most a proportional overhead of $(n-1)(1-R)\cdot T$ for an experience of length T and thus a ratio bounded by $1+2\cdot(1-R)$, which is always smaller than the upper bound found in the first case, except for two nodes.

To summarize our competitive analysis, we have shown that the competitive ratio although determined by the parameters B and R can be upper-bounded by n, or $1+2\cdot(1-R)$ if $n=2$. The worst case scenario being when the nodes are always together, but have just not enough encounters for triggering the collaboration. A good choice of parameters can help to guarantee tighter bounds, but as we will see on the evaluation on the real-world dataset, the results are already far below this upper-bound.

3.6 Evaluation

3.6.1 Dataset descriptions

Nokia dataset

First, we investigated the large Nokia Data Collection Campaign dataset [135] to asses the scalability of our algorithm and the potential impact of collaboration. From this dataset, we selected one year of data and took the Bluetooth scanning tables. Over the 166 users, the average time spent with another user in the neighborhood according to the Bluetooth scans is 8%, with one third of the users over 10% as shown on the Figure 3.5. For this evaluation, we only consider the encounters within the community of 166 users, representing 22% of all Bluetooth scans. This would be the typical case if only a few people with some social connections[2] were running a continuous sensing software and using our protocol for collaborative sensing. Latter results can of course be improved a lot if the protocol is deployed at larger scales or in denser communities. The distribution of the length of the encounters and intervals between them are shown in Figures 3.6 and 3.7.

As the data was recorded on mobile phones, a smart scheduling of the sensors was implemented to avoid draining the battery too fast. But this non-uniform measurement rate,

---

[2] Recruitment of participants to the Nokia collection campaign was done by asking group of friends, co-worker, etc using a snow-ball effect.

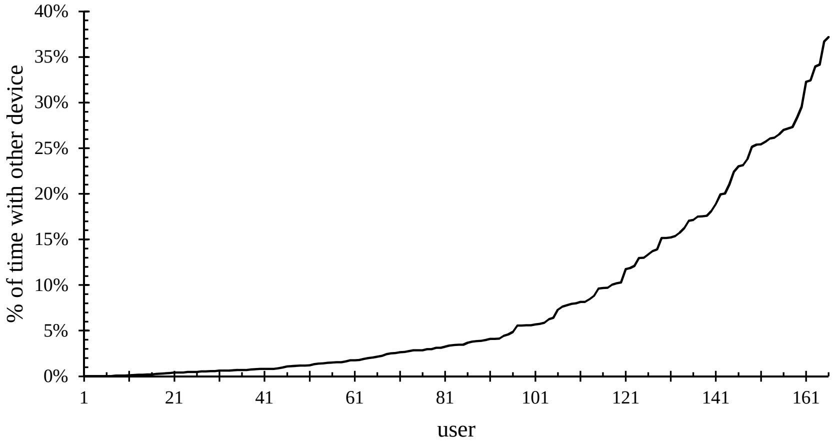


Figure 3.5: Percentage of time with another of the 166 user in the BT vicinity

responsible for the irregularities on Figure 3.6 also slightly affected our simulation by missing surrounding BT devices every now and then. To avoid that, the W parameter was set to 4 and R to 75%, thus flattening a bit the small intervals.

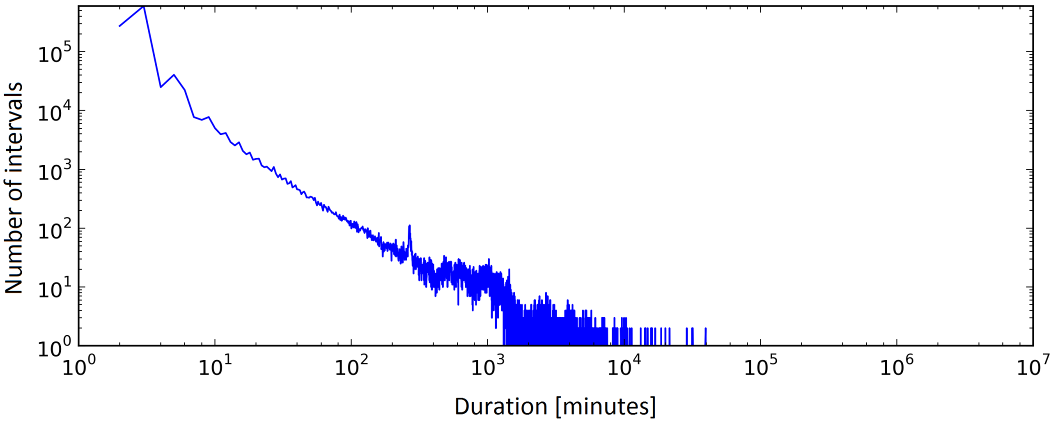


Figure 3.6: Distribution of the intervals between two encounters.

The simulation of our strategy was run over this dataset with the two objective functions described in Section 3.5, namely minimizing both the error and the sensing cost and minimizing the error with a fixed budget. For the first one, our strategy is directly used, but for the second one, small modifications are needed on Algorithm 1, namely on line 6 the defined budget is taken instead of computing it and on line 36, sensing is only performed if the remaining budget is greater than 0.

During the simulations, the input matrices needed for Equations 3.9 and 3.10 were also generated and in a second time given to Octave[3] for solving the minimization problem using

---

[3] http://www.gnu.org/software/octave/index.html

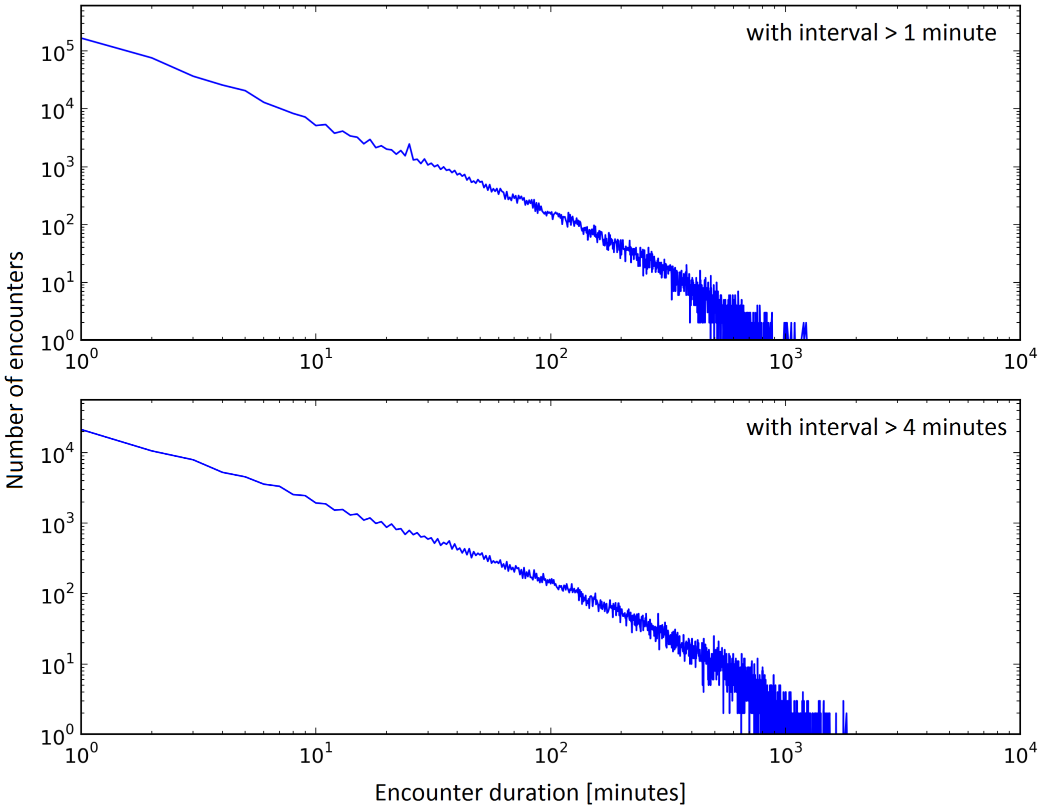


Figure 3.7: Distribution of the duration of the encounters. Removing the intervals smaller than 4 minutes reduces by one order of magnitude the total number of encounters.

sequential quadratic programming. The computation was done on continuous values as Octave does not provide an integer version for this kind of solver. As the budget was reinitialized every day (as if the users were charging their devices every night), the problem was also cut into 365 independent subproblems and it took around 1 day on a 12 cores @ 2.3Ghz server to compute the optimal values.

Paléo dataset

In this section we show in more detail the performances of our strategy on a dataset collected during an experiment run by Naini et al. [198] during the Paléo Festival[4] with 10 users over 10 hours. The scanning software was recording the Bluetooth physical addresses every minute and GPS was turned on when the user was moving. The motion of the user was detected using the accelerometer and GPS location when available, similar to the scheduling algorithm presented in Section 6.4.2.

---

[4] http://www.paleo.ch/

The timeline, in Figure 3.8, shows when GPS location was available and other devices' bluetooth were detected. The green and yellow slots both indicate that the device had a GPS fix, but for green no neighboring devices were detected. The red slots indicate when the device had no GPS fix and could have received its location from another nearby device if it had sent it. Finally the blue slots indicate when other devices are detected, but no one has a GPS fix. In total, 411 times a user's phone detected another user's phone. As users 3 and 7 were having trouble with the Bluetooth detection, they were not considered in the later evaluation. From the timeline, we can already see that users 2 and 10 (resp. 4 and 5) were together during a significant part of the evening and therefore have a large potential for collaboration and energy saving.

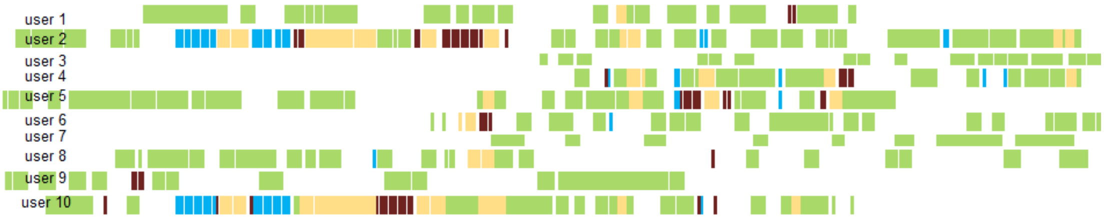


Figure 3.8: Timeline from the Paléo dataset showing when GPS (Green and Yellow) and Bluetooth encounters (Red, Blue and Yellow) are available. Red means that a GPS location is included in the BT beacon, whereas Blue is when there is no location. (for B&W printing, from darker to lighter: red, blue, green, yellow)

The evaluation was done by emulating our strategy over this dataset and comparing it to two baselines, one using only the base strategy and collaborating when possible and another without the collaboration. The parameters W and R were directly optimized using grid search during the experiment. But looking at the histogram of the interval and duration of encounters, see Figure 3.9, we can already tell that the encounters were rather short, and may have had small interruption of a few minutes.

3.6.2 Results

Nokia dataset

When optimizing for both error and energy consumption, the optimal offline computation is able to reduce the objective function by 5% compared to our baseline without collaboration. With a δ of 0.8 it represents a reduction of cost of 10% and 3.25% of energy. Interestingly the distribution among the users is not uniform, as shown on Figure 3.10, and while half of the users do not seem to be able to benefit from collaboration, some of them are able to save more than 10% of energy. This can be explained by the distribution of the encounters themselves, from Figure 3.5.

At $\delta = 0.5$, our online strategy is able to reach 25% of the optimal strategy compared to the baseline. As we will see later, the online algorithm is always penalized by short encounters

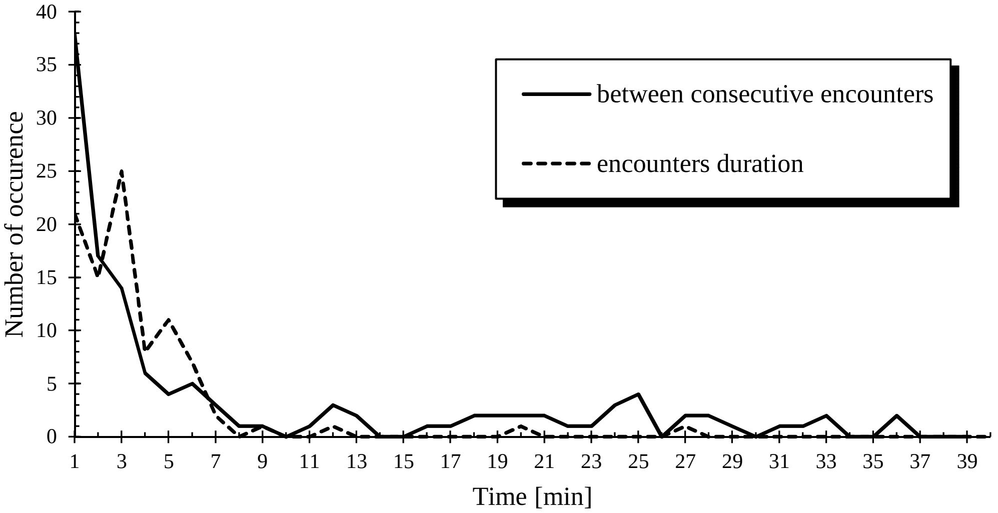


Figure 3.9: Histogram of the time between two consecutive Bluetooth encounters and encounters duration over all the users.

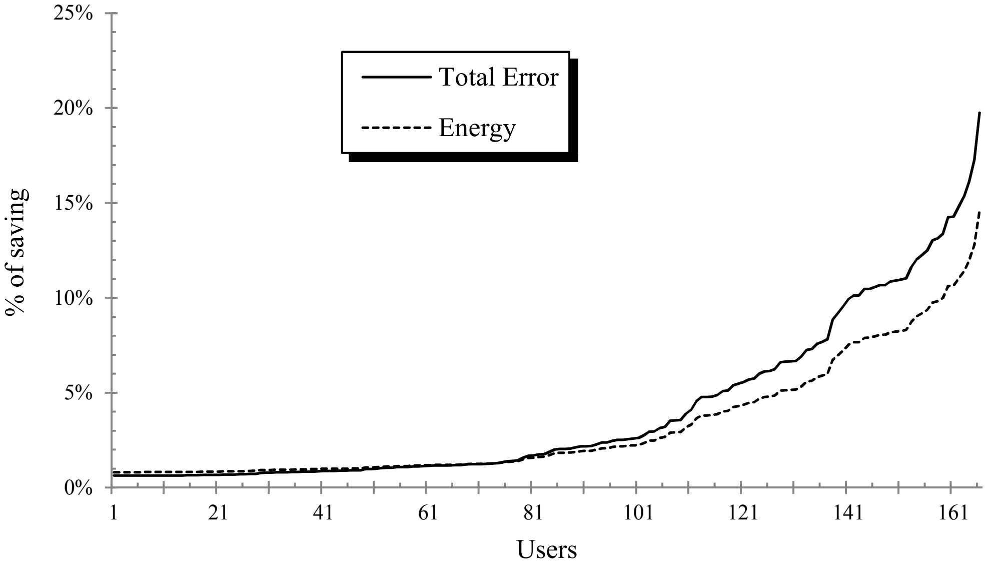


Figure 3.10: Distribution of energy and total cost saving for all users using the optimal solution.

which often generate duplicate measurements like the one we can see on Figure 3.3-C.

When fixing the budget and minimizing only the error, we get similar results. As shown on Figure 3.11, the optimal offline error reduction compared to not collaborating is varying between 8% and 23%, depending on the initial budget. With a low budget, the value of each measurement is higher and a good scheduling and sharing scheme can help a lot. Our online strategy is not able to performs as well because of the short length of most encounters compared to the base sampling rate (roughly one sample per hour for a budget of 30). Indeed it would need to predict the next encounters for the next hour. In the middle, with a base sampling period between 2 and 5 minutes the online algorithm is around 50% as good as the optimal offline one. The high values on the high budget side for the offline algorithm are due to the aliasing effect of computing the optimal schedule with continuous values. Indeed, with a budget of 1200 and an experiment of 1440 steps continuous divisions are much more regular than discrete ones.

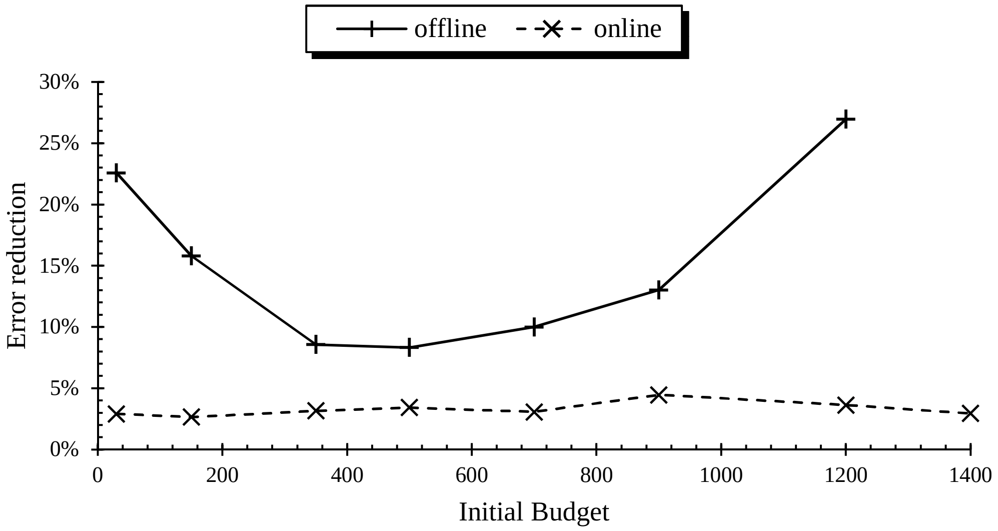


Figure 3.11: Error reduction induced by collaboration and measurement exchanges, showing the offline optimal solution and our online algorithm.

From Figure 3.12 we can see that introducing the concept of synchronization between nodes and keeping track of previous encounters can divide by three the number of duplicates compared to the baseline illustrated on Figure 3.3-B.

Based on these encouraging results, we analyzed in more detail the behavior of our strategy on the smaller Paléo Festival dataset.

Paleo dataset

First our strategy was simulated with different values of W and R, performing a grid search to find the optimal combination. As $W \cdot R$ was anyway rounded to an integer, we directly chose the values of R that gave integers. On the Figure 3.13 we can see that a good choice of values are $W = 6$ with $R = 5/6$.

Using these parameters we compared our strategy to the baseline with and without information exchange. As shown on Table 3.2, by transmitting their last measurements, the nodes were able to reduce the total cost of the baseline by 3%, but with a lot of duplicated measurements. Our strategy was able to improve the total cost by 3 percent more and at the same time divided the duplicate measurements by 3 (from 505 to 141) and reduced the budget used by 1/3 . The

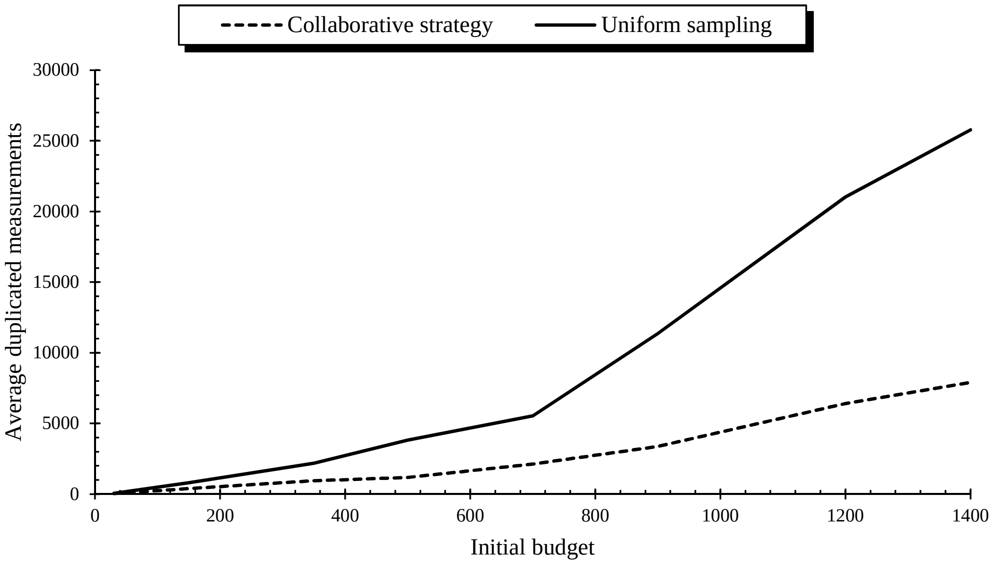


Figure 3.12: Average number of duplicated measurements per node over one year for different budgets, the optimal strategy always being at 0.

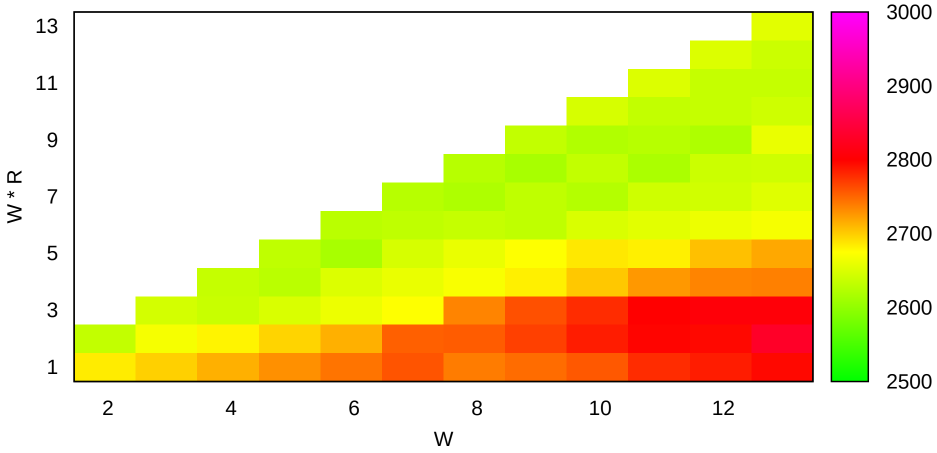


Figure 3.13: Heatmap of the total cost with $\delta = 0.2$ according to the R and W parameters

fairness was also improved by distributing more evenly the sensing costs over the nodes. The experiments using fixed budget showed a similar behavior.

Strategy	Budget used	Total Error	Total cost	Duplicates	Fairness
Baseline	1640	5720	2784	-	-
Baseline+communication	1640	5354	2711	505	0.39
Collaborative	1072	7723	2617	141	0.72

Table 3.2: Performance comparison between the baseline (regularly sampling) with and without communication, and our collaborative strategy.

3.7 Conclusion

In this chapter, we explored a new strategy to implement energy-efficient collaborative sensing in a completely ad-hoc manner, among multiple agents with uncontrolled mobility. Its evaluation showed the reduction of energy budget of smartphone users and at the same time a satisfactory level of sensing accuracy. We also expressed the offline optimal solution as a non-linear optimization problem under constraints and computed its numerical approximation. With sampling periods between 2 and 5 minutes, our online strategy showed experimentally an cost saving ratio of 2 compared to the offline optimum. The evaluation was then performed on two real-life datasets.

4 Optimal Sampling

4.1 Introduction

As we have seen in the previous chapter, mobility is key. Not only it does allow the sensing devices to collaborate but they can also cover much larger areas than static sensors. This chapter introduces optimizations for such mobile sensing devices, by determining the sampling policy in the case of a mobile environmental sensor device.

Traditional wireless sensor networks (WSN) based environmental monitoring systems (like SensorScope [117] and MacroScope [279]) typically deploy sensors at some pre-selected positions, monitor these fixed sensors continuously, and analyze their measurements. These traditional static WSN with fixed sensors have however a couple of obvious limitations, e.g. (a) the system is inflexible for monitoring location-varying environment; (b) it is expensive to deploy and maintain a large set of static sensors; (c) for a very large area needing to be monitored, it is impossible to get enough static sensors to cover the complete area.

To overcome such limitations, researchers recently start to build WSN with mobile sensors. There are a lot of emerging mobile sensing platforms, e.g.,: (a) the OpenSense project in Switzerland builds sensors and puts them on public transport, like buses and trams [2]; (b) the floating sensor network project at UC Berkeley builds a water monitoring system using drift sensors to analyze water contamination [278]; (c) mobile phones are used to establish a community seismic network to detect earthquakes and rare events [82]. The mobility of sensors today stimulate substantial research and poses technical challenges for the communication and information systems infrastructure, to scale up from isolated well controlled systems to an open and scalable infrastructure.

In the OpenSense project, the use case of this chapter, we deployed environmental sensors on moving buses to monitor air quality (using sensors to measure CO_2, CO, NO_2, etc.). The mobility of the sensing node implies that the sensing policy has to be adapted to the varying environment, as a single time series model would not be able to capture accurately all the different locations visited. Defining the time or the place to take measurements is similar to the more traditional problem of sensor placement in static deployments. Therefore we designed a two-tier framework, where the first tier takes care of partitioning the time series in segments which can be modeled with a single model and the second tier defines the sampling strategy.

4.2 Related Work

We seek to design an optimal mobile sensing strategy, which provides an appropriate balance between "sensing coverage maximization" and "sensing cost (sampling) minimization" of an individual moving sensor. There are three main topics related to this goal: (1) optimal sensor placement in static WSN; (2) mobile sensing; and (3) time series segmentation.

4.2.1 Sensor Placement in WSN

Determining an optimal sensing placement in an arbitrary sensor field is a kind of *art gallery* problem, which is NP-hard [156] and requires near-optimal solutions like the submodularity method [141]. A couple of efficient near-optimal algorithms are also provided [72, 142, 164]. In the work of Krause et al. [142], a greedy solution is designed by using mutual information when selecting the next sensor point, which has better performance than traditional random solution or other basic entropy-based sensor selection. In [164], the sensor placement is modeled as a min-max optimization problem, and a simulated annealing based algorithm is provided. In [72], the method supports imprecise sensor detection like terrain properties, which can support sensor placement with probabilistic guarantee in a polynomial time. All of these works are not designed for mobile sensors. Nevertheless, relevant optimization formulation and greedy solutions can be adopted in our framework.

4.2.2 Sensing from Mobile Sensors

Recently, a number of mobile sensing applications have been emerging, particularly in urban computing, such as OpenSense for air quality monitoring [2] or floating sensors for water contaminants analytics [278]. One work similar to our optimal sensing strategy for the OpenSense project is the Ear-Phone [238], which provides an end-to-end participatory urban noise mapping system and generates a noise map from a small set of sensor readings in a sparse spatio-temporal sensing field. This is similar to our objective in OpenSense,in which a small number of moving buses are used with environmental sensors (e.g., CO_2). However, our work in OpenSense has additional challenges: (a) the monitoring area in OpenSense is a 2D map, or even a 3D one, not a 1D road line like in Ear-Phone; and (b) OpenSense is real mobile sensing that deploys sensors on moving buses, whilst Ear-Phone fixes mobile phones beside the road.

4.2.3 Time series segmentation

Segmenting time series is an important topic in many areas such as data mining, signal processing, and applications of financial and environmental data. According to well-known time series segmentation studies like [108, 132, 133], the segmentation methods can be divided into three categories, i.e., *sliding window*, *top-down* and *bottom-up*. Sliding window algorithms can work online and efficiently, but the results are poor and sensitive to parameters; whilst *top-down* and *bottom-up* methods have better segmentation results but cannot be directly used in real-time applications. To balance both offline high-accuracy and online efficient-computation, a couple of combination algorithms are proposed, such as SWAB (Sliding Window And Bottom-up) by Keogh et al. [132], piecewise linear segmentation (mixing both constant and linear function) [158], FSW (Feasible Space Window) & SFSW (Stepwise FSW) [172], SwiftSeg (a polynomial approximation of a time series in either sliding or growing windows) [85]. In this project, we study different types of segmentation methods, and evaluate them with the combination of different sampling strategies to achieve optimal sensing for mobile sensors.

4.3 Two-Tier Optimal Mobile Sensing

This section presents our two-tier optimal mobile sensing framework, named OptiMoS, and its problem formulation. Figure 4.1 sketches the framework of OptiMoS for achieving the optimal sensing, as well as the data flow in this procedure.

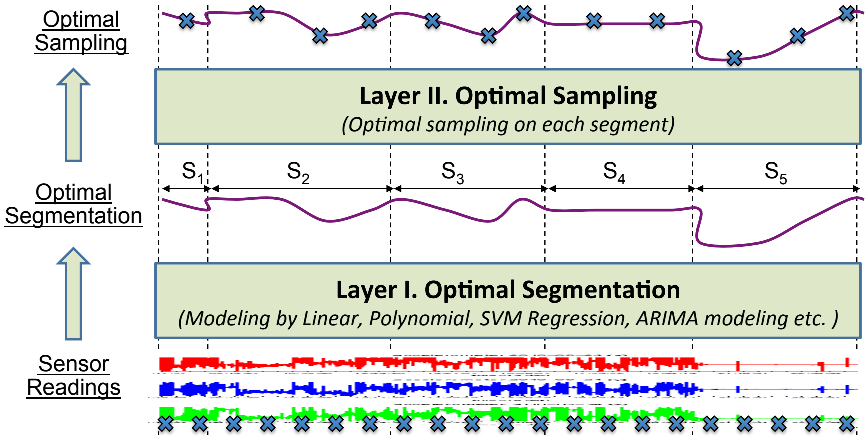

Optimal Sampling
Layer II. Optimal Sampling
(Optimal sampling on each segment)
S1
S2
S3
S4
S5
Optimal Segmentation
Layer I. Optimal Segmentation
(Modeling by Linear, Polynomial, SVM Regression, ARIMA modeling etc.)
Sensor Readings

Figure 4.1: OptiMoS's two-tier optimal sensing framework

In the lower tier of OptiMoS, the initial input is the raw sensor readings collected by moving sensors, i.e., multiple dimensional spatio-temporal time series data. Each reading record is the "×" symbol in Figure 4.1 (i.e., $R_i = \langle t, l, x_1, \cdots, x_m \rangle$), which includes sensing time t, sensing location l, and environmental measurements x_1 to x_m.

The objective of this tier is to find the optimal (or near-optimal) segmentation based on data modeling on these raw readings. OptiMoS supports all kinds of modeling methods, e.g., simple linear regression, polynomial regression, SVM (Support Vector Machine) based regression, time series ARIMA (Auto-Regressive and Moving Average) modeling. As the result of the first tier, we can achieve an optimal (or near optimal) segmentation.

In the upper tier of OptiMoS, we focus on studying individual optimal segments, computed from the lower tier. For each segment, the objective is to find the best sampling from the mobile sensor readings, i.e., to select only a subset of sensor readings ("×" symbols in Figure 4.1) that can keep enough modeling information for regression of the whole segment and interpolation of non-selected sensor readings.

4.3.1 Problem Statement

The problem can be formulated as follows:

Given a sequence of initial mobile sensor readings $\mathscr{R} = \{R_1, \cdots, R_N\}$ *of size N from continuously moving sensors, where each record* $R_i = \langle t, l, x_1, \cdots, x_m \rangle$ *consists of M types of sensor readings (from* x_1 *to* x_m*) together with the timestamp* (t) *and the location (*$l = \langle$ *longitude, latitude, altitude* $\rangle$*), the objective of OptiMoS is to identify the best sampling of such sequences of sensor readings that can guarantee the majority of sensor reading information (i.e., sensor coverage maximization) at the minimum sampling rate (i.e., energy cost minimization).*

As shown in Figure 4.1, our solution to this problem is to provide a two-tier optimization framework. We also compare it with traditional one-tier mobile sensor sampling without segmentation (see the experiment in Section 4.4).

4.3.2 Near-Optimal Segmentation

A model $\mathscr{M}$ on a segment $\mathscr{R}_i$ can be linear, polynomial, SVM regression, ARIMA etc. In this project, we study linear and SVM regression (we use the LibSVM package [45]), and evaluate their performances. We apply the RSS (*Residual Sum of Squares*) to quantify the error for modeling a sequence $\mathscr{R} = \{R_1, R_2, \cdots, R_N\}$ (see Formula 4.1). The residual $res(R_i)$ is computed using $\hat{R}_i$ which is the approximation of R_i by the learned model $\mathscr{M}(\mathscr{R})$.

$$RSS(\mathscr{M}(\mathscr{R})) = \sum_{i=1}^{N} (res(R_i))^2 = \sum_{i=1}^{N} (|R_i - \hat{R}_i|)^2$$
$$\text{where } \hat{R}_i = \mathscr{M}(\mathscr{R})|_{R_i} \tag{4.1}$$

Finding the best K segments is equivalent to identifying the best K-1 division points: R_{d_1}, R_{d_2}, $\cdots$, $R_{d_{K-1}}$; then, for each segment $\mathscr{R}_i$, we have a sub sequence of readings between two division points, i.e, $\mathscr{R}_i = \{R_{d_{i-1}}, R_{d_{i-1}+1}, \cdots, R_{d_i}\}$. For the first segment ($\mathscr{R}_1$), R_{d_0} indicates the first reading R_1. This optimal segmentation problem is an unconstrained optimization problem. Ideally, the segment number (K) is not known in advance, and needs to be discovered automatically as a part of the optimization problem, as shown in Formula 4.2.

$$\underset{K,d_1,d_2,\cdots,d_{K-1}}{argmin} \sum_{i=1}^{K-1} RSS(\mathscr{M}(\{R_{d_{i-1}},\cdots,R_{k_{d_i}}\})) \tag{4.2}$$

For simplicity, in the first step of this work, we can assume K is given and we will test a reasonably small set of different K values (e.g., $K \leq 10$ in our experiment), to analyze the convergence and achieve a near-optimal segmentation.

Optimal approach

The segmentation problem in Formula 4.2 has an optimal solution that can be found using an exhaustive search, where the algorithm is recursively searching the best point to divide i segments into $i+1$ segments. This can be computed by *dynamic programming* (DP) [23, 108], with the complexity of $O(KN^2)$ where K is the segment number and N is the number of points in $\mathscr{R}$. With such quadratic complexity, DP is impractical for segmenting real-life large scale time series. Nevertheless, we can apply DP to segment a short sequence using small $K(\leq 5)$, and evaluate other segmentation methods comparing with the optimal modeling error from DP.

It is worth noting that our objective is to find a segmentation that is not only the optimal one for the training sequence, but also applicable to other similar sequences (as testing data). The segmentation result on one day of *Bus-line-1* should be generally consistent with sensing data from other days on the same line. Therefore, the optimal segmentation from the training day might not be the best for testing days. This is the over-fitting issue which needs to be avoided in OptiMoS. In the experiment, we show such experimental evidence.

Top-down Binary Segmentation

As the optimal segmentation by DP is impractical for real-life long sequence of mobile sensing data because of its high complexity, researchers proposed many greedy segmentation methods, such as the top-down binary split method [132]. The idea is to hierarchically split the sequence with maximum error into two sub-sequences, until the number of segments reaches K. We call this traditional top-down binary segmentation algorithm *Binary*.

In *Binary*, the algorithm always chooses a segment with the maximum model-based regression error (RSS) to make further segmentation, which may cause the division only in one segment and its subsegments. As a result, the segmentation result could be totally unbalanced. To overcome this problem, we design an extended algorithm of *Binary*, called *Binary*$^+$.

```
Algorithm 2: segmentDP ($\mathscr{R}, i, j, k$)
Input  : $\mathscr{R} = \{R_1, R_2, \cdots, R_N\}$          //mobile sensor readings
         $i, j$                  //to make next segmentation in $\langle R_i, R_j \rangle$
         $k$                                //the number of segments
Output : optimalRSS              //model error by optimal segmentation
/* find the optimal segment */
if $k = 1$ and $j > i$ then
    print i, j ;                         //print optimal sub-segments
    return RSS($\mathscr{R}, i, j$);     //compute model error RSS $\langle R_i, R_j \rangle$
/* impossible to find the optimal segment */
if $j - i < k$ then
    return $\infty$;
/* recursive segmentation (from $k$ to $k$-1) */
optimalRSS $\leftarrow \infty$;          //initialize the optimal RSS found so far
foreach $id \in [i+1, j-1]$ do
    firstSegRSS $\leftarrow$ RSS($\mathscr{R}, i, id$);
    restSegRSS $\leftarrow$ segmentDP ($\mathscr{R}, id, j, k-1$);
    totalRSS $\leftarrow$ firstSegRSS + restSegRSS ;
    optimalRSS $\leftarrow$ min{optimalRSS, totalRSS};
return optimalRSS ;
```

In *Binary*$^+$, we modified the *RSS* error measurement to include two types of penalties to prevent *Binary* from always choosing the top *RSS* error and making segments that are too short: a negative penalty according to how much error can be reduced after such segmentation; and a positive one proportional to the length of the new segments:

$$\widehat{RSS} = RSS(\mathscr{M}(\mathscr{R})) - (RSS(\mathscr{M}(\mathscr{R}_{left})) + RSS(\mathscr{M}(\mathscr{R}_{right}))) + \lambda \times length \tag{4.3}$$

Heuristic Segmentation

The *Binary* and *Binary*$^+$ methods only consider the middle point of a segment as the cutting point, but this is not always the optimal solution. Therefore, we additionally design an error-based greedy method that uses the model residual of each record to identify segment division. The residual is computed with Formula 4.1. For this heuristic method, our simple greedy strategy is using the top error point as the division point for the segmentation. Recursively, we recompute the new models for new segments, and find the next top error point as the new division point, until we reach K segments. This segmentation is called "*Heuristic*".

Similar to *Binary* this method can be biased by some outliers and produce extremely short segments. Thus, we design an extended version called "*Heuristic*$^+$" that also uses the penalty

Algorithm 3: segmentBinary$^+$ $(\mathscr{R}, K)$

```
Input  : $\mathscr{R} = \{R_1, R_2, \cdots, R_N\}$          //mobile sensor readings
         $K$                                              //the number of segments
Output : $segQueue$                                       //list of segments
segQueue ← $\emptyset$;                                   //initial priority queue
segQueue.push($\mathscr{R}$, 1, $N$);   //insert the first segment into the priority queue
foreach $k \in [2, K]$ do
    /* retrieve & remove top error segment from the queue */
    topErrorSeg ← segQueue.pop();
    /* divide the segment into two subsegments */
    $S_1$ ← ($\mathscr{R}$, topErrorSeg.begin, topErrorSeg.division);
    $S_2$ ← ($\mathscr{R}$, topErrorSeg.division, topErrorSeg.end);
    /* add the two new sub segments into two the sorted list*/
    calculate $\widehat{RSS}$ for $S_1$ and $S_2$;
    segQueue.push($S_1$);
    segQueue.push($S_2$);
return segQueue ;
```

function from *Binary*$^+$ to avoid two division points being too close, i.e., the segment is too short. In addition, *Heuristic*$^+$ does not look for the largest error, but for the longest contiguous sequence of errors exceeding a certain threshold (e.g. the error median) and then randomly choosing one of its endings. This way it can isolate segments that have a bias in $\mathscr{M}(\mathscr{R})$.

The *Binary* and *Binary*$^+$ methods focus on finding the best segment to divide; whilst the *Heuristic* and *Heuristic*$^+$ methods focus on finding the best division points to apply the segmentation. The last segmentation method we propose in OptiMoS, named B^+H^+, combines *Binary*$^+$ and *Heuristic*$^+$. The idea is to consider both "the maximum error segment to divide" but also "the maximum error subsequence to apply the division". The combination is done as follows: the segment to divide is chosen by *Binary*$^+$ and then, inside this segment, *Heuristic*$^+$ is applied to find the segmentation point. This way we ensure that the segments have a better distribution, like in *Binary*$^+$, but also that the segmentation points are put in regions where the current model has its worst performance and thereby a certain improvement can be expected.

4.3.3 Near-Optimal Sampling

After getting the optimal segmentation, OptiMoS needs to identify the best sampling of mobile sensor readings for each individual segment. To quantify whether a sampled reading sequence $\mathscr{R}_{sub}$ is good or not, we define the "*information loss*", $\mathscr{L}(\mathscr{R}, \mathscr{R}_{sub})$, i.e., the RSS increase ratio between $\mathscr{R}_{sub}$ and the complete readings $\mathscr{R}$.

$$\begin{aligned}\mathscr{L}(\mathscr{R},\mathscr{R}_{sub}) &= \frac{RSS(\mathscr{M}(\mathscr{R}_{sub}\rightarrow\mathscr{R})) - RSS(\mathscr{M}(\mathscr{R}))}{RSS(\mathscr{M}(\mathscr{R}))}\times 100[\%] \\ &= \frac{\sum\limits_{i=1}^{N}(R_i - \mathscr{M}(\mathscr{R}_{sub})|_{R_i})^2 - \sum\limits_{i=1}^{N}(R_i - \mathscr{M}(\mathscr{R})|_{R_i})^2}{\sum\limits_{i=1}^{N}(R_i - \mathscr{M}(\mathscr{R})|_{R_i})^2}\end{aligned} \tag{4.4}$$

where, $RSS(\mathscr{M}(\mathscr{R}_{sub}\rightarrow\mathscr{R}))$ means the RSS error for the approximation of the complete sequence $\mathscr{R}$ by using the model $\mathscr{M}(\mathscr{R}_{sub})$ that learned from the sub sequence $\mathscr{R}_{sub}$.

Similar to reformulating the sensor placement problem in static WSN, we chose to define the optimal sampling problem in OptiMoS by: *Given a limited sampling rate δ, find the best sampling set $\mathscr{R}_{sub}$ that has minimum information loss $\mathscr{L}(\mathscr{R},\mathscr{R}_{sub})$*, as presented in the Formula 4.5. For the experimentation we chose δ from the set $\{1, 1/2, 1/3, 1/4, 1/5\}$.

$$\underset{\mathscr{R}_{sub}}{argmin}\ \ \mathscr{L}(\mathscr{R},\mathscr{R}_{sub})\ \ \text{such that}\ \ |\mathscr{R}_{sub}|/|\mathscr{R}| \leq \delta \tag{4.5}$$

Optimal approach

The optimal sensor placement (or sampling) in an arbitrary sensing field is an NP hard problem [72, 142, 164]. For a simplified problem with a limited number of mobile sensing points (N), the optimal sampling at the sampling rate δ requires an exhaustive search amongst all possible subsets of readings. This needs to consider all combinations of δN points from the initial N points, which has the complexity of $O(N^{\delta N})$ and is still NP-complete. Therefore, we look for near-optimal greedy solutions for optimal sampling in OptiMoS.

Distribution based Sampling

Intuitive greedy solutions for sampling the mobile sensor readings for each individual segment are using some statistical distributions, e.g., *uniform* and *normal.* In this project, we evaluate the uniform sampling and the random sampling.

- *Uniform Sampling*– To uniformly select the δ percentage of mobile reading points in the segment, the algorithm selects the sensing point at each interval δN. We can apply such uniform sampling $\delta N - 1$ times, each time with a different offset. The final accuracy of the uniform sampling is the average of several trials.
- *Random Sampling*– In this method, we select randomly, with a uniform distribution, δN points from the segment, and make a certain number of trials. Similar to the uniform

sampling, the final accuracy of random sampling is the average of several trials.

Entropy based Heuristic Sampling

In distribution based sampling, selecting sensing points only considers position distribution. There is no bias in general and it is not taking into account the relevance of a certain measurement for the model.

To provide better sampling, we design an error-based entropy sampling. Similar to the heuristic segmentation methods using modeling errors, we first applied the residual $res(R_i)$ as the entropy for the selection of sensor readings, and kept only the points with the highest error. But this was too sensitive to outliers and bias of the model. So we designed an entropy approximation using a relative residual rather than the absolute value. The idea is using a *leave-one-out* basic model to approximate the residual and check how large the gap is between real residual and the approximated one (see Formula 4.6). To approximate the residual of point R_i, the *leave-one-out* model uses the residuals of nearby points (in window size $2w$, i.e. w for left and w for right) and builds simple interpolation (e.g., using basic mean, linear, Gaussian).

$$\hat{res}(R_i) = res(R_i) - \text{leaveOneOutAppr}(R_i, w) \tag{4.6}$$

Mutual Information based Heuristic Sampling

Both the absolute error *res* and the relative error $\widehat{res}$ in the entropy based sampling are calculated using the first computed Model. However, the relevance of sampling points are actually varying after each sampling step. Such information change can be modeled using the *mutual information* that can measure the mutual dependence of two random variables (in our case the candidate point to be selected and the points already selected).

Mutual information can reduce information dependency, therefore it has been extensively used in many topics such as feature selection [219] and traditional static sensor placement [142]. Therefore, we additionally designed a mutual information based sampling method in OptiMoS.

We defined a loop procedure that recursively recomputes the new $\widetilde{res}$ of the candidate sensor readings. Such new $\widetilde{res}$ needs to remove the mutual information from sensor readings already selected (see Formula 4.7).

$$\widetilde{res}(R_i; \mathscr{R}_{sub}) = \hat{res}(R_i | \mathscr{R}_{sub}) - \widehat{res}(R_i) \tag{4.7}$$

where, $\mathscr{R}_{sub}$ is a set of sampling points already selected so far, R_i is a candidate reading for adding to $\mathscr{R}_{sub}$, $\hat{res}(R_i|\mathscr{R}_{sub})$ is the relative residual computed only using the selected readings $\mathscr{R}_{sub}$, and $\hat{res}(R_i)$ is the relative residual computed by all of the sensor readings $\mathscr{R}$ without sampling.

The first point is taken using the relative error based entropy; then, we recursively compute the mutual information between the candidate data reading and the selected data readings, and choose the sensing point with the maximum new residual that removes the redundancy from existing sampling points. This procedure is stopped when it reaches the final sampling rate.

Algorithm 4: samplingMutualInfo ($\mathscr{R}, \delta$)

```
Input  : $\mathscr{R} = \{R_1, R_2, \cdots, R_N\}$            //mobile sensor readings
         $\delta$                //the sampling rate (ratio of samples to keep)
Output : $\mathscr{R}_{sub}$                        //sampling set, with size $\delta N$
/* initialization */
$\mathscr{R}_{sub} \leftarrow \emptyset$;                          //initial empty sampling set
$M \leftarrow round(\delta N)$;                   //the size of the final sampling set
/* get the first sample with only entropy */
foreach $R_i \in \mathscr{R}$ do
    compute the entropy $\hat{res}(R_i)$;                          //by Formula 4.6
$firstSample \leftarrow \underset{R_i}{argmax}\ \hat{res}(R_i)$;
$\mathscr{R}_{sub} \leftarrow \mathscr{R}_{sub} \cup firstSample$;
/* get the following samples with mutual information */
while $|\mathscr{R}_{sub}| < M$ do
    foreach $R_i \in \mathscr{R} - \mathscr{R}_{sub}$ do
        compute $\widetilde{res}(R_i; \mathscr{R}_{sub})$;                     //by Formula 4.7
    $nextSample \leftarrow \underset{R_j}{argmax}\ \widetilde{res}(R_j)$;
    $\mathscr{R}_{sub} \leftarrow \mathscr{R}_{sub} \cup nextSample$;
return $\mathscr{R}_{sub}$ ;
```

4.4 Experimental Evaluations

This section presents the experimental results of our two-tier optimal mobile sensing framework,OptiMoS. We evaluate OptiMos' different segmentation strategies and various sampling methods using real-life environmental data from mobile sensors from OpenSense [2]. All the input data (e.g., CO_2 values) were linearly scaled to $[0, 1]$ for normalization. The algorithms were implemented in Java using the API provided by Weka [98] for building the models (SVM regression and linear).

4.4.1 Segmentation Results

To evaluate the various segmentation methods proposed in Section 4.3.2 and make a comparison, we define a metric for quantifying the model error reduction by segmentation (from the initial non-segment sensor readings $\mathscr{R}$ to the K segments $\mathscr{R}_i$), i.e., the ratio of modeling errors called "*RSS_Ratio*".

$$RSS\_Ratio = \frac{\sum_{i=1}^{K}(RSS(\mathscr{R}_i))}{RSS(\mathscr{R})} \times 100[\%] \tag{4.8}$$

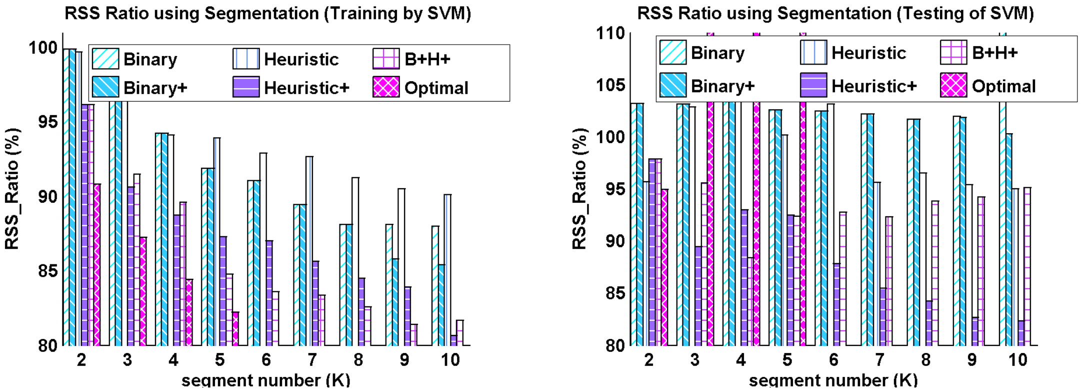
The Figure 4.2a presents the *RSS_Ratio* achieved using SVM regression on one-day mobile sensing as the training data. The six segmentation methods (i.e., *Binary*, *Binary*$^+$, *Heuristic*, *Heuristic*$^+$, B^+H^+, and *Optimal*) are tested with different segment numbers, from 2 to 10. Clearly, with more segments (i.e., larger K), the error ratio can be reduced. Initially, such error reduction is significant at small K, but later it becomes more stable when K becomes larger.

For individual segmentation methods on the training *RSS_Ratio* errors, we observed that the *Heuristic*$^+$ and B^+H^+ methods are better than other segmentation methods. We additionally compared them with the optimal solution (*Optimal*) using dynamic programming for small numbers of segments (i.e., $K \leq 5$). We omitted the optimal solution for larger K, as the computation time was too expensive. The *RSS_Ratio* achieved by the *Heuristic*$^+$ and B^+H^+ methods are closer to the optimal solution compared to other methods. Therefore, the error-based heuristic methods are well suited for model-driven segmentation.

To further evaluate the segmentation results learned from one day training data, we tested them on mobile sensing data from other days on the same bus line. This test can evaluate the robustness of our segmentation results. Fig. 4.2b shows the testing errors as *RSS_Ratio*. We observed that *Heuristic*$^+$ is clearly better than other methods, and even better than the *Optimal* for most cases ($K = 3,4,5$). This is the over-fitting problem, i.e., the optimal segmentation for training data is not necessarily the best for the testing data.

Figures 4.2a and 4.2b present the segmentation results using the SVM regression model. For linear regression, we observe similar trends amongst different segmentation methods. Additionally, the optimal segmentation using linear model in the training data works even worse on the testing data. The linear model has more prominent over-fitting problems compared to the SVM modeling.

The Figure 4.3 shows the regression values with segmentation (obtained from *Heuristic*$^+$ when K=5). We observe better regression results by using segmentation, particularly at the duration of 0am-10am. In terms of the concrete modeling errors, $RSS(\mathscr{M}(\mathscr{R}))$ using SVM regression is 0.070 and $RSS(\mathscr{M}(\mathscr{R}))$ by linear model is 0.081; the two models respectively have model error decreased by 12.7% and 19.3% compared to their modeling errors without segmentation.



(a) Training on Day-1

(b) Testing on Day-2

Figure 4.2: Results using SVM-inferred segments from the training set applied on the testing set

Therefore, segmentation can help more in reducing the error for a simple model (e.g., linear) compared to an advanced one (e.g., SVM).

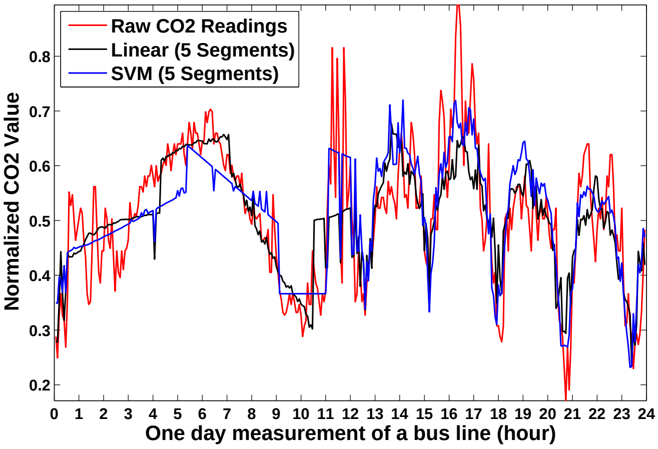


Figure 4.3: Regression in segments ($Heuristic^{+}$, $K = 5$)

4.4.2 Sampling Results

To evaluate the different sampling methods that we presented in Section 4.3.3, we adapted the *RSS_Ratio* formula to compute the *RSS* from the model built with the sampled measurements, as shown in Formula 4.9. Those definitions, 4.8 and 4.9, can then easily be merged for evaluating combinations of segmentation and sampling as presented in the next section.

$$
\begin{aligned}
RSS\_Ratio &= \frac{RSS(\mathcal{M}(\mathcal{R}_{sub} \rightarrow \mathcal{R}))}{RSS(\mathcal{M}(\mathcal{R}))} \times 100[\%] \\
&= \frac{\sum_{i=1}^{N}(R_i - \mathcal{M}(\mathcal{R}_{sub})|_{R_i})^2}{\sum_{i=1}^{N}(R_i - \mathcal{M}(\mathcal{R})|_{R_i})^2}
\end{aligned}
\tag{4.9}
$$

Figure 4.4 presents the experimental results when using four sampling methods, i.e., *Uniform*, *Random*, *Entropy* and *Mutual_Information*, with various sampling rates (i.e., $\delta = 1/2, 1/3, 1/4, 1/5$). In the plot, the *RSS_Ratio* from the *Uniform* and *Random* sampling methods are the average over 10 runs. It is interesting to notice that the uniform sampling on the average has better performance than the random sampling. Both *Entropy* and *Mutual_Information* sampling methods have quite good results. There is almost no error increase compared to the original data sequence when using *Entropy* and *Mutual_Information* based samplings at high sampling rate ($\delta \geq 1/4$). Their *RSS_Ratio* are almost 100%. However, at sampling rate $\delta = 1/5$, we clearly see that *Mutual_Information* performs better than *Entropy*.

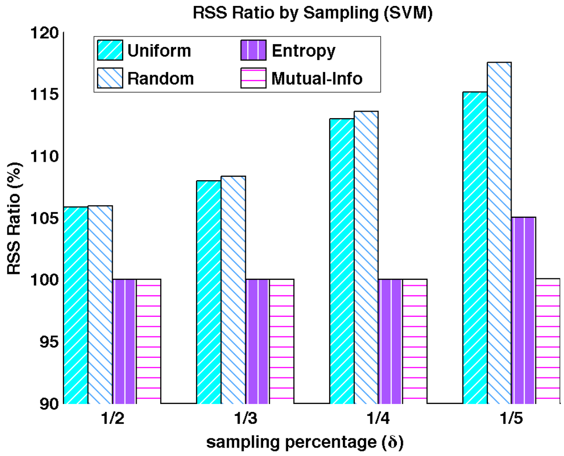


Figure 4.4: RSS Ratio by different sampling methods (SVM)

To further study the performance of *Entropy* and *Mutual_Information* methods at smaller sampling rate ($\delta < 1/5$), Fig. 4.5 shows the results of sampling using a linear model. We observe a similar trend with the SVM model in Fig. 4.4 at $\delta \geq 1/5$, i.e. *Mutual_Information* is the best for all cases. It is worth noting that *Entropy* has a better performance at high sampling rate ($\delta \geq 1/5$), almost the same as *Mutual_Information* when ($\delta \geq 1/4$), but it becomes significantly worse when the sampling rate is too small ($\delta < 1/5$). This is because *Entropy* sampling is always choosing the top-ranking error points, without consideration of currently selected points. This can work well for large sampling rates, but for small sampling rates, the data points will have more bias and thus a decreased performance.

We have already seen with sampling rate $\delta \geq 1/4$ in Figure 4.5 and 4.4, both *Entropy* and *Mutual_Information* sampling methods can guarantee almost 100% *RSS* ratio, i.e., minimal

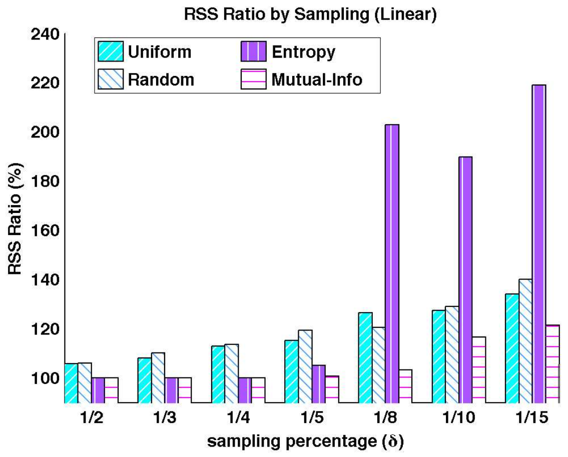


Figure 4.5: RSS Ratio by different sampling methods (Linear)

informational loss using sampling data compared to using the complete data. To compute how many samples are needed for achieving such a minimal informational loss, we study the convergence of *RSS_Ratio* w.r.t. the sampling rate. The Figure 4.6 shows the *RSS_Ratio* convergence for sampling the one-day mobile sensing measurement (1440 points in total for 24 hours with one record per minute). We observe that both linear and SVM regression can achieve almost 100% *RSS_ratio* from 128 sampling points, i.e., the sampling rate at $\delta = 128/1440 \simeq 1/11$. In such case, the sampling rate at 1/11 can guarantee almost zero information loss modeling (around 110% *RSS* ratio), which is a very effective sampling.

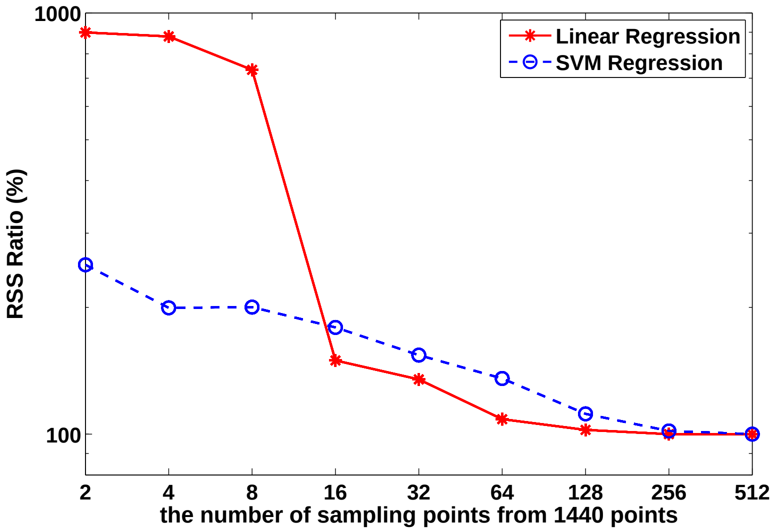


Figure 4.6: RSS Ratio convergence using Mutual-Information sampling

4.4.3 Near-Optimal Combination

The final experiment is to study the exhaustive strategies of mobile sensing in OptiMoS, i.e., combining different segmentation and sampling methods. To keep the graph readable, we do not show all possible combinations, but select the ones that showed good performance in the previous experimental results, while having an efficient implementation. For the segmentation, B^+H^+ and *Optimal* are not shown and for sampling, the performance of *Entropy* based sampling is as good as *Mutual_Information* when sampling rate is reasonably high ($\delta \geq 1/4$), while being more efficient to compute. Therefore, Figure 4.7 shows the 4×3 mobile sensing strategies in OptiMoS, i.e., combining four segmentation methods (*Binary*, *Binary*$^+$, *Heuristic*, and *Heuristic*$^+$) with three sampling methods (*Uniform*, *Random*, and *Entropy*) using SVM regression.

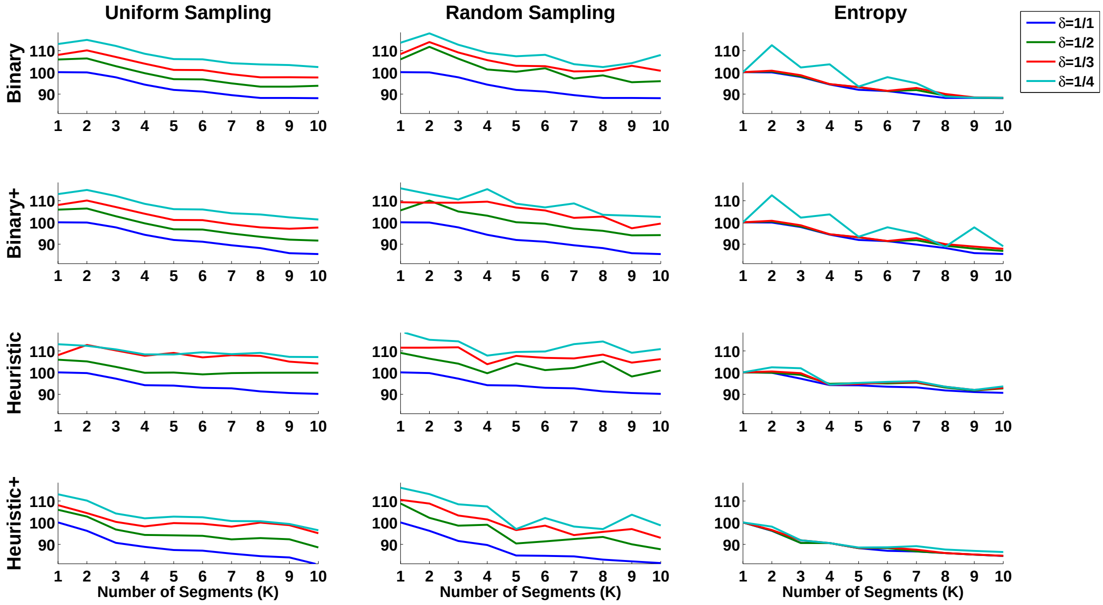


Figure 4.7: RSS Ratio for different combinations of segmentation and sampling (SVM)

For each combination in Figure 4.7, the plot shows the convergence of *RSS_Ratio* w.r.t. the number of segments (K) at different sampling rates (δ). All the plots show a decreased trend of *RSS_Ratio* with more segments (from 1 to 10), and with larger sampling ratio (from 1/4 to 1/1). The *Entropy* sampling with heuristic segmentation (both *Heuristic* and *Heuristic*$^+$) show the best convergence when segment number $K \leq 3$; for $K \geq 4$, the combination of *Entropy* and *Heuristic* still keeps the best *RSS_Ratio* convergence.

4.5 Conclusion

In this chapter, we defined a novel and complete two-tier framework, namely *OptiMoS*, that enables an optimal mobile sensing strategy. We studied both segmentation and sampling of mobile sensor readings, and designed several methods for segmentation (*Optimal*, *Binary*, *Binary*$^+$, *Heuristic*, *Heuristic*$^+$, and B^+H^+) and sampling (*Uniform*, *Random*, *Entropy*, and *Mutual_Information*). We analyzed real-life environmental monitoring sensors on moving mode of transport, built exhaustive experimental studies to validate our optimal mobile sensing strategy, and showed its good performance.

5 State identification

5.1 Introduction

As described in the previous chapter, mobile devices can have clearly defined states, in which the sensed data can be well approximated by a single model, such systems will be later on called *stateful systems*. The specific pattern signature of the states could be, for example, used for annotating the measurement time series with contextual information. We present in this chapter an algorithm for finding those stable subsequences in time series, in an online fashion, while optimizing for devices with limited computational resources.

Stateful systems are not limited to smartphones but they also include computer systems, whose states can be observed through various metrics and smart-meters. The latter can monitor the power consumption from a single appliance to the whole household with a very high temporal resolution [220]. Their integrated storage and computational capabilities enables them to predict their own future states or usage patterns; this could be useful, e.g., for participating in a demand response program [303], or if electricity prices follow real-time dynamic pricing schemes, then drawing/storing energy from the grid when the price is cheap and using it later when the price is expensive could save money [191, 289].

States, in the case of smartphone sensors, can take on various semantic meanings. For example, repeated accelerometer patterns can reflect the activities of the user and the GPS patterns would reveal her daily routine. Putting these together would make it possible to forecast the behavior of the user and to adapt recommendations about the clothes to wear. Knowing the habits of a user can also help in detecting when she deviates from it and would be a case for helping her in this new and unknown situation, like proposing places to visit in a new city. In other systems, these deviations are good indicators of unusual operation and anomalies. For smart-meters , e.g. unusually long *on* states detected on an appliance could mean that the owner forgot to turn the appliance off. Detecting this unintended event leads to energy and cost savings. Another example is an atypical power consumption pattern, which could mean that the device requires maintenance or indicate an imminent failure.

However, the data generated by all these sensors (from buildings, public infrastructure, smartphones, smart electronic appliances), is big data in every dimension: big volume, big velocity and big variety. A centralized approach typically involves communicating data from sensors to a central server for processing. Most of the today's popular location-based services are examples of centralized solutions, i.e. GPS information from a mobile phone is communicated to a central server, computation for the service requested is performed on that server and the result is sent back to the phone. Similar processes are envisioned for participatory sensing scenarios. Additionally, since sensor data contain sensitive personal information, this solution is susceptible to breach of privacy. It also requires vast data storage and incurs significant computation and communication costs. With an increasing number of sensors, scalability becomes a serious issue. Furthermore, the high cost of large storage requirements could potentially cause sensor data processing to be dominated by a few big players (Big Brothers holding most of our data).

Therefore, identifying the states would also help in reducing the storage needs for the time series, by allowing a higher level representation, but also making the further processing of the time series faster by removing some of the noise and working on smaller data. This enables the devices to work in a decentralized way, where we store and process the data as close to the sensors as possible. Since only a little data (usually aggregated) leaves the sensor or smartphone, privacy is preserved, communication costs are greatly reduced and the whole system is more scalable. However, the advantages of a decentralized approach require that other challenges be solved: smart devices (such as smart appliances and smartphones) only have limited storage and computational power. Solving these challenges means processing big (streaming) data as efficiently as possible while maintaining data usability.

Moreover, to be able to handle such a variety of systems, we need an unsupervised algorithm as it would be too tedious to generate the sets of potential states for each different system and, as their environment may change, new states can appear that would not be recognizable in a supervised framework, if they weren't present in the training phase.

To this end, the solution should satisfy the following requirements:

1. For more efficient processing, streaming sensor data should be processed online (efficiently, some limited delay can be tolerated);
2. For more efficient storage, data can be transformed into another representation. However, it should allow efficient processing of data mining algorithms, without “reverting” the representation; Encoding can be lossy, but the states must be preserved as much as possible;
3. Since a sensor is able to measure different thing, the solution should be able to handle multidimensional time series;
4. Since not all states can be observed apriori, the solution should be able to learn in an unsupervised way.

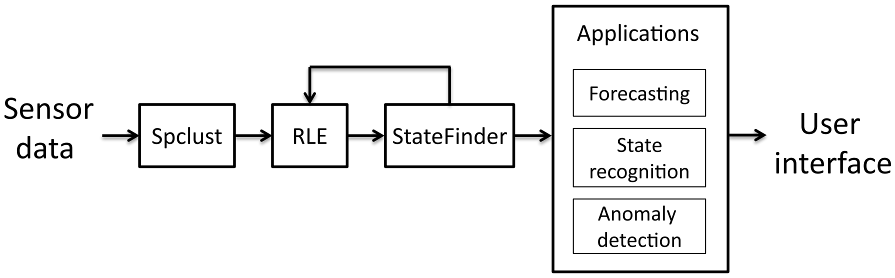


Figure 5.1: An overview of our solution. We start with the initial symbolization process (Spclust, see Section 5.3). Next, we repeatedly compress the representation using run-length encoding (RLE) and recognize states using StateFinder (see Section 5.4). The output is then used in various applications (see Section 5.5 and 5.6).

5. Since the length of states might vary, the solution should also be able to find states of various lengths. Thus, algorithms that assume a fixed/predefined length of states (or sequence or sliding windows) might not be suitable.

5.1.1 Solution overview

Hitherto, there is no solution in the literature which fulfills all the requirements above (see our literature review in Section 5.2). This work fills the gap and proposes a framework which satisfies them (see Figure 5.1).

Figure 5.1 shows the overview of our framework. First, we start with the initial symbolization process (Spclust), i.e. we transform sensor data (time series) into a symbolic representation. This step is useful for handling multidimensional time series while also reducing the data size. Simple methods such as median or uniform segmentation [301], or other clustering techniques, such as KMeans, can also be used. Although in principle, any segmentation or clustering method can be used in this step, one should bear in mind the requirements above.

Second, we repeatedly recognize states and convert them into symbolic representations. We assume that a system is more likely to maintain its current state rather than to change to another. Thus, in order to have a more compact representation, we apply run-length encoding (RLE) on the resulting symbols. Our algorithm *StateFinder* is then used on the symbolic representation to find the system's states.

By looking back at the two steps of our framework OptiMoS, in Chapter 4, we can generalize the second one, namely sampling, as a kind of encoding. Indeed, we tried to obtain a representation of the time series segment with a subset of its points such that the decoding function would perform some regression to approximate all the original points.

We maintain these principles here, with some variations. First we redefine the objective function for the segmentation, by using a Markovian model and looking for states (see the definition of a state in Section 5.4). Second, we encode the states by mapping the model parameters, in this case a transition matrix, to new symbols. Also we only consider the segments which satisfy our definition of a state and do not encode the transitions between them. These can for example be the anomalies.

Then the sequences of symbols which are recognized as a state are converted into another (higher level) symbol, with similar states most likely being converted into the same symbol. The next iterations of RLE and StateFinder are applied until there are no more states to be found or a predefined level is reached. The entire process is performed in an online and unsupervised way. Finally, the end result is a stream of symbols of greatly reduced size compared to the original stream, and each symbol represents a state of the system.

5.2 Related Work

To the best of our knowledge, our work is most similar to that of Wang et al. [292], as they include segmentation and clustering of the segments and where the clusters represent states of a Markov Chain. But the similarity stops there. Their time series subsequences are defined by line segments, they model the process with a Hidden Markov Chain and require an extended Viterbi algorithm to find the optimal state sequence. After the initial segmentation, a second refinement step is needed, which makes the algorithm less suitable for online processing. All segments are associated with a state, and they do not discards transition phases and meaningless phases. Other related works can be separated into the following three categories depending on their primary goal, which of course does not mean that they do not apply the other processes.

5.2.1 Time series segmentation

For identifying the start and end of a state, segmentation has to be carried out. The different approaches are all specific to the properties of the state they have to delimit. But all these methods assume that every part of the time series must belong to a state or a segment. On the contrary, we want to keep the transitional part, as well as the rare sequences and anomalies, as raw symbols so as to lose less information.

Complementing the works presented in Section 4.2.3, a large family of methods for segmenting time series aim to summarize them based on piecewise approximations using different functions [158], rather like our first step of the symbolization process. Segmentation points are found either by minimizing the model's fitting error or at regular intervals, as with the Piecewise Aggregate Approximation [130]. Other methods [59] map a predefined set of patterns onto the time series to identify the segments dynamically and more advanced schemes [96] directly use a set of approximation functions to select the best adapted models for each segment. In the latter, precision and recall are also used as metrics for evaluating the fitness of a model.

Other methods, meanwhile, aim more specifically at identifying the system's operating modes. Srinivasan et al. [271] use the process state itself in a feedback loop, as contextual information to classify the patterns in the time series. The classification's output labels then allow the time series to be segmented into the operating process's different states. For the same purpose, but using HMM and a modified Viterbi algorithm, Kohlmorgen and Lemm [139] presented an online algorithm for segmenting time series based on probability density.

5.2.2 Online frequent pattern mining

In addition to segmentation, online processing has also been pushed into the field of mining frequent patterns. But there is a fundamental difference between our definition of state and what is referred to as patterns in the following papers: patterns are characterized by their length, whereas states are by their dynamics. Moreover a specific pattern can appear in different states, depending on the time series dynamics.

In their survey, Han et al. [99] (Chapter 5.3) present some methods for online frequent pattern mining of data streams, but the algorithms actually focus on frequent itemsets. Mueen and Keogh [196] also applied their algorithm maintaining exact motifs over a time-window to do a summarization or compression of a time series. But like the other more recent works [47, 275], and those mentioned in the survey by Aggarwal [6], all these algorithms show size-dependent pattern identification.

5.2.3 Time series subsequences clustering

Ramoni et al. [236] present a clustering algorithm, namely BCD, which is able to cluster time series subsequences, using "event markers" (simultaneous changes on at least 3 sensors) as segment separators. Each segment is then represented as a transition matrix of a first order Markov chain. These transition matrices can be averaged with a background knowledge sample matrix. Then the clustering algorithm partitions the matrix set using the probability of partition, given the data. Matrices are merged recursively with the nearest ones, according to their symmetrized Kullback-Leibler distance, until the marginal likelihood cannot be improved.

In the work by Denton et al. [67], time series subsequence clustering is done using a kernel-density-based algorithm. The subsequences are extracted using a sliding time window of fixed length. Comparing their distribution in feature space with random walk data allows for outliers to be discarded and only clusters the more frequent patterns. However, the size of the subsequences has to be predefined and only similar ones can be compared, as they use the Euclidean distance.

In [148], the authors use the SAX encoding for a 2-level scheme clustering whole time series. They use DTW (dynamic time wrapping) for aligning patterns and CAST for clustering, using affinity between points to decide if they should belong to a cluster. But CAST, like DBSCAN, also need a threshold for the similarity. In our case, we do not need to perform a costly DTW, because representing the subsequence as a transition matrix removes the notion of absolute position of the symbols. They also use Euclidian distance between the timeseries, thus the need for DTW. They had to redefine the distance to apply it on SAX symbols, defining them equidistantly. On the first level they cluster symbolized whole time series and then within each cluster, they again apply clustering, but on raw data.

Rakthanmanon et al. [235] introduced the idea, which we also applied, that not all the data should be clustered and transition phases should be ignored. They developed a clustering algorithm working on discrete time series, based on the Minimum Description Length principle. Motifs are clustered recursively for as long as the description length of the time series can be reduced, using Euclidean distance between subsequences and entropy to compute the description length. However, the external motif discovery algorithm used needs to know the length of the motifs beforehand, or it must try all possibilities.

Overall, these clustering algorithms are offline as they either use information from the whole time series or need several passes through the time series before assigning the final cluster label to a segment. On the contrary, our algorithm only needs one pass and is designed to require only a limited amount of memory. In the case of multiple levels of StateFinder, they can be performed one after the other in a pipeline and thus only needing one pass.

5.2.4 Time series features extraction

Alternative representations of time series can be generated by extracting features from the time series, usually using a sliding window [5]. But one can also use an infinite window with an exponential decay or define each window instance by first segmenting the time series. Fulcher and Jones [86] presented in their work an extensive list of the feature extraction methods used in various area of research such as particle physics, meteorology, finance or medicine. Combining with various preprocessing methods and varying the parameters allows the generation of a very large amount of different features. The presented features can be grouped into general categories:

- **Basic Statistics**: this group includes the min, max, average variance and other moments, but also zero-crossing, range, auto-correlation and automutual information among others. They are widely used and fast to compute, but taken separately, they are weak classifiers.
- **Frequency Domain**: this group uses the same basic statistics but after applying a Fourier transformation to the time series. It includes also the wavelet based features. They are also relatively fast to compute, but are more meaningful in the case of an oscillating time series.
- **Model-Based**: this group uses various models to fit the distribution or the time series and the features can be the models parameters or residuals. Depending on the model

these features can be costly to compute, but would tell a lot about the underlying time series if the model matches well.

- **Entropy-Based**: this group contains the methods to approximate the time series entropy, such as the Approximate Entropy [93], Sample Entropy [245], distribution entropy, Lempel-Ziv complexity,etc. They are slower to compute than the basic statistics and are not specific to any kind of time series.
- **Domain Specific**: this group includes the computation of the fractal dimension and related values, but also hypothesis testing on the time series, and ECG specific heart rate variability features. They are usually slow to compute and make strong assumptions about the kind of time series.

Our approach can be classified among the Model-Based ones, using a Markovian model which represents the dynamics of our states. But by using two steps, namely segmentation and encoding, we make it faster to compute by selecting the relevant segments before mapping them to symbols. The choice of a fading factor instead of the sliding window allows us to deal with arbitrary time scales of the time series, as we will see later.

5.3 State-Preserving Symbolic Representation

In this section, we outline an initial step needed to deal with noisy n-dimensional time series.

Definition 1. *We define an* n-dimensional time series *formally as* $S = \{s_1, s_2, s_3, \ldots\}$. *Each* $s_i \in S$ *is a tuple* $(t_i, \vec{V_i})$, *where* t_i *is a timestamp and* $\vec{V_i} \in \mathbb{R}^n$ *is a vector of measurements. We have* $i < j$ *iff* t_i *is earlier than* t_j.

Next, we explain how we learn to map measurement vectors (or *measurements*) onto symbols to reduce data granularity and process it more efficiently, while preserving system's states as much as possible. This process can also be viewed as a preprocessing step (see Figure 5.1). The learning algorithm below, *Spclust*, can be run either during the sensor calibration phase or on top of historical data that capture the measurement distribution. As similar measurement vectors are converted to the same symbol, the symbolization process can also be thought of as a clustering process, where measurement vectors are replaced by their cluster labels. We assume that the initial data used to build the direct mapping between symbols and measurements is representative of the measurement distribution of the whole time series.

5.3.1 State Preservation Index

Measurements which are more likely to belong to the same state, should be converted into the same symbols. Thus, we consider the following two principles when developing our symbolic representation: (i) measurements with similar values are more likely to belong to the same state, and (ii) a system is more likely to stay in its current state than change to another state.

Let $C = \{c_1, \dots, c_{|C|}\}$ be a cluster configuration (a set of clusters), where each cluster $c \in C$ is an n-polytope. Since a measurement $\vec{V}$ is essentially a point in an n-dimensional space, we write $\vec{V} \in c$ if the point $\vec{V}$ lies in the n-polytope c. We express the existence of a connection between clusters c_1 and c_2 through s_i as a binary function:

$$is\_conn(c_1, c_2, s_i) = \left\{ \begin{array}{ll} 1 & \text{if } \vec{V}_i \in c_1 \wedge \vec{V}_{i+1} \in c_2 \\ 0 & \text{otherwise} \end{array} \right\}, \tag{5.1}$$

and its distance as:

$$dist\_conn(c_1, c_2, s_i) = \left\{ \begin{array}{ll} dist(\vec{V}_i, \vec{V}_{i+1}) & \text{, if } \vec{V}_i \in c_1 \wedge \vec{V}_{i+1} \in c_2 \\ 0 & \text{, otherwise} \end{array} \right\}. \tag{5.2}$$

In our experiments (see Section 4.4), we use the Euclidean distance. However, any distance metric for two vectors could also be used here.

We define the number of connections between c_1 and c_2 as

$$\#conn(c_1, c_2) = \sum_{s_i \in S} is\_conn(c_1, c_2, s_i). \tag{5.3}$$

And, for a cluster c, we define its diameter, *diam*(c), as the farthest distance between any two measurements in c. We measure how *compact* the clusters in configuration C are by:

$$intra\_coeff(C) = \sum_{c \in C} \frac{\#conn(c, c)}{diam(c)}, \tag{5.4}$$

i.e., the higher the *intra_coeff* value, the more compact the clusters are. Note that, the division by the cluster diameter is needed, so that the measure is cluster-size invariant.

We define the average connection length between two clusters, c_1 and c_2, as:

$$avgConnLen(c_1, c_2) = \frac{\sum_{s_i \in S} dist\_conn(c_1, c_2, s_i)}{\#conn(c_1, c_2)}.$$

Then, we measure the connection density between clusters in configuration C as:

$$inter\_coeff(C) = \sum_{c_1 \in C} \Big[\sum_{\substack{c_2 \in C, \\ c_2 \neq c_1}} \frac{\#conn(c_1, c_2)}{avgConnLen(c_1, c_2)} \Big], \tag{5.5}$$

i.e., the fewer the connections between clusters and the farther apart the connections between them, the lower the *inter_coeff* value. As we prefer to have clusters which are separated as far as they possibly could be, and with fewer connections between them, lower *inter_coeff* is

Algorithm 5: Spclust

```
Input: $n$-dimensional measurement space $M$, grid size $G$, threshold $T$
Output: cluster configuration
 1 quantize $M$ into $n$-dimensional hypercubes of length $G$
 2 forall the hypercubes $h$ do
 3     count measurements in $h$
 4     if $count \geq T$ then
 5         $density(h) \leftarrow 1$
 6     else
 7         $density(h) \leftarrow 0$
 8 find connected component (clusters) on all hypercubes $h$, where $density(h) = 1$
 9 assign distinct label to each cluster
10 enlarge the clusters incrementally to include hypercubes $h$ with $density(h) = 0$ until all
   hypercubes are clustered
11 return clusters
```

preferred in general.

Given a cluster configuration C, we design a scoring function, *state preservation index*, which aims to measure how good the configuration is in preserving the system's state transition:

$$spi(C) = \frac{intra\_coeff(C)}{inter\_coeff(C)}, \tag{5.6}$$

For a cluster configuration C, the higher its *spi* value, the better. That is, we prefer clusters with the following properties: compact, densely intra-connected, well separated and sparsely inter-connected.

5.3.2 The Spclust Algorithm

In general, finding the best cluster configuration for a time series (which maximizes *spi*) is intractable. In order to approximate it, we propose Spclust (see Algorithm 5), which is based on a spatial clustering algorithm, Wavecluster [261]. This algorithm detects clusters of arbitrary shape; it is fast, designed for large databases and insensitive to noise. In contrast to Wavecluster, which uses wavelet transformation, Spclust uses a simple step function transformation (see line 3-7).

Then, it expands the resulting clusters to include hypercubes with zero density (see line 10). In Wavecluster, those hypercubes are not clustered since they are translated into noise or white space. In our context, however, we need to partition the space of measurements, as we might encounter these values in the future. To this end, we borrow the idea from maximum margin classifiers to enlarge the resulting clusters incrementally until all hypercubes are clustered.

Next, from a set of possible grid size, $\mathscr{G}$, and threshold, $\mathscr{T}$, we select $G^* \in \mathscr{G}$ and $T^* \in \mathscr{T}$ that maximize the *spi* value of the resulting configuration. Finally, we transform each measurement in the n-dimensional time series S with its cluster labels and obtain a symbolic time series $\widehat{S}$. We define a symbolic time series formally as follows.

Definition 2. *Let $\mathscr{A}$ be our alphabet or a set of symbols. We define a* symbolic time series *as $\widehat{S} = \{\widehat{s}_1, \widehat{s}_2, \widehat{s}_3, \ldots\}$, where each $\widehat{s}_i \in \widehat{S}$ is a tuple (t_i, a_i), t_i is a timestamp, and $a_i \in \mathscr{A}$ is a symbol. Additionally, t_i is earlier than t_j iff $i < j$.*

5.3.3 Run Length Encoding

The first processing step, symbolization, as explained in the previous section, generates a time series of symbols. These can be seen as representing a certain low level state of the measured system. As these symbols are taken from a small set, they have a tendency to repeat themselves. To improve storage space and access efficiency, we apply a run length encoding compression algorithm over symbolic time series. As our input time series contains only timestamp-value tuples, we assume that a measurement stays valid until the next one. However, special care was taken, in the experiments, so that no huge gap (sensor down-time) was filled this way.

Definition 3. *Let $\{\widehat{s}_1, \widehat{s}_2, \widehat{s}_3, \ldots\}$ be a symbolic time series, where each element is a tuple (t_i, a_i), as defined earlier. The output of RLE is a sequence of $\{\bar{s}_1, \bar{s}_2, \bar{s}_3, \ldots\}$, with each element being a triple $(t^b_j, t^e_j, \bar{a}_j)$ which is defined as the shortest sequence of triples such that:*

$$RLE: (\mathbb{R}, \mathbb{R})^n \mapsto (\mathbb{R}, \mathbb{R}, \mathbb{R})^m$$

$$RLE(\{\widehat{s}_1, \widehat{s}_2, \widehat{s}_3, \ldots\}) := \{\bar{s}_1, \bar{s}_2, \bar{s}_3, \ldots\}$$

where $\forall i \in [1, n], \forall j \in [1, m]$, if $t_i \in [t^b_j, t^e_j)$, then $\bar{a}_j = a_i$.

This lossless process transforms a sequence of symbols into a single triple composed of a starting time t^b, ending time t^e and a symbol. The semantic being that the state of the measured system is represented by the symbol from the time t^b (included) to the time t^e (excluded). Hereafter, unless stated otherwise, we represent time series as RLE compressed triples.

5.4 The StateFinder Algorithm

5.4.1 Overview

We define a *state* as a consecutively repeating pattern of symbols that has several (significant) occurrences along the time series. In previous works, the term *pattern* can be found termed as *primitive shape* [62], *frequent temporal pattern* [114] and *motif* [53], and what we call *states* is more similar to the definition of the *operating modes* [139] or *process states* [271] (illustrated in Figure 5.2). Finding states at various scales in an online manner, however, is not trivial. Additionally, the complete list of system states might be unknown beforehand. Thus, the solution must find the states in an unsupervised manner. However, current solutions in the

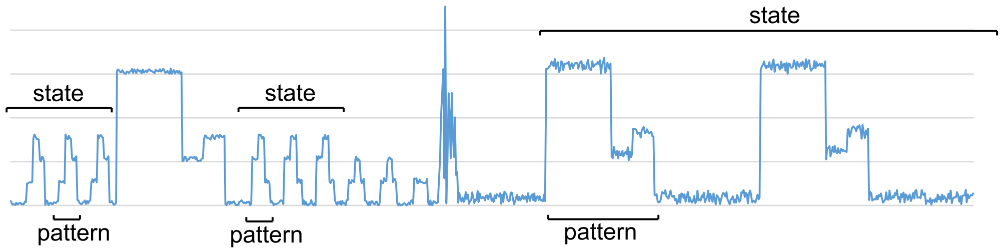


Figure 5.2: Illustration: several consecutive occurrences of a pattern indicate that the system is in a certain state.

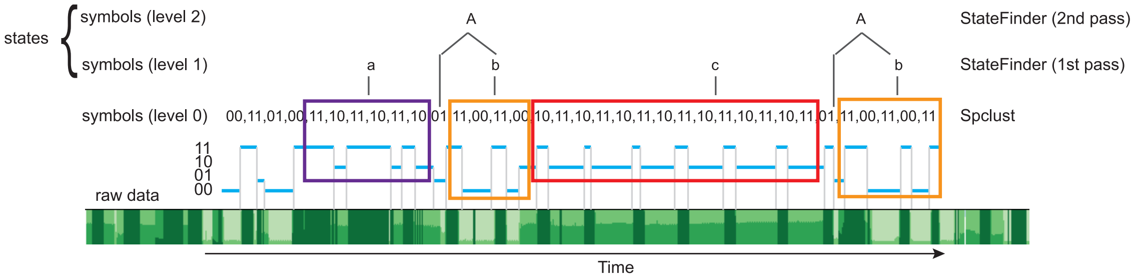


Figure 5.3: We apply the Spclust algorithm to the raw data (sensor measurements) to produce the symbolic time series. After applying RLE to the symbolic time series (not visible in this figure), we apply StateFinder to find the states. The StateFinder algorithm can be applied several times to build a multi-level state representation. The raw data is displayed using the horizon graph [106].

literature (see Section 5.2) aim to find either some predefined patterns or patterns (which can also be extended into states) of a fixed length only.

Our goal here is to find states of various lengths in an online and unsupervised way, and replace them with new symbols (hence reducing the overall data size, see Figure 5.3 for an illustration), while allowing various inferences to be performed on top of the symbolic-state representation (see application examples in Section 5.5 and 5.6). Additionally, more complex patterns can be found by applying multi-pass StateFinder, e.g., 1^{st} pass of the StateFinder is applied to RLE triples, 2^{nd} pass of the StateFinder is applied to the outcome of the 1^{st} pass, and so on.

The StateFinder algorithm consists of two steps: (1) identifying subsequences with a repeating pattern, (2) clustering relevant subsequences and assigning a (new) symbol to them. We explain the steps in more detail below.

5.4.2 Segmentation

One straightforward option is to take all possible subsequences and cluster them to directly separate the repeating ones from others, seen as noise. However, it has been shown by Keogh et al. [131] that this approach does not actually work and produces meaningless output.

Therefore, we first need to isolate the potential subsequences that show a regular pattern. Since the patterns that we are interested in are characterized not by their length, but by their dynamics, we use a Markov model as the basis of our approach below.

We use a first-order Markov chain, but higher orders can also be used to identify more complex states. In our case, the use of a first-order Markov chain as a model for the underlying process is a low-cost and generic approximation and does not imply that the process itself has such properties. If it is known that the time series' processes are of higher order and enough computational resources are available, then it is possible to use higher order Markov chains. But the improvement will only be visible for complex patterns.

Length-frequency set. For each symbol transition, we define an iteratively updated frequency of the next symbol length.

Definition 4. *Let* $\bar{a}_{u-1}$ *and* $\bar{a}_u$ *be the symbol of the* $u-1^{th}$ *and the* u^{th} *entry of the RLE represented input. Additionally, let* $\lambda \in [0,1)$ *be a fading factor. For* $j \in \mathscr{A}$ *and* $k \in \mathbb{N}$*, we denote the (decaying)* length-frequency *of symbol* j *with length* k *following symbol* i*, as* $h^k_{i,j}$*, and update it over time:*

$$h^k_{i,j}(0) := 0$$

$$h^k_{i,j}(u) := \begin{cases} h^k_{i,j}(u-1)\cdot\lambda + 1 & \text{, if } i = \bar{a}_{u-1} \wedge j = \bar{a}_u \wedge \\ & \qquad k = t^e_u - t^b_u \\ h^k_{i,j}(u-1)\cdot\lambda & \text{, if } i = \bar{a}_{u-1} \\ h^k_{i,j}(u-1) & \text{, otherwise.} \end{cases}$$

Hereafter, we refer to the set of all $h^k_{i,j}$*s as the* length-frequency set.

In the definition above, we use the fading factor, λ, to forget the old entries instead of using the sliding window approach, since the searched patterns could have different lengths. Thus, setting a predefined window length would be problematic, as a too small window would fail to capture long patterns and conversely, a too big window would fail to capture short patterns.

Segments. When the underlying process generating the symbols is in a stationary state, the elements of the length-frequency set tend to be stationary as well, whereas between the states, they may change a lot. In general, the length-frequency set can be used to predict the next symbol (and its length). To give a "smooth" estimation of the likelihood of the u^{th} entry of the RLE triples, we compute $Pr(u)$ the probability that symbol $\bar{a}_u$ of length $t^e_u - t^b_u$ follows $\bar{a}_{u-1}$ using the Gaussian Kernel Density Estimation (KDE). Let $i = \bar{a}_{u-1}$, $j = \bar{a}_u$, and $\mathscr{B}_{i,j}(u) = \{k \mid h^k_{i,j}(u) > 0\}$ be the set of symbol j's lengths, following symbol i, with non-zero (decaying) frequency. Then, we have:

$$Pr(u) := \frac{\sum\limits_{k \in \mathscr{B}_{i,j}(u-1)} \left(h_{i,j}^{k}(u-1) \cdot e^{-\frac{((t_u^e - t_u^s)-k)^2}{2 \cdot \sigma_{i,j}(u-1)^2}} \right)}{\sum\limits_{\alpha \in \mathscr{A}} \left[\sum\limits_{k \in \mathscr{B}_{i,\alpha}(u-1)} h_{i,\alpha}^{k}(u-1) \right]}, \tag{5.7}$$

where $\mathscr{A}$ is the (symbol) alphabet set and $\sigma_{i,j}(u) = 0.9An^{-1/5}$ is the kernel bandwidth computed using the Silverman's rule of thumb [270]. That is, $n = |\mathscr{B}_{i,j}(u)|$ and $A = \min(sd, \frac{IQR}{1.34})$, where *sd* and *IQR* are the standard deviation and the inter-quartile range of the set $\{h_{i,j}^{k}(u) \mid k \in \mathscr{B}_{i,j}(u)\}$, respectively. The gaussian term in the numerator is useful in smoothing the symbol length estimation, while the denominator normalizes it over all symbols (and symbol lengths) that occur after $\bar{a}_{u-1}$. Note that, in the Gaussian KDE above, we omit the usual normalization factor ($\frac{1}{\sigma_{i,j}(u-1)\sqrt{2\pi}}$) so that the maximum value of *Pr*(u) is 1, and is independent from $\sigma_{i,j}$. This allows us to use a single threshold, as defined below. After computing *Pr*(u), we define the prediction error as:

$$\mathscr{E}(u) := 1 - Pr(u). \tag{5.8}$$

Then, a *segment* is a predictable subsequence of RLE triples:

Definition 5. *A subsequence of m contiguous RLE triples starting at the u^{th} entry is a* segment, *denoted as* $\widetilde{S}(u,m) = \{(t_j^b, t_j^e, \bar{a}_j)\}$, *iif* $t_j^e = t_{j+1}^b$ *and* $\mathscr{E}(j) < p$ *for all* $u \leq j \leq u+m-1$, *where* $p \in [0,1]$ *is a threshold parameter. A* maximum segment *is a segment that cannot be lengthened by adding the preceding or following symbols. By definition, maximum segments are non-overlapping.*

Hereafter, whenever the context is clear, we omit the parameters u and m and refer to a segment simply as $\widetilde{S}$. Figure 5.4 shows an example on how segments are formed.

Complexity. To analyze the time complexity of processing each RLE triple, we need to consider both the time to update the length-frequency set (Definition 4) and the time to compute the prediction error (Equation 5.8). Both of them iterate over the set of non-zero length-frequencies, namely, at the u$^{\text{th}}$ entry, we iterate over the set $\mathscr{H}_{\bar{a}_{u-1}}(u) = \{h_{i,j}^{k}(u) \mid k \in \mathscr{B}_{i,j}(u), i = \bar{a}_{u-1}\}$ for $j \in \mathscr{A}$ and $k \in \mathbb{N}$. We refer to this set as $\mathscr{H}$ for brevity. This gives us $\mathscr{O}(|\mathscr{H}|)$ time complexity. Below, we show that $|\mathscr{H}|$ is bounded.

Lemma 1. *Let λ be the fading factor in updating the length-frequency set and ϵ be the smallest representable positive value greater than zero (either predefined or limited by machine precision), i.e., for a value v where $0 \leq v \leq \epsilon$, we have $v = 0$. Then, $|\mathscr{H}| \leq \log_{\lambda} \epsilon$.*

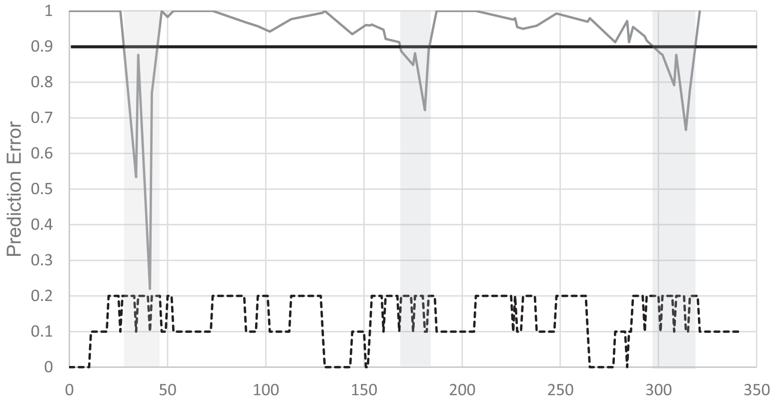


Figure 5.4: Illustration on how segments are formed, using the data from the bird call dataset in [235]. The dashed line shows the (three) symbols at level 0, plotted as values 0.0, 0.1, and 0.2, the thin solid line is the prediction error over time, and the thick solid line is the threshold parameter $p = 0.9$. The gray areas show the segments.

Proof. Since for a given (StateFinder-) pass the size of the alphabet set is constant, $\mathscr{H}$ can only grow (linearly) when we process an RLE triple by adding a new value for k that has never been seen before for a given transition. In the worst case scenario, each incoming triple has a new length k. If this happens, however, the value of the oldest element in $\mathscr{H}$ would be $\lambda^{|\mathscr{H}|}$, since we apply a fading factor when updating the length-frequency set. That is, when $|\mathscr{H}| > \log_\lambda \epsilon$, the oldest value would be less than ϵ, thus keeping $|\mathscr{H}| \leq \log_\lambda \epsilon$. □

Therefore, we have $\mathscr{O}(\log_\lambda \epsilon)$ time complexity, which is essentially $\mathscr{O}(1)$, since ϵ is a constant. Regarding the space complexity, at any u$^{\text{th}}$ entry, following Lemma 1, we need to store $|\mathscr{H}_i(u)|$ entries for each $i \in \mathscr{A}$. Thus, we have space complexity $\mathscr{O}(|\mathscr{A}| \cdot \log_\lambda \epsilon)$, which is also $\mathscr{O}(1)$.

5.4.3 Clustering

Following the definition of segments, we represent a segment as a transition matrix of its symbols, i.e., the Segment Transition Matrix.

Definition 6. *The* Segment Transition Matrix *of segment $\widetilde{S}$ is an $n \times n$ matrix $M(\widetilde{S}) = \{m_{i,j}(\widetilde{S})\}$, where $n = |\mathscr{A}|$ is the size of the alphabet and $m_{i,j}(\widetilde{S})$ is defined as follows, using x as a temporary variable for better readability:*

$$x_{i,j}(\widetilde{S}) := \sum_{(t_k^b, t_k^e, \bar{a}_k) \in \widetilde{S}} \begin{cases} t_k^e - t_k^b - 1 & , \textit{if } i = j = \bar{a}_k \\ 1 & , \textit{if } i = \bar{a}_k \wedge j = \bar{a}_{k+1} \\ 0 & , \textit{otherwise} \end{cases}$$

$$m_{i,j}(\widetilde{S}) := \frac{x_{i,j}(\widetilde{S})}{\sum_{q=0}^{n} x_{i,q}(\widetilde{S})}$$

The row i, column j of the resulting matrix contains the probability to have a transition from symbol i to j, the diagonal being the probability to stay on that symbol. As our notation excludes the ending time (t_k^e) from the interval, we need to subtract 1 in the top case.

To select which maximum segments are relevant and will be replaced by new symbols, we group them into clusters. The clustering algorithm should: (1) be able to run online (as in the segmentation step), thus limiting its computational needs, (2) make no assumption about the number of clusters in advance, as it may increase over time, (3) be consistent, i.e., once a segment is assigned to a cluster, it should not change its assignation in future iterations. To satisfy all the requirements above, we adapt the clustering algorithm described in [139] by computing the distance between two segments $\widetilde{S}^a$ and $\widetilde{S}^b$, using the cosine distance of their vectorized transition matrices:

$$\delta(\widetilde{S}^a, \widetilde{S}^b) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} m_{i,j}(\widetilde{S}^a) \cdot m_{i,j}(\widetilde{S}^b)}{\sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} (m_{i,j}(\widetilde{S}^a))^2} \cdot \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} (m_{i,j}(\widetilde{S}^b))^2}} \tag{5.9}$$

Note that, this distance measure does not need to know the ordering of symbols and distance between them, which are undefined in most cases.

When a new segment is identified, the algorithm assigns it to the nearest cluster within a maximum distance $d_{max} \in [0, 1]$. If there is no cluster within d_{max}, then a new cluster for the segment is created. Each cluster is associated with a distinct "higher level" symbol (see also Figure 5.3). If a segment is associated with an already existing cluster (within d_{max}), it is replaced with a new (higher level) RLE triples as follows. Let $\widetilde{S}(u, m)$ be the segment to be replaced and Δ be the symbol of the cluster associated with the segment $\widetilde{S}(u, m)$. Then, we replace $\widetilde{S}(u, m)$ with a single triple $(t_u^b, t_{u+m-1}^e, \Delta)$. Finally, we call the clustered segment a *state*.

5.4.4 Improving segmentation accuracy

As Figure 5.4 shows, there is a delay before a state is detected. This is due to the history implicitly contained in the *length-frequency set*, which influences the prediction-error computation. Thus, only after a few occurrences of the pattern, does the prediction error pass below the threshold. If we are able to keep in memory a short sliding window of the previously seen data points, then it is possible to look back to when the beginning of a segment is detected, in order to identify the real start of the segment. This is done by recomputing the prediction errors backwards from the start of the segment, with the transition matrix computed from the current symbols in the segment. For a fully online system, this would add the fixed delay of the length of the added window. So this is a trade-off between delay and accuracy, which can be specific to the application.

Algorithm 6: StateFinder

```
Input: $\bar{s} = (t^b, t^e, \bar{a})$                      //RLE compressed symbol
Output: $\emptyset$ | $s$ | $[s_1, s_2, ...]$          //nothing, one symbol or several symbols
Global: $H \leftarrow \emptyset$                          //the length-frequency set, initially empty
        $B \leftarrow \emptyset$                          //a buffer, initially empty
        $C \leftarrow \emptyset$                          //the clusters, initially empty
        $last\_symbol \leftarrow \bar{s}$   //last processed symbol, initially the first symbol
e $\leftarrow 1 - Pr(\bar{s}|H)$;                            //from Formula 5.7
r $\leftarrow \emptyset$;                                    //by default return nothing
update($H|last\_symbol, \bar{s}$);                           //from Definition 4
if $e \leq threshold$ then
    B.append($\bar{s}$);
else if $|B| < min\_size$ then
    $r \leftarrow B$;
    $B \leftarrow \emptyset$ ;                               //flush the buffer
else
    $m \leftarrow$ to_matrix($B$);                           //from Definition 6
    /* get the nearest cluster if it exists within the defined radius */
    $c \leftarrow$ find_cluster($C, m$);
    update($C|m$);
    if $c$ is not null then
        $r \leftarrow c$;
    else
        $r \leftarrow B$;
        $B \leftarrow \emptyset$ ;                           //flush the buffer
$last\_symbol \leftarrow \bar{s}$;
return $r$;
```

5.4.5 Reverting States

One of the goals of Spclust and StateFinder is to produce a symbolic time series to support higher level applications, without converting it back to the sensor's original measurement values. Since a symbolic time series is much shorter than its original version, this property is desirable, especially due to the limited sensor storage and computational power. However, for the sake of completeness, we illustrate below how one can revert a symbolic time series back to its original values.

Converting symbols (level 0) generated by Spclust to its original values In this case, we can simply use the cluster centroids to approximate the original values of the symbols. Note that, using Spclust, we have a one-to-one mapping between clusters and symbols.

Converting RLE compressed triples to non-compressed triples. Each symbol is repeated n times with $n = (t_e - t_s)/r$, where r is the sampling rate of the original data. As sensors may have a variable sampling rate or some gaps, this transformation could produce more/less triples than the original time series.

Converting symbols from level 1 or higher to one level lower. The main idea, is to use the Segment Transition Matrix representing the cluster associated to this symbol and follow its transitions. To find the starting symbol in the case where this one is not available, we use an heuristic that aims to find a symbol that has the lowest incoming transition probability and a positive outgoing transition probability. Formally, we take the symbol i where the sum of the elements in the ith row, except the diagonal, is greater than 0 and the sum of the elements on the ith column, except the diagonal, is minimal. Then according to the transition probability to the next symbol, we build a sequence. Even though we might produce a slightly different sequence from the original one, it will eventually have the same symbol distribution.

5.5 Applications

In this section, we show how various higher level applications can be built on top of our state representation without converting it back to the original values, i.e., state forecasting, anomaly detection, and compression. Additionally, since state recognition is a “natural” capability of StateFinder, we refer the readers directly to Section 5.6.2.

5.5.1 State Anomaly Detection

As the normal behavior of the system (i.e., things sensed by the sensors, such as human or natural phenomena) is characterized by some stability, it would be converted to higher level symbols (or states) by the StateFinder algorithm. An anomaly, however, would not be converted and its symbols preserved through the system. In other words, a capability to detect anomalies is also implicitly built into the symbolic conversion. This makes the anomaly detection task using our symbolic representation rather straight-forward, without reverting them to their original values.

Additionally, RLE triple representation and StateFinder also make the whole process more efficient since the normal behavior is now much shorter (see also Figure 5.3 for an illustration).

To detect anomalies, we adapt the algorithm from Wei et al. [297], where the authors use the distribution of symbol occurrences in the whole training set (in the supervised case), or in the previous time window (in the unsupervised case) as the normal reference. Unfortunately, this does not work in our case, since human behavior is daily periodical. Thus, instead of using the whole training set or only the previous time window, we use the behavior of the same time of day/week (in the training set) as the normal reference. In addition, the Segment Transition Matrix implicitly contains the symbol distributions and thus can be directly used for estimating them.

More formally, let r be a time duration (e.g. 15, 30, or 60 minutes), as our detection resolution.

We assume that the system of interest inherits daily periodical behavior. We then divide the a day into time intervals $\{R_1,\ldots R_n\}$, each of length r. Then, we compute, over the entire training set, $\{D_1,\ldots D_n\}$, the distribution of symbol lengths which occur within each time interval, where D_i is the symbol length distribution within R_i. We also compute $\{D'_1,\ldots,D'_n\}$, the distribution of symbol lengths in the test set for each time interval. Next, the anomaly score of time interval R_i in the test set is computed as the dissimilarity between D'_i and D_i. The less similar D'_i is to D_i, then the more likely R_i in the test set contains anomalous behavior. Moreover, setting r differently allows us to see anomalies with a different granularity.

5.5.2 State Forecasting

The algorithm is inspired by the pattern-based forecasting in [187, 194, 262]. It consists of three steps: clustering, pattern similarity search, and prediction. Below we give an overview of the algorithm. Let us assume that we have a symbolic time series $\widehat{S} = \{\widehat{s}_1,\ldots,\widehat{s}_t\}$ available as the training set, a forecast horizon h, and window length w. Our goal is to forecast $\widehat{S}^* = \{\widehat{s}_{t+1},\ldots,\widehat{s}_{t+h}\}$, i.e., sensor values up to h time periods following $\widehat{S}$.

Clustering Divide $\widehat{S}$ into a set of symbolic time series, $\mathscr{S} = [\widehat{S}_1,\ldots\widehat{S}_l]$, where each $\widehat{S}_i \in \mathscr{S}$ has length h, and if $i < j$, then $\widehat{S}_i$ is a series preceding $\widehat{S}_j$. Cluster the symbolic time series in $\mathscr{S}$, and let $c(\widehat{S}_i)$ be the cluster label of $\widehat{S}_i$.

Pattern similarity search Find all *matching* sequences of length w, $\{\widehat{S}_i,\ldots,\widehat{S}_{i+w-1}\}$, where $i \le l-w$ and $[c(\widehat{S}_i),\ldots,c(\widehat{S}_{i+w-1})] = [c(\widehat{S}_{l-w+1}),\ldots,c(\widehat{S}_l)]$.

Prediction Let $\{\widehat{S}_{i_1+w},\ldots,\widehat{S}_{i_k+w}\}$ be the *predictor set*, i.e., the set of series following the matching sequences. The predicted series, $\widehat{S}^*$, is obtained by aggregating the predictor set.

For the clustering step, we used the KMedoids algorithm. Note that, any other unsupervised learning techniques can also be used here. As in other unsupervised learning techniques, however, given different parameter settings, we are often uncertain which setting delivers the best cluster configuration. This holds even if the parameters are very intuitive, such as k in KMedoids, which is the number of clusters to be created. To select the best configuration, we use similar techniques to [187, 302]. We perform the clustering step several times using different configurations, i.e. different k, and evaluate the resulting configurations using the Silhouette, Dunn and Davies-Bouldin indices. The best cluster configuration is chosen by majority voting over the three indices.

5.5.3 Data Compression

Because symbolization is a kind of time series summarization technique, StateFinder can also be used for compressing the time series. By adapting the parameters at each iteration, it is possible to trade the accuracy of the high level symbols with a better compression. Compressing a completely random time series, however, would not work with this method.

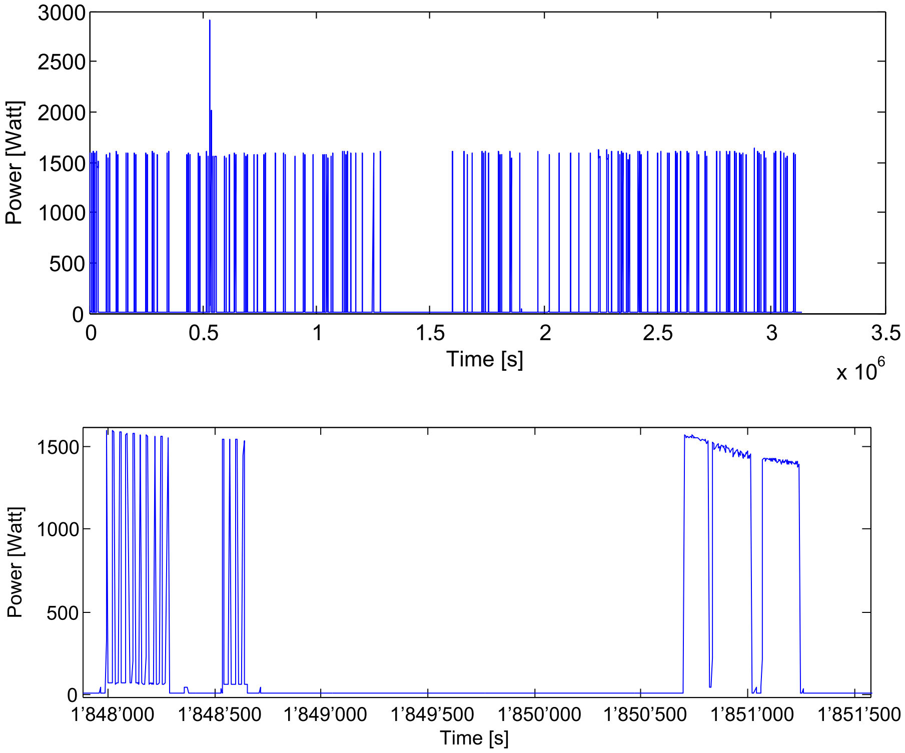


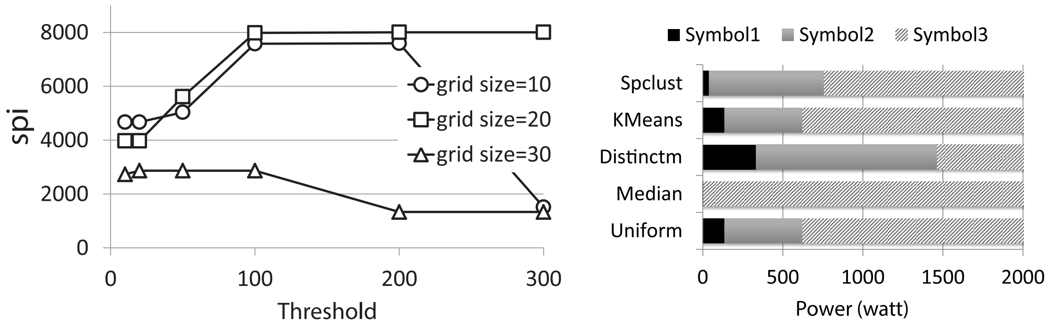
Figure 5.5: Complete microwave dataset, showing one month of usage (top) and one hour of microwave oven usage (bottom), showing different power levels.

5.6 Experiments

In this section, we demonstrate the applications of our approach to real-world datasets. To show the versatility of StateFinder, we deliberately use datasets from various domains, such as household energy usage, human activity recognition, and location tracking. Unless stated otherwise, we use the fading factor $\lambda = 0.9$ to update the length-frequency set (Definition 4). According to Papadimitriou et al. [212], a value around 0.96 is reasonable for a stationary time series, however as we are dealing with switching between several stationary states, a lower value is preferred.

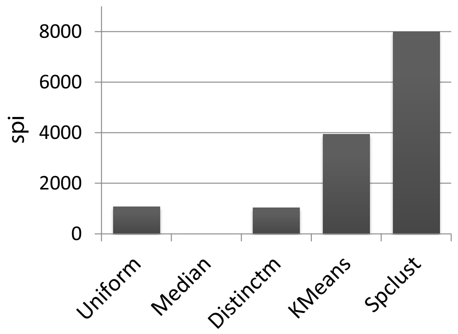
5.6.1 State Preservation Index

In this experiment, we use the REDD dataset [140], containing smart meter readings for 6 houses, divided by appliances, with a 3-second sampling rate. We show here the result from channel 11 of house 1 which represents the microwave oven, since (compared to others) it contains more heterogeneous states and has fewer missing values. The time series plotted in Figure 5.5a has 745,878 values, representing one month of data. Figure 5.5b represents one hour of this same time series. On the left of the Figure, when the microwave oven is not set on full power, as it is on the right, there are distinguishable heating cycles.



(a) State preservation indices of Spclust using different grid size and threshold The higher the better.

(b) Power to symbol mapping of different methods.



(c) State preservation indices of different methods. The higher the better.

Figure 5.6: Evaluation of symbolic representation of the microwave dataset. In (a) the highest eval value is achieved using grid size=20 and threshold=100, 200, or 300. Next, for (b) and (c), we set Spclust grid size to 20 and threshold to 200. The alphabet size of uniform, median, and distinct median is set to 3. For KMeans, we set k=3.

Figure 5.6 shows the *state preservation indices* (or spi, see also Equation 5.6) of different symbolization methods. For Spclust, Figure 5.6a shows that the highest state preservation index is obtained using grid size=20 and thresholds equal to 100, 200, or 300. In addition, we also compare the symbolization results from Spclust using other symbolization methods, i.e uniform, median, distinct median [301] and KMeans clustering. Figure 5.6b shows how Spclust and the other symbolization methods map the microwave power usage into symbols. In this case, Symbol 1 can be interpreted as encoding the *off* or *standby* states, Symbol 3 encodes the *on* state, whereas Symbol 2 encodes the *intermediary* state. Spclust appears to be the most suitable method to perform the symbolization since it has the highest spi value (see Figure 5.6c).

Driven by the fact that Spclust significantly outperforms other methods, while having rather similar power-to-symbol maps, we carefully analyzed the mapping and the dataset. There are some small power usages around 30 watts which are put in the off state by almost all methods (except median). This power reading appears only once in a while, and in a short period (3 or 4 seconds) between the off states. Thus, it makes sense to assume that it indeed belongs to the off state. We also observe that there are some intermediary power usages around 60 watts. Unlike the 30 watt power usages, these 60-watt power usages last longer (around 20 or 30 seconds) and systematically between the on states (see e.g., Figure 5.7A). While other methods include these usages in Symbol 1 (off or standby states, which is not the case), Spclust includes them in Symbol 2 (intermediary state, which confirms our insight). After consulting our colleagues from the Electrical Engineering department, they confirmed that Figure 5.7A indeed shows how a microwave works when it is set to a *medium* power. Rather than emit the exact medium power, it emits high power intermittently (proportional to the power setting). The 60-watt power usages are used for lighting, rotating the plate, and (possibly) turning on its fan.

5.6.2 State Recognition

As StateFinder is designed to extract states in the input data, *state recognition* is the most direct use of its output. The resulting states can then be mapped to a specific semantic for example, using classification if one already has a model. For instance, we show below how StateFinder is able to identify different heating levels of a microwave oven.

Using Spclust grid size=20 and threshold=200, we extracted the states using StateFinder on the same dataset as in Section 5.6.1. Figure 5.7 shows the prototypes of each state found, by reverting the transition matrix to raw power value. The four heating levels that were used during the experiment, are noticeable as the states A, B, C and D, while the other states show different microwave usage patterns. Overall, Figure 5.7 also shows that StateFinder is able to identify states of various length or time scales (e.g., seconds, minutes, hourly). Note that, the capability to automatically discover appliances' states in an unsupervised way is crucial, especially since there are countless new electrical appliances released every day. For instance, a good smart home or energy management system should be able to learn and recognize the states of an appliance that is newly bought by the home owner. In this case, supervised learning might not work very well, since it would fail to recognize states that it has never seen before.

In addition, our algorithm is not specific to one domain, but rather to a certain definition of the searched states. In the next experiment, we illustrate another use-case involving mobile phone accelerometer data. We use the dataset provided on the UCI Machine Learning Repository by Anguita et al. [11].

The data of user 6 is particularly interesting since the patterns are clearly distinguishable by eye, and therefore make it possible to validate the output of our algorithm. In the Figure 5.8 (top) we show the tBodyGyro-mean()-X feature which, as its named implies, represents the X axis of the gyroscope with the 50Hz data being smoothed over a sliding window of 128 readings. The data is first symbolized using 3 symbols and then given to the StateFinder. Different darkness levels indicate the different symbols: 0 to 2 for the symbol level 0, and 3 to 4 for the symbol level 1, which represent the states generated by the StateFinder. To better

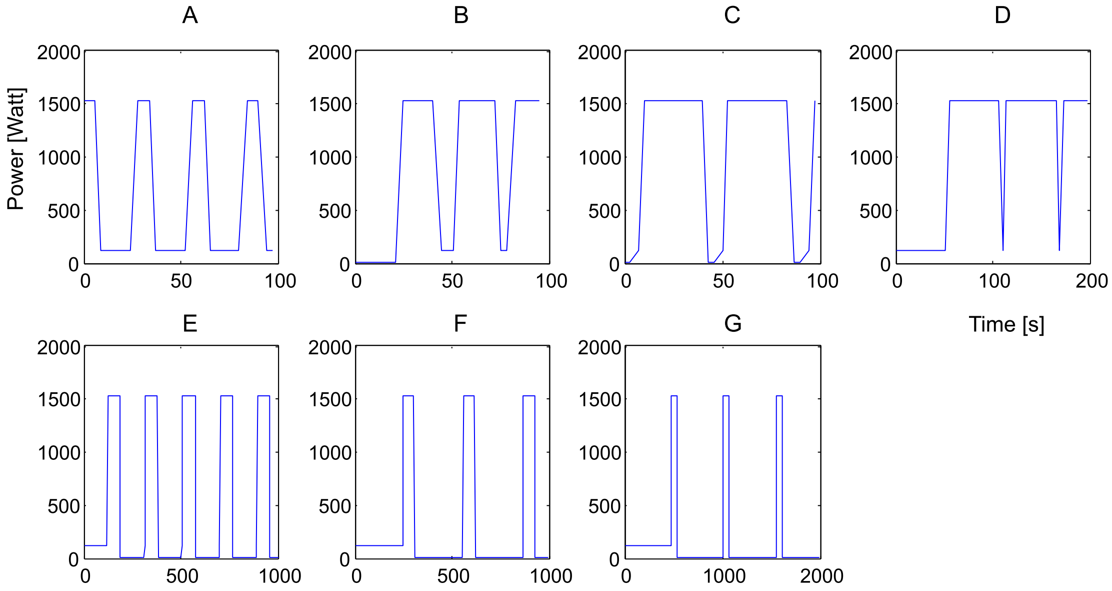


Figure 5.7: Prototypes of the states, generated from the transition matrices of the microwave time series in Figure 5.5a.

understand the state identification, Figure 5.8 (bottom) shows on the y-axis symbols level 0.

The states found can be matched to the user's activity. From the ground truth, provided with the dataset, we are able to associate a semantic with the two identified states, namely symbols 3 and 4. The first is "going upstairs" while the second one is "going downstairs". As other labels like "sitting" or "standing" are not linked to any specific pattern on this axis of the gyroscope, no state is linked to them.

5.6.3 State Forecasting

For this experiment, we use GPS traces from the Nokia Lausanne Data Collection Campaign dataset [155]. The ten users having the largest number of location records during the first 6 months of the study were selected to minimize the gaps in the resulting time series and to avoid the users who were not carrying their phones with them all the time. In this experiment, we aim for next-hour forecasting, i.e. h is 1 hour. We compute the prediction result, $\widehat{S}^*$, as the medoid of the predictor set.

The symbolization is achieved through Spclust and results in 9 to 34 symbols, depending on the mobility of the users. The first level symbols mostly represent the places visited by the users, dividing the regions like a Voronoi diagram. On average, 24% of the symbols at level 0 are converted to symbols at level 1 (states), which can be seen as the routes taken by the users in their daily routines. Pattern-based forecasting is evaluated by taking the first 80% of the dataset as a training set (from 0 to t) and testing it on the next hour $(t+1)$. Then

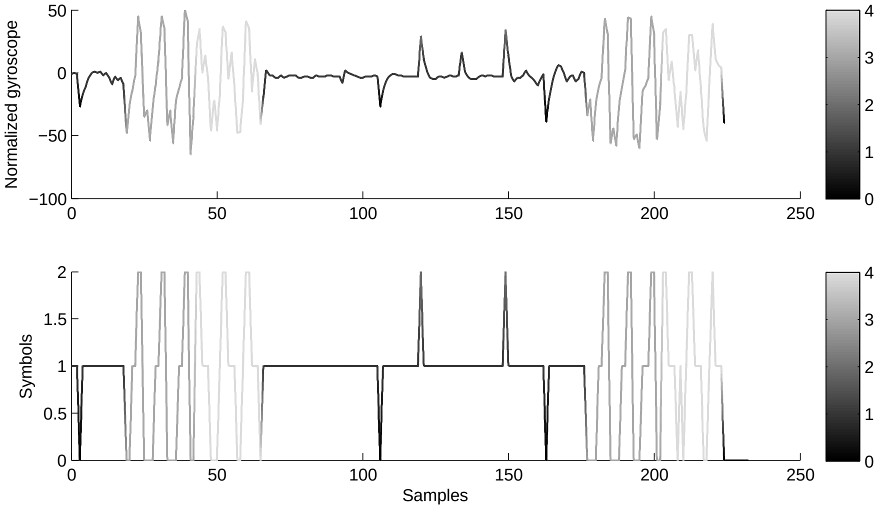


Figure 5.8: Gyroscope value on x-axis, recorded on a smartphone while the user was performing different activities. Below is the symbolized version using 3 symbols and the color indicates the state detected by the StateFinder.

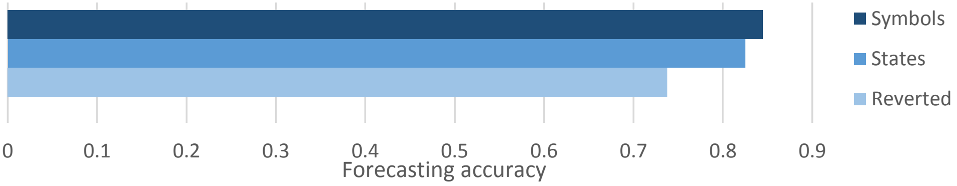


Figure 5.9: Comparison of the 1-hour forecasting accuracy using both data representations, symbols (level 0) and states.

we repeat the experiment by forecasting $t+2$ after including $t+1$ to the training set. We run the forecasting algorithm at different symbolization levels and for each of them, the forecasting accuracies are averaged over all users and iterations. Figure 5.9 shows the accuracy of forecasting before and after running StateFinder. Using the time series of states (symbols level 1) allows predictions that are as good as with the symbols (level 0), showing that the states are consistently found and replaced by higher level symbols. As the reverting of the sates (using the Segment Transition Matrix) also introduces some error, the third bar illustrates the accuracy when comparing the reverted predicted values to the original time series, but this result depends a lot on the reverting function which still has room for improvement.

The running time for the forecasting algorithm, shown on Figure 5.10, is strongly related to the data size, thus giving advantage to more compressed data representation (higher level symbols). Additionally, the time for computing RLE and the StateFinder are some orders of magnitude lower, making them good candidates to be run on low power devices.

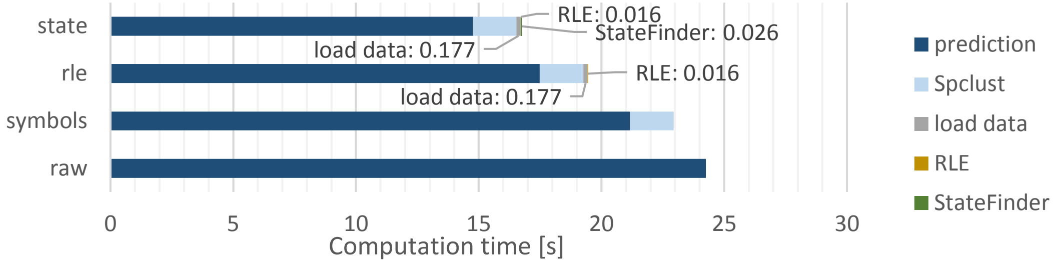


Figure 5.10: Timing for the pattern-based forecasting algorithm, on top of different data, and the processing steps for generating those data.

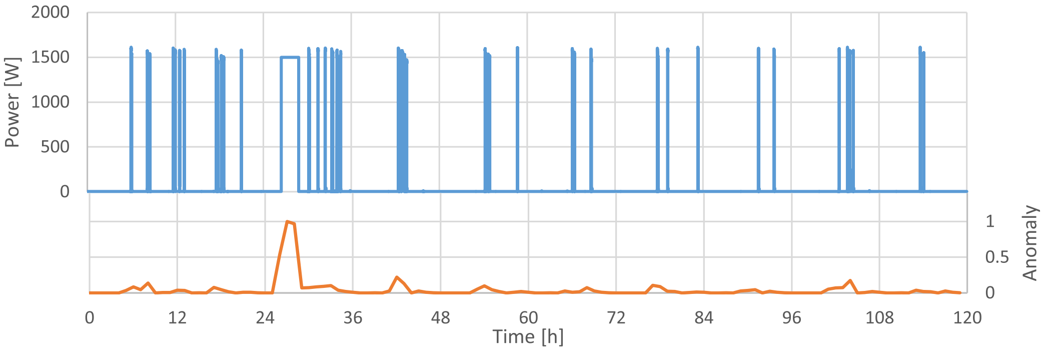


Figure 5.11: Anomaly detection on states and symbols generated from the microwave dataset. The top plot shows the raw data including the anomaly around hour 30. The bottom plot shows the anomaly probability computed from the the StateFinder output symbols.

5.6.4 State Anomaly Detection

We run our anomaly detection algorithms on symbols at level 1, using the same smart-meter dataset and symbolization process as in Section 5.6.2. For this experiment, we add a synthetic anomaly, as if the microwave oven was left running for around 2 hours. In Figure 5.11, the top plot shows the raw data where the anomaly was added in the first quarter. The bottom plot, which shows the output of our anomaly detection algorithm that indicates the probability of an anomaly happening at any given time, clearly shows a peak at the time when the anomaly happened.

5.6.5 Running Time

To assess the impact of the time series size and show that StateFinder can be run on infinite stream, we measured the time needed to process time series of various lengths on a server with a CPU @ 2.30GHz. We used the same datasets as for the forecasting experiment (GPS traces) and the microwave power from the state identification experiment. The third one is a random time series where symbols (between 0 and 7) and their duration (between 1 and 1000) where chosen uniformly at random. Figure 5.12 shows the processing time of StateFinder with

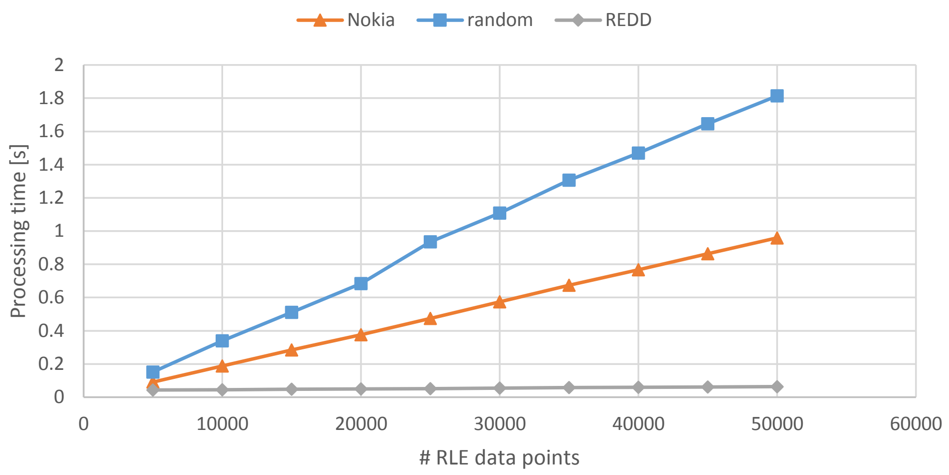


Figure 5.12: Processing time for running StateFinder on different datasets and dataset sizes.

regard to the size of the time series. Time series that were too short were repeated to match the length of the largest. It clearly shows that the processing time grows linearly with the data size, thus having a constant processing time per element. The different slopes can be explained by the number of symbols and the randomness of the data. The larger the number of different symbols in the time series, the more time it will take to process, but the more predictable and repetitive they are, the faster StateFinder will be. In this case, with only 3 different symbols and a lot of identifiable states, the microwave power time series allows a very fast processing, below 0.1 seconds for 50’000 values. Whereas a completely random time series with 8 symbols is even slower than the GPS trace time series with 64 symbols.

5.7 Conclusion

In this chapter, we presented an online and unsupervised state recognition algorithm, namely *StateFinder*, that is able to detect and identify states in symbolic time series. Since initial symbolization determines StateFinder performances, we proposed a *state preservation index* to assess how good a symbolic representation is at preserving systems’ state transitions. We also proposed the *Spclust* algorithm to build the initial symbolization, also for multidimensional time-series. In addition, learning on top of higher level symbols is faster, requires much less space, and produces comparable accuracy. It also enable us to perform analytics locally on smart devices and appliances, which have limited computational and storage capabilities. However, further refinement should be done for non-stationary time series. Additionally, since Spclust and StateFinder preserve interesting events and reduce data size, they could be used to complement Round Robin Database, i.e., to store data further in the past given the same amount of space available.

6 Application example: Air pollution exposure estimation for personalized health

6.1 Introduction

This chapter presents a concrete example of a context-based application that was developed throughout my thesis. It is a real world example of an end-to-end implementation of a mobile sensing application fusing data from various sources.

According to the World Health Organization[1], indoor and outdoor air pollution causes 7 million premature deaths annually worldwide, the large majority being due to cardiovascular diseases. The pollutants considered to create a risk by the European Environment Agency[2] are carbon monoxide (CO), Nitrogen dioxide(NO_2), Sulfur dioxide (SO_2), Ozone (O_3) and Particulate Matter (PM). Except for the latter, all these pollutants are found in a gaseous state at room temperature and cannot be seen with the naked eye. PM can also go unnoticed at low concentrations, but usually its sediments can be seen [30]. Therefore, to enable people to sense the invisible and to detect hazardous concentrations, artificial sensors are needed.

The most noticeable effects of air pollution on health come from inhalation. Therefore, an estimation of the exposure must take into account the breathing rate and volume. These can be monitored using specialized sensor belts, on the chest, or approximated by identifying the user's activity. Indeed, the intensity of the activity is directly related to the calories burned and the oxygen needs, acquired through breathing [125]. Moreover the immediate breathing microenvironment, or context, needs to be taken into consideration. While being at the same location, studies have shown that someone in a car with closed windows is less exposed to fine particles than someone biking [33].

The goal of this application is to provide the users with an estimation of the amount of air pollutant they have been exposed to during the time they were carrying the device. It has been shown by several studies that even limited exposure to certain air pollutants can have

---

[1] http://www.who.int/mediacentre/news/releases/2014/air-pollution/en/

[2] http://www.eea.europa.eu/publications/environmental_monograph_2006_5

noticeable effect on someone's health[284, 285]. Therefore, this application can be used as a diagnostic helper, by providing the doctors with more accurate information on their patients' environment, or in a preventive fashion, helping the users to understand if some of their habits could be changed to reduce their exposure.

To this purpose, we use the contextual information from a personal device, such as its location, activity and - when available - environmental parameters, and combine them with globally collected data from an aggregation server. More precisely we consider the air quality data collected by sensing nodes installed on the roofs of buses[3]. The scenario can be expanded by also propagating back information sensed by the mobile devices to the aggregation server, opening the door to all the community sensing challenges, such as privacy and trust management.

Our application comprises of two components: the personal devices and an aggregation server. Smartphones, in the role of the personal devices, are in charge of continuously collecting the data from their sensors and also providing an interface for interacting with the user. They are equipped with an external sensor platform for expanding their sensing capabilities. On the server side, the Global Sensor Networks [1] streaming middleware is deployed for managing the data streams from the various sources and maintaining a model of the air quality over the covered region.

6.2 Related Work

6.2.1 Assessing pollution exposure

The usual pollutant concentration values used in epidemiological studies or related exposure estimation projects are either based on dispersion models or using expensive personal monitors, which are typically bulky and need to be processed offline after the study.

In the first category, we find the platform developed by Mun et al. [197], named PIER, Personal Environmental Impact Report. They use a mobile application, recording the location of the user to identify the transportation mode (stationary, walking, driving) and upload this data to an aggregation server. The exposure, but also the carbon footprint, are computed on the server using PM 2.5 models. The CalFit mobile application from de Nazelle et al. [65] also records the GPS traces, but, in addition, uses the accelerometer for classifying the activity. The processing of data and exposure estimation is done offline with a dispersion model for NO_2. In their previous work, the same authors describe a general framework for integrating activity and especially its intensity into the exposure estimation [64], based on the computation of the Metabolic Equivalent of Task (MET) [124]. But this was only using simulations as inputs.

In their review of human exposure to air pollution, Steinle et al. [272] show the transition from static sensors and models toward personal exposure devices in epidemiological studies. They also emphasize the increased usage of GPS traces to complement the time-activity diary. For

---

[3] see the OpenSense project description: http://opensense.epfl.ch

example, Morabia et al. [192] used personal exposure monitors for their studies, combining GPS and the diary to compute the MET for assessing the exposure to fine particles in private cars and public transportation. But the complete replacement of written diaries (sometimes on electronic devices) by automated context identification from mobile devices is not yet topical.

6.2.2 Personal monitoring devices

In the context of the Internet of Things and participatory sensing, some pollution monitoring platforms have been developed, including various air quality sensors. They also include the option for uploading the data to a centralized server and visualizing the traces, but they do not provide a complete exposure estimation that would include the activity of the user as well.

The N-Smart project [113], later continued within the Common Sense project [78], prototyped a Bluetooth extension that could be attached to the back of a smartphone and also an external device for measuring various pollutant concentrations in addition to the deployment of sensors on street sweepers. Built on the MetroSens architecture, Bikenet [80] is a complete sensor deployment on a bike, covering the whole biking experience, including CO_2 concentration. As an example of the Mobsens platform, the PollutionSpy application connects via Bluetooth to a sensor data logger, monitoring CO, CO_2, NO_x and SO_2. The citisense project [16] also prototyped an external Bluetooth sensor box with electrochemical sensors, continuously mapping the measured values with their coordinates, efficiently using the GPS. The TECO Envboard, developed by Budde et al. [37] is an external device that includes a lot of different sensors for air quality (VOC, O_3, CO, NO_x), but also for recognizing the activity (accelerometer, gyroscope) and location with the GPS. The same authors also built a prototype of a dust sensor using the camera and flash of mobile phones, based on the reflection on particulate matter [38]. The Node+ commercial product was used by Devarakonda et al. [69] in their participatory sensing deployment to assess the difference between the carbon monoxide concentration inside and outside a car. Finally, within the OpenIoT project, Podnar Zarko et al. [222] implemented an external sensor board for CO, NO_2 and O_3 to showcase their publish-subscribe back-end infrastructure.

Our work is the follow up on the first OpenSense mobile prototype [103], integrating the pollution sensors with activity and other contextual information. It lies at the intersection of the fields of pollution exposure modeling, usually simulated, and mobile environmental sensing, usually only collecting data. Therefore, it is the first prototype of a complete system that includes an activity-based air pollution exposure estimation as well as an on-device air quality sensor and a large back-end of mobile sensors covering a whole city.

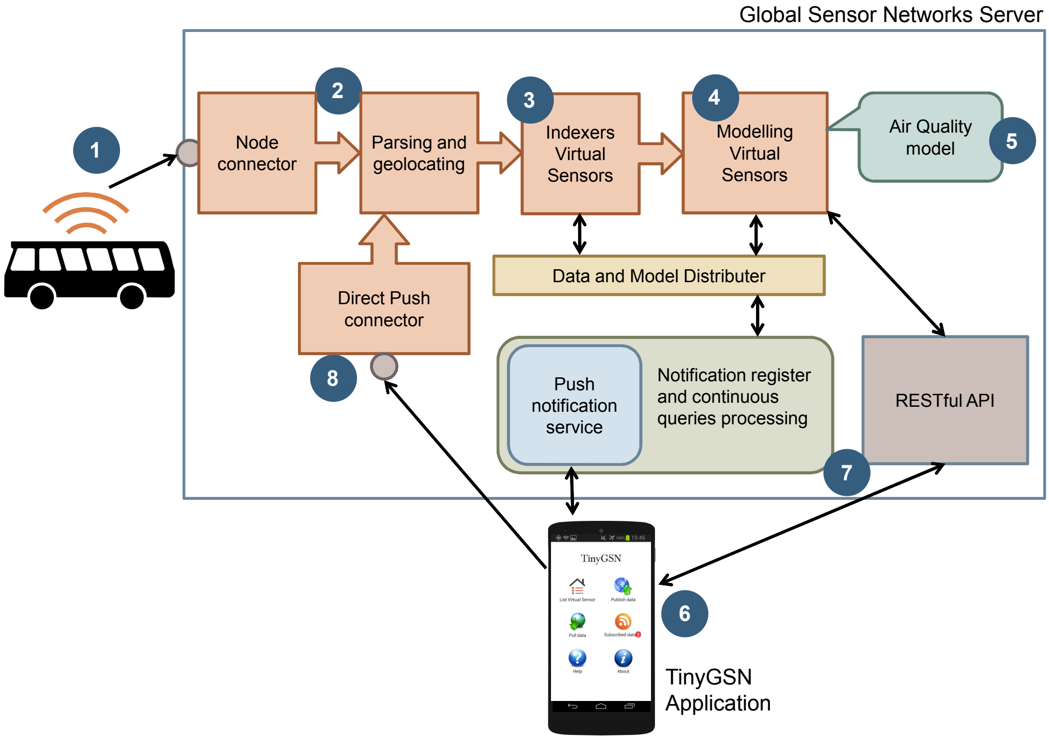

Global Sensor Networks Server
1
2
Node connector
Parsing and geolocating
3
Indexers Virtual Sensors
4
Modelling Virtual Sensors
Air Quality model
5
Direct Push connector
Data and Model Distributer
8
Push notification service
Notification register and continuous queries processing
RESTful API
7
TinyGSN
6
TinyGSN Application

Figure 6.1: System Block Diagram

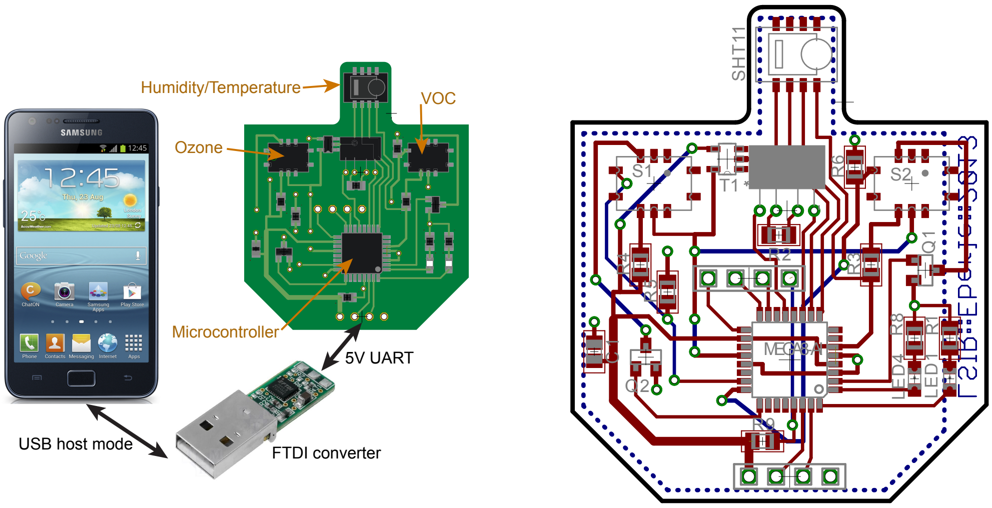
6.3 System architecture

The general architecture of our system is shown in Figure 6.1. The main data source for the air pollution concentration is the OpenSense Lausanne deployment (1). Around ten buses are equipped with a sensor box on their roof and regularly send their measurements to the GSN back-end, through the GSM network. The measurements from the various sensors (CO_2, CO, NO_2, PM) have their own data streams and therefore have to be joined with the GPS stream to geo-locate them (2). The stream join is performed by mapping the pollution sensor streams onto the GPS one, interpolating on the location. Then, a Virtual Sensor takes care of computing the pollution indexes (3), either by thresholding with the official scales or with regards to a relative scale compared to a reference year. Finally, the data is output to real-time air quality models through a special kind of virtual sensor (see Section 6.5)(4). The current implementation of the models includes a nearest-neighbor one and a static yearly model (5). These are chained and the one with the lowest uncertainty is automatically chosen at query time. The mobile version of GSN, called tinyGSN (6), is used by the participants to get the pollution concentration data from GSN. Two mechanisms have been implemented for this purpose (7): either it does a single query on the RESTful API or it registers a continuous query and gets the data through push notifications (in this case Google Cloud Messaging). TinyGSN is also used for managing the local sensors, including the accelerometer, GPS and air quality sensors, and is able to push its measurements back to GSN (8), to be used to improve the pollution models. The following sections give more details on the implementation for each module.

6.4 Personal Sensing Device

6.4.1 Hardware

In addition to the smartphone itself, in this case a Samsung Galaxy S2, we used a home-made sensor extension to allow us to measure the temperature, humidity and some parameters about the air quality, namely the ozone and volatile organic compounds (VOC). This extension is in the form of a small printed circuit board, connected to the phone through its USB port. It uses the capability of the phone to act as a USB host and to provide the external devices with power. A converter (FTDI USB to RS232 UART Serial Converter) takes care of converting the signal sent from the micro-controller to the USB port, by emulating a serial port. Figure 6.2 shows the electrical circuit of the sensor board and how it connects to the smartphone.



(a) Connection of the PCB to the smartphone

(b) Detailed schematics of the PCB

Figure 6.2: Air Quality extension for smartphones

The sensors selected for this prototype are metal-oxide sensors, that operate by heating a sensitive layer and measuring its resistance. They were chosen because of their very small size and the ozone sensor was already validated in the OpenSense project for reporting accurate values after calibrating them [104]. The second sensor, measuring VOCs, was selected for its relevance to indoor air quality. Moreover, a temperature and humidity sensor was added for calibrating purposes. Except for the latter, the sensors have no factory calibration and the resistance of their sensitive layer was varying from one to another. Therefore, once built, the extensions first had to be calibrated, following the indications given on their data-sheets and then regularly re-calibrated to avoid drifts [259].

The other sensors used as input are the ones from the smartphone. The accelerometer, gyroscope, GPS and Wi-Fi were triggered by the continuous location monitoring module, moreover the accelerometer's output was also fed to the activity recognition module.

6.4.2 Software

On the smartphone we ran the tinyGSN application, a lightweight version of GSN, that was adapted for this scenario. We implemented a smart scheduler for adapting the sensor sampling and reducing the energy consumption for the continuous location tracking module and added an activity recognition virtual sensor.

The principles of tinyGSN, following the ones of GSN, are to present the users with a zero-programming way of setting up complex data-stream processing pipelines. For this purpose, the usual xml configuration files heavily used in GSN were moved to the local database and can be modified from the user interface, thus allowing the user to quickly create new virtual sensors and connect them together. The wrappers for most of the local sensors are part of the base version of tinyGSN and new wrappers can easily be added. For example, we implemented a wrapper for gathering data from the USB device presented above. The wrappers' design is following the best practices for continuous sensing on the Android platform, by running as `IntentServices` being regularly woken up by the `AlarmManager`. Proceeding this way allows the operating system to optimize the scheduling of various background processes and gives the CPU more sleeping time.

The virtual sensor processing classes can also be easily extended and can be almost directly copied from the GSN server. The activity recognition task has been implemented in such a class and has a parameter to define the classification model to use. The various models' implementations are taken from the WEKA [98] data mining toolkit. The models are trained offline, from the WEKA graphical user interface and exported in files, which are then loaded by tinyGSN according to its parameters.

As the wrappers are running as services, they have to be defined as singletons, providing data to several virtual sensors through queue data structures, themselves encapsulated in `StreamSources` instances. Each queue is subscribing to the data stream of a wrapper and applies the sliding window aggregation. Its results are then pushed to the corresponding virtual sensor. This flexibility allows streams to be split and merged without any redundant pieces. A specific wrapper takes care of reading the output of a virtual sensor and allows them to be chained.

A scheduler has also been implemented as a separate service and, according to predefined rules on the sensor output, can selectively change their duty cycles. The scheduler is able to gracefully degrade its performance if some sensors are not available or no virtual sensors are attached and running. Since one of the constraints of the system was being able to run over 24 hours without recharging the battery, we also designed an adaptive sampling scheme for the location tracking part, following the sensor triggering mechanism presented in Chapter 2.2.1. This scheme was implemented through the scheduler and affects the duty cycle of the accelerometer, GPS and Wi-Fi scanning (see Section 6.6.2).

6.5 Aggregation Server

To manage and process the streams of data coming from all the sensors and to run point and continuous queries from the mobile application, the Global Sensor Networks (GSN) [1] server is used. It is able to acquire data from various sources and to deploy new processing classes on-the-fly, allowing fast development of new features. Several improvements to the original GSN server are detailed further in this section (See the GSN documentation for details about its implementation and architecture[4]).

Without getting too much into the details, GSN is based on *wrappers* and *virtual-sensors*. *Wrappers* abstract the different sources, as, for example sensors or files, and propagate every new piece of data into the system to the connected *virtual-sensors*. On their side, *virtual-sensors* are the processing classes: they can store and process data items before forwarding them to the next *virtual-sensor*, if any. A *DataDistributer* is responsible for forwarding the data items to the *virtual-sensors* internally, but also externally using a push mechanism.

We have made several additions to GSN, the main ones being the following: support for models, improved asynchronous communication and an internal monitoring system.

The first modification to GSN is the support of models directly in the *virtual-sensors*. We developed a *Modeling Virtual Sensor* which is in charge of a list of models, extending the abstract Java class `AbstractModel`. Every time the virtual sensor receives new data, it propagates it to all its models, allowing them to stay up-to-date. It also allows direct queries on the models, by using the REST API or by registering a continuous query to the *ModelDistributer*, an extension of the *DataDistributer*. An example of usage of such a module is the generation of a pollution map from the reports of some mobile sensor nodes. For demonstration purposes, we implemented a nearest neighbor model, which returns the geographically closest measurement point, within a defined time window. But much more complex models can also be used.

The second improvement to the GSN server relates to its internal and external communication. Previously, the communication between virtual sensors was implemented through the *local*, *remote* and *remote-rest* wrappers. The *local* and *remote* wrapper registered a query with the *DataDistributer* and the latter was responsible for pushing any new stream elements to the local virtual sensor or as an xml file through the HTTP protocol to a remote GSN instance. Whereas the *remote-rest* wrapper was based on the pull paradigm, regularly polling the remote server to check if any new data was available, again using xml encoding over the HTTP protocol. The weaknesses of this system were manifold. First the local wrapper was

---

[4] https://github.com/LSIR/gsn/wiki/Documentation

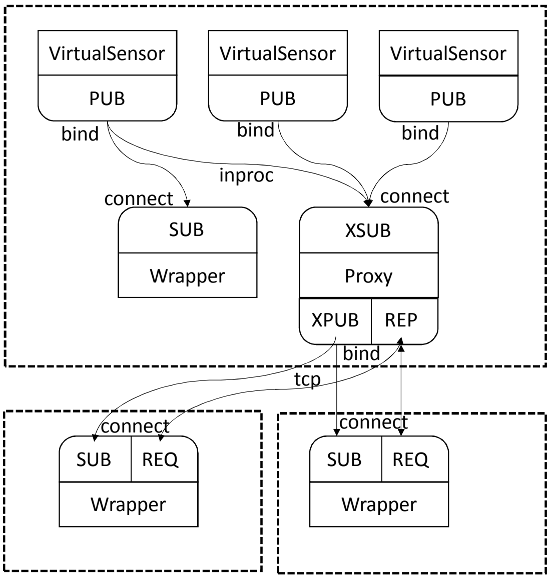


Figure 6.3: Internal and external communication, based on ØMQ

running on the same thread as the *DataDistributer*, which was also running all the local virtual sensors' processing classes, thus preventing parallelization of the stream processing. Secondly, the remote wrappers were not optimized for efficiently serializing the stream elements and the overhead of the HTTP protocol was also limiting the processing rate. Moreover, remote disconnections were sources of instability, sometimes blocking the whole process.

Therefore, we based our new communication system on the concept of message queues, unifying internal and external communications. In this scheme, the *DataDistributer* is only responsible for publishing the new stream elements to the corresponding queues. Then either a local or remote wrapper subscribes to the queue and processes the data in its own thread, this way having at least one thread per virtual sensor. We chose the ØMQ (zeromq)[5] implementation for the queues as it was the only one not relying on an external broker and providing the needed functionalities. More precisely, we used the PUB, SUB, XPUB and XSUB endpoints for publishing and subscribing to the queues, and REQ and REP for providing the virtual sensors' metadata, as shown in Figure 6.3.The queue implementation takes care of the transport protocol, transparently providing both endpoints for publishing and subscribing. For serialization, we decided to use Kryo[6], as it appeared to be more efficient according to our experiments (see Section 6.7). An interesting side effect of these modifications is that a virtual sensor can now be moved from one GSN instance to another, without any change in its configuration.

Finally the last major improvement of GSN relates to the monitoring of its own resources.

---

[5] http://zeromq.org/

[6] https://github.com/EsotericSoftware/kryo

When running hundreds of virtual sensors and parallel data-flows, one may want to understand which one is using which proportion of the system's resources, either for identifying malfunctioning virtual sensors, memory leaks or for helping to design the data streams across several servers and distribute the load. We can also imagine a system that is able to automatically scale when the load increases and move virtual sensors to new machines as necessary. Monitoring is also needed to detect anomalies and processing errors, like parsing the sensor measurements or detecting when a sensor goes faulty. Currently, we implemented a monitoring end-point on GSN that allows external tools such as Munin or collectd[7] to gather data. Virtual sensors and wrappers then register their probes with this end point and get polled regularly. We have implemented probes for the input and output data rate, out-of-order items, processing errors, and configurable data anomalies.

6.6 Sensor Fusion

6.6.1 Activity and breathing

Activity recognition is performed on top of the accelerometer output, using WEKA's implementation of the Random Forest classifier. We chose six different activities that represented the various intensity of action one may perform during a typical day: sitting, standing, walking, biking, climbing stairs, and running. To reduce the battery consumption, the accelerometer was sampled at 16Hz, which has been shown to be the optimal choice for most activities [310], every ten seconds, for a duration of two seconds.

Our estimation of pollution exposure is based on pNEM [125], a probabilistic Exposure Model from the National Ambient Air Quality Standards, and especially its step 4: "Estimate the pollutant concentration, alveolar ventilation rate, and physiological indicator (if applicable) associated with each exposure event."[125, p. 2-1]

V_A is the alveolar ventilation rate, expressed as a number of liters of air breathed per minutes. We assume it is constant during an activity period, where Δt is the activity duration and defined as:

$$V_A(\Delta t) = 19.63 \cdot \min\left(ECF \cdot MET \cdot RMR, NVO_{2max} \cdot BM \cdot (1.212 - 0.14 \cdot \ln(\Delta t))\right), \tag{6.1}$$

where ECF is the Energy Conversion Factor($\cong 0.21$), MET is the Metabolic Equivalent of Task, RMR is the resting metabolic rate, NVO_{2max} is the maximum oxygen uptake rate per kg and BM is the body mass. The MET values, only depending on the activity performed, were obtained from the Compendium of Physical Activities [9], and averaged for the few activity categories recognized by our classifier. The RMR can be computed as a linear function of the weight,

---

[7] http://munin-monitoring.org/ and https://collectd.org/

using coefficients from a table based on age and gender. Finally the NVO_{2max} can also be obtained from a table summarizing experimental results with various populations (age and gender). Both of these tables can be found in Section 5-5 of the report of Johnson et al. [125].

The alveolar ventilation rate is directly proportional to the oxygen uptake which is itself limited by the body capacity. Indeed, the circulatory system can carry a limited amount of oxygen to the muscles and if they need more energy, they will switch from an aerobic to an anaerobic process, producing energy without oxygen. Therefore in Formula 6.1 we take the minimum between the oxygen uptake rate computed from the activity and the maximum oxygen uptake rate given the duration of the activity, which decreases logarithmically with the duration. In practice, we should also take into account consecutive activities and have sliding windows for limiting the maximum oxygen uptake rate, but for our implementation, we mostly use this limit to avoid unrealistic exposures during short term activities, considering that our participants will not run a triathlon during the experiment.

V_A is then computed on the activity recognition virtual sensors' output, given the parameters entered by the participants, namely gender, weight and age.

6.6.2 Location

The continuous location tracking part is based on a state machine which reflects the high level user's state, along two orthogonal dimensions: moving/stationary, and indoor/outdoor. In addition, an additional state is used when the device is lost, meaning that it does not have enough information to infer its location. Then, based on the current state and possible transitions, a defined set of sensors are enabled with a specific duty cycle. The rules determining the transitions and duty cycles are implemented inside the scheduler of tinyGSN. The stationary states can be triggered when the GPS location is not changing for a few minutes, the visible Wi-Fi fingerprint is stable enough or the accelerometer values indicate no movement. The reason for entering the state also defines the trigger for exiting it, like a change in Wi-Fi fingerprint or for the GPS and accelerometer, a sudden change in accelerometer activity. Once in the stationary state, the GPS is shutdown and only the sensors needed for detecting a state change are sampled, at low rate. While being moving outdoors, the GPS is used for obtaining the location, whereas for indoor location, we rely on Wi-Fi fingerprints. The latter can be learned over time by associating the latest GPS location with each Wi-Fi access point and estimating its actual location, or simply by using the Android Location API and relying on Google's online database.

6.6.3 Pollution concentration

As the external ozone and VOC sensors only have a limited reliability, we try to mostly rely on the interpolation generated from the OpenSense deployment for outdoor pollutants concentration. For indoors, however, we have to rely on the attached sensors. Therefore, we use the time spent outdoors to collect data from the device and the OpenSense deployment to compute calibration parameters that we will then use when going indoors. At the GSN server level, the modeling virtual sensor is used to host the model. In our case the preliminary tests were done on a nearest neighbor model and a static yearly model, but more recently land-use regression models were also built within the OpenSense project and will also be integrated.

6.6.4 Pollution exposure

To estimate the actual exposure to pollution, we fuse the data from the alveolar ventilation rate and the pollutant concentration in the air. As different pollutants cause different kinds of chemical reactions within the lungs, we do not consider here the actual body absorption or derived chemical reactions in our simplified model, but only estimate the quantity of pollutants that were inhaled over a certain period. For that, we transform the concentration, sometimes given in parts per billion (ppb) or ppm (parts per million) of volume into micro-grammes per cubic meter ($\mu g/m^3$) and multiply it by $\frac{V_A \cdot \Delta t}{1000}$ to obtain the quantity of pollutant in micro-grammes inhaled for a given activity. By summing the activities of the day, we obtain the daily exposure.

6.7 Experiments

6.7.1 GSN server

To evaluate the performance of the new communication system, we deployed GSN on 24 virtual machines, provisioned with 2 cores and 3GB of RAM. Each GSN server had 25 virtual sensors: one generating data every 10 ms and 24 connected to the other instances, including itself. The storage was kept in memory using the H2:mem database to reduce the disk writing overload. In the first experiment, the remote wrapper was used to connect the 24 virtual sensors, whereas in the second one the zeromq wrapper was used with xml serialization and finally in the last experiment zeromq was using the Kryo serialization. Each experiment lasted around 20 minutes, during which two snapshots were taken. One after a few minutes to ensure that all the connections had been established and one 15 minutes later. These snapshots contained the number of elements processed by each of the 24 virtual sensors.

On Figure 6.4, we can see that with the remote wrapper, the production of stream elements is almost able to work at its maximal rate, but due to the overhead of the communication and serialization protocols, consumers are not able to receive the data fast enough. Changing the communication to the zeromq wrapper seems to reduce the production rate, the CPU being the bottleneck. This is because now the communication being faster, it needs more resources to deserialize the stream elements and starts competing with the computational resources of the producer. Finally, the Kryo serialization reduces again the time needed to process one element, increasing the incoming data rate and thus also reducing the resources available for the producer. Interestingly in this last experiment, stream element production

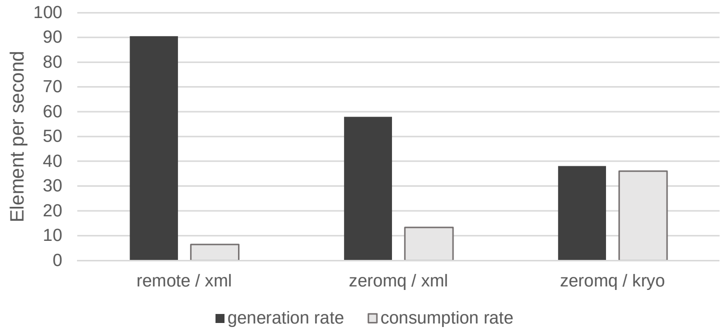


Figure 6.4: comparison of elements produced and consumed per second for the 3 scenarios: using remote wrapper, using zeromq wrapper with xml serialization or Kryo serialization

sitting	standing	walking	biking	climbing stairs	running	class
28	2	0	0	0	0	sitting
1	35	1	2	2	1	standing
0	2	14	0	2	1	walking
0	1	0	31	2	0	biking
0	4	2	3	9	1	climbing stairs
1	2	0	0	1	25	running

Table 6.1: Confusion matrix for the Random Forest activity classifier

and consumption reached an equilibrium, meaning that the serialization process is no longer the bottleneck. It should be noted that the figures are computed per virtual sensor and thus the actual number of items processed per second is around 900.

6.7.2 tinyGSN

Various experiments have been conducted with tinyGSN. The first one was to evaluate the activity recognition module. The data was recorded by three volunteers during one hour. They were instructed to perform a predefined sequence of activities and were taking notes of the starting and ending times. The features were generated over the 2 second window of duty cycled accelerometer values. They included statistics over the time and frequency domains, but focused on the ones that were fast to compute. Among the different classifiers tested (C4.5, Random Forest, Naive Bayes, Logistic and Multi Layer Perceptron), the Random Forest showed the best results on a 10-fold cross validation with an F-Measure of 0.82 and an ROC Area of 0.94. The confusion matrix is show in Table 6.1.

The second experiment evaluated the battery duration while continuously recording the location. With the scheduler taking care of enabling and disabling the sensors, we also removed all unneeded applications from the phone and set it to flight safe mode. During 4 days, a volunteer carried the phone everywhere she went and was asked to let the battery

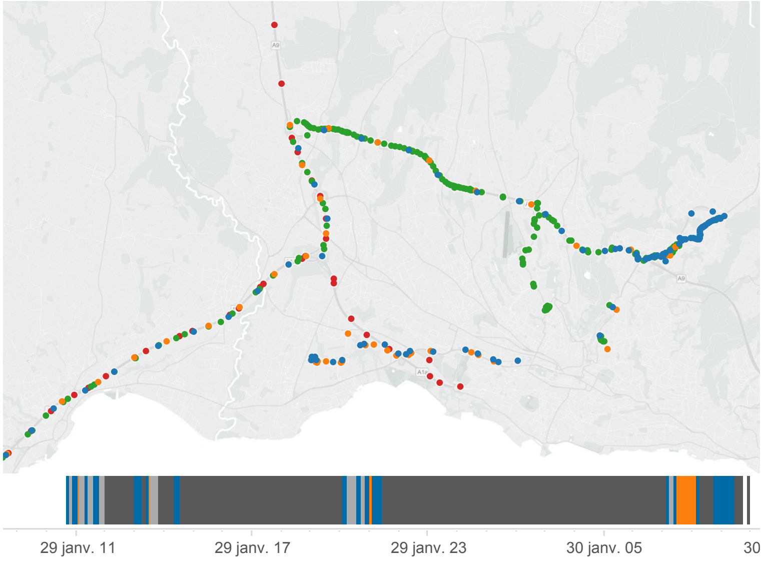


Figure 6.5: The dots on the map represent the GPS points collected, colored by day, whereas the timeline at the bottom shows the various states of the system on the first day. (blue: lost, orange: GPS moving, light gray: stationary by accelerometer, dark gray: stationary by Wi-Fi)

completely deplete before charging again. In average, it lasted around 26 hours.

From the Figure 6.5, we can seen that some points are missing on the trajectory, either due to the unavailability of GPS, like in the metro or by wrongly detecting a stationary state with the accelerometer, for example when the user was driving, but interpolating the points could be a solution. The timeline indicates that most of the time was spent in a low power *stationary* state and that the GPS needed some time to start, being inefficient for short periods of motion.

The system is currently used as part of a cohort study at the Centre Hospitalier Universitaire Vaudois (CHUV), linked to the OpenSense project and we expect to get the results soon, demonstrating the complete system at work.

6.8 Discussion

In this chapter, we presented an end-to-end system, fusing the data collected locally on the smartphone, through the accelerometer, GPS and Wi-Fi, with global air quality data gathered by a mobile sensor network, mounted on public transportation. We showed how the combination of different data sources allows us to extract more information, improving the utility of the system. For example, the continuous location tracking part, takes advantage of the GPS, Wi-Fi and accelerometer measurements for reducing its needs and allows us to run an experiment of 24 hours. The activity, inferred from the accelerometer is also fused with the local pollution concentration to produce a realistic exposure estimation. Finally, the local and global air quality sensing systems complement each other to provide outdoor as well as indoor air pollution information.

However, this prototype has some weaknesses. To make the extension board small enough so that it can be attached directly to the smartphone, we had to choose Metal Oxide Sensors. On the downside, those sensors are less suited for precise measurement and are usually used for threshold detection in places where the gas concentration can vary over several orders of magnitude. Even if it has been shown that it is possible to calibrate them to obtain valid measurements, we don't reach the same level of accuracy, mostly because we have no control over the position and placement of the sensor (pocket, table, wind,...) and for the next prototypes iterations, we would suggest using bigger sensors that are also more accurate.

On the software side, the GSN platform, including the mobile application tinyGSN, is well suited for a rapid development of such prototypes, but shows significant limitations when used under heavy load. We found a way to overcome those limitations, at least to support the OpenSense back-end requirements. For a larger participatory sensing scenario, it may be required to rethink GSN from scratch, building from the lessons learned.

7 Conclusion

In this thesis we explored different approaches that could enable continuous context sensing on smartphones in a way that would preserve the scarce resources available on the mobile platforms. The challenges we faced were trading off the battery life with the accuracy and latency of the context inference.

We presented a generic mobile device sensing architecture, composed of the sensors, the sensor abstraction layer, the sensor management modules, and the context extraction framework. We used this architecture to put in context the contributions of this thesis, related to device collaboration, sensor scheduling, and stream data processing.

The first approach we presented in Chapter 3 related to the use of local communications to exchange sensing information with neighboring devices. Indeed, proximity, location and environmental sensing can be offloaded to nearby external sensors, which can also be the smartphones themselves. In the latter case, as the devices can send and receive measurements, we designed a protocol for synchronizing the exchanges and fairly distribute the sensing tasks. We showed both theoretically and experimentally, by running a simulation over two real-world datasets, that it can indeed reduce the energy needed in the case where devices spend enough time in the vicinity.

The second approach, in Chapter 4, focused on the way to schedule mobile sensors, optimizing for both the accuracy and energy needs. We formulated the optimal sensing problem as a decision problem and proposed a two-tier framework for approximating the solution, called Optimos. The first tier is responsible for segmenting the sensor measurement time series. The segments produced are optimized to reduce the approximation error of a model, like a linear or SVM regression. The second tier takes care of estimating the optimal sampling, selecting the measurement times that contribute the most to the model accuracy. We provided near-optimal heuristics for both tiers of our framework and evaluated their performances using environmental sensor traces.

The third approach, described in Chapter 5, was based on the principle that sensor data is inherently noisy and redundant. Therefore, we proposed an online algorithm, called StateFinder, that identifies repeated patterns in time series and produces a compressed symbolic output. The first symbolic transformation, called spClust, applies clustering on the raw sensor data, and the subsequent iterations encode repetitive sequences of symbols into new symbols. We also presented a metric to evaluate how good a symbolization method is at preserving the state information in time series and showed that spClsut outperforms other clustering methods. We also showed that the output of StateFinder can be used directly for various data mining tasks, such as classification or forecasting, without impacting much on the accuracy, but reducing greatly the complexity and running time.

In the last Chapter, we presented an example of an application, called ExposureSense, that demonstrated the many opportunities to enhance contextual information when fusing sensor data from different sources. We developed a complete ecosystem for assessing the exposure of the users to air pollutants in their daily lives. On one side we gather fine grained air quality information from mobile sensor deployments, for example on the roofs of buses and aggregate them in an interpolation model and on the other side, we continuously capture the user's context, including location, activity and surrounding air quality with a home-made sensor board. We also presented the various models used for fusing location, activity, and local and global air quality information in order to produce the exposure estimation.

7.1 Future directions

The research in mobile sensing is technology driven, in the same way that recent technological developments, such as Google Glass or the Apple Watch, have highly influenced research directions, bringing new challenges to be solved. These new sensors and sensor platforms enable new sensing modalities, new interactions with the users and further integration into our daily lives. But also many of the technologies presented in our survey (see Chapter 2) and in the thesis itself are ready for the technology transfer and may reach the market soon, closing the loop.

When changing the scale, from the individual, as in the approach of this thesis, to the population at large, a very important aspect, orthogonal to our research, relates to the issues of privacy and trust. Indeed, locally collecting sensor information for the user's own usage only has a limited impact. Similar to the advantages of fusing data from various sensors, fusing data from individuals also increases its utility and benefits to everyone. But are we ready to trade our data for these advantages, up to what point and with whom? Whether the population decides or not to adopt this new paradigm, the most important point remains that it must be with full knowledge of the facts.

In addition to the user's context, on which we focused in this thesis, sensing devices also benefit from being aware of their own context. Even if the resources available on smartphones may make some of the resource optimization discussed in this thesis useless, numerous much smaller devices, sometimes referred to as smart-dust, will benefit from them and become smarter by understanding their context. In the same way, after giving them the possibility to sense their environment, the connected and smart *things* will also want to understand it and take action.

Finally, sensors are massive producers of data and greatly contributed to the Big Data era. While researchers are putting a great deal of effort in supporting the storage and processing of such huge amount of data in datacenters, the solution could actually come from the network edges. Indeed, the cumulated distributed resources available at the data producers' side is very often neglected. However, those resources could be used to pre-process the data, extract information or even actively participate in the retrieval process.

Bibliography

[1] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks. In *2007 International Conference on Mobile Data Management*, pages 198–205. Conference and Custom Publishing, May 2007. ISBN 1-4244-1241-2. doi: 10.1109/MDM.2007.36.

[2] K. Aberer, S. Sathe, D. Chakraborty, A. Martinoli, G. Barrenetxea, B. Faltings, and L. Thiele. OpenSense: Open Community Driven Sensing of Environment. In *GIS-IWGS*, pages 39–42, 2010.

[3] K. Aberer, J.-E. Ranvier, M. Vasirani, Z. Yan, M. Catasta, G. Christodoulou, I. Gavrilovic, F. Hrisafov, M. Monney, A. Ouaazki, B. Perovic, and H. Radu. Memo-it: Don't write your diary, Sense it. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct*, pages 203–206, New York, New York, USA, Sept. 2014. ACM Press. ISBN 9781450330473. doi: 10.1145/2638728.2638753.

[4] V. Agarwal, N. Banerjee, D. Chakraborty, and S. Mittal. USense – A Smartphone Middleware for Community Sensing. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 56–65. IEEE, June 2013. ISBN 978-0-7695-4973-6. doi: 10.1109/MDM.2013.16.

[5] C. C. Aggarwal. *Data streams: models and algorithms*, volume 31. Springer Science & Business Media, 2007.

[6] C. C. Aggarwal. *Managing and Mining Sensor Data*. Springer US, Boston, MA, 2013. ISBN 978-1-4614-6308-5. doi: 10.1007/978-1-4614-6309-2.

[7] J. Aggarwal and L. Xia. Human activity recognition from 3D data: A review. *Pattern Recognition Letters*, 48:70–80, Oct. 2014. ISSN 01678655. doi: 10.1016/j.patrec.2014.04.011.

[8] N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland. Social fMRI: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6): 643–659, Dec. 2011. ISSN 15741192. doi: 10.1016/j.pmcj.2011.09.004.

[9] B. E. Ainsworth, W. L. Haskell, S. D. Herrmann, N. Meckes, D. R. Bassett, C. Tudor-Locke, J. L. Greer, J. Vezina, M. C. Whitt-Glover, and A. S. Leon. 2011 Compendium of Physical Activities: a second update of codes and MET values. *Medicine and science in sports and exercise*, 43(8):1575–81, Aug. 2011. ISSN 1530-0315. doi: 10.1249/MSS.0b013e31821ece12.

[10] K. Altun and B. Barshan. Pedestrian dead reckoning employing simultaneous activity recognition cues. *Measurement Science and Technology*, 23(2):025103, Feb. 2012. ISSN 0957-0233. doi: 10.1088/0957-0233/23/2/025103.

[11] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Proceedings of the 4th International Conference on Ambient Assisted Living and Home Care*, IWAAL'12, pages 216–223, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-35394-9. doi: 10.1007/978-3-642-35395-6_30.

[12] M. Azizyan, I. Constandache, and R. Roy Choudhury. SurroundSense. In *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09*, page 261, New York, New York, USA, Sept. 2009. ACM Press. ISBN 9781605587028. doi: 10.1145/1614320.1614350.

[13] P. Bahl and V. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 2, pages 775–784. IEEE, 2000. ISBN 0-7803-5880-5. doi: 10.1109/INFCOM.2000.832252.

[14] P. Baier, F. Durr, and K. Rothermel. PSense: Reducing Energy Consumption in Public Sensing Systems. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 136–143. IEEE, Mar. 2012. ISBN 978-1-4673-0714-7. doi: 10.1109/AINA.2012.33.

[15] R. Balani. Energy consumption analysis for bluetooth, wifi and cellular networks. Technical report, Univ. California at Los Angeles, 2007.

[16] E. Bales, N. Nikzad, N. Quick, C. Ziftci, K. Patrick, and W. Griswold. Citisense: Mobile air quality sensing for individuals and communities Design and deployment of the Citisense mobile air-quality system. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pages 155–158, 2012.

[17] J. Ballesteros, M. Rahman, B. Carbunar, and N. Rishe. Safe cities. A participatory sensing approach. In *37th Annual IEEE Conference on Local Computer Networks*, pages 626–634. IEEE, Oct. 2012. ISBN 978-1-4673-1564-7. doi: 10.1109/LCN.2012.6423684.

[18] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner. Virtual Compass: Relative Positioning to Sense Mobile Social Interactions. In P. Floréen, A. Krüger, and

M. Spasojevic, editors, *Pervasive Computing SE - 1*, volume 6030 of *Lecture Notes in Computer Science*, pages 1–21. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12653-6. doi: 10.1007/978-3-642-12654-3_1.

[19] G. Bauer and P. Lukowicz. Can smartphones detect stress-related changes in the behaviour of individuals? In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 423–426. IEEE, Mar. 2012. ISBN 978-1-4673-0907-3. doi: 10.1109/PerComW.2012.6197525.

[20] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, M. Terada, and R. Han. WhozThat? evolving an ecosystem for context-aware mobile social networks. *IEEE Network*, 22(4):50–55, July 2008. ISSN 0890-8044. doi: 10.1109/MNET.2008.4579771.

[21] A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Mishra, and K. Seada. Fusing mobile, sensor, and social data to fully enable context-aware computing. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications - HotMobile '10*, page 60, New York, New York, USA, Feb. 2010. ACM Press. ISBN 9781450300056. doi: 10.1145/1734583.1734599.

[22] C. Belley, S. Gaboury, B. Bouchard, and A. Bouzouane. An efficient and inexpensive method for activity recognition within a smart home based on load signatures of appliances. *Pervasive and Mobile Computing*, 12:58–78, June 2014. ISSN 15741192. doi: 10.1016/j.pmcj.2013.02.002.

[23] R. Bellman. On the Approximation of Curves by Line Segments Using Dynamic Programming. *Communications of the ACM*, 4(6):284, 1961.

[24] S. Ben Mokhtar and L. Capra. From pervasive to social computing. In *Proceedings of the 2009 international conference on Pervasive services - ICPS '09*, page 169, New York, New York, USA, July 2009. ACM Press. ISBN 9781605586441. doi: 10.1145/1568199.1568229.

[25] P. Bhargava, N. Gramsky, and A. Agrawala. SenseMe: A System for Continuous, On-Device, and Multi-dimensional Context and Activity Recognition. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 40–49. ICST, Dec. 2014. ISBN 978-1-63190-039-6. doi: 10.4108/icst.mobiquitous.2014.257654.

[26] S. Bhattacharya, P. Nurmi, N. Hammerla, and T. Plötz. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing*, 15: 242–262, Dec. 2014. ISSN 15741192. doi: 10.1016/j.pmcj.2014.05.006.

[27] N. Bicocchi, G. Castelli, M. Lasagni, M. Mamei, and F. Zambonelli. Experiences on sensor fusion with commonsense reasoning. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2012*, pages 596–601, 2012. ISBN 9781467309073. doi: 10.1109/PerComW.2012.6197585.

[28] G. Bigwoodt, A. C. Viana, M. Dias de Amorim, and M. Boc. Collaborative data collection in global sensing systems. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 243–246. IEEE, Oct. 2011. ISBN 978-1-61284-928-7. doi: 10.1109/LCN.2011.6115199.

[29] Z. Bin and Y. Kim. Connectionless broadcast channel and mobility of Bluetooth. In *Vehicular Technology Conference. IEEE 55th Vehicular Technology Conference. VTC Spring 2002 (Cat. No.02CH37367)*, volume 2, pages 918–922. IEEE. ISBN 0-7803-7484-3. doi: 10.1109/VTC.2002.1002622.

[30] J. Blom and R. Hänninen. Air pollution in everyday life: toward design of persuasive urban air quality services. *Persuasive Technology*, page 5, 2012.

[31] A. Bogomolov, B. Lepri, M. Ferron, F. Pianesi, and A. S. Pentland. Daily Stress Recognition from Mobile Phone Data, Weather Conditions and Individual Traits. In *Proceedings of the ACM International Conference on Multimedia - MM '14*, pages 477–486, New York, New York, USA, Nov. 2014. ACM Press. ISBN 9781450330633. doi: 10.1145/2647868.2654933.

[32] D. Brateris, D. Bedford, D. Calhoun, A. Johnson, N. Kowalski, K. Martino, T. Mukalian, J. Reda, A. Samaritano, and R. R. Krchnavek. iOS hardware as a sensor platform: DMM case study. In *2011 IEEE Sensors Applications Symposium*, pages 308–311. IEEE, Feb. 2011. ISBN 978-1-4244-8063-0. doi: 10.1109/SAS.2011.5739825.

[33] D. J. Briggs, K. de Hoogh, C. Morris, and J. Gulliver. Effects of travel mode on exposures to particulate air pollution. *Environment international*, 34(1):12–22, Jan. 2008. ISSN 0160-4120. doi: 10.1016/j.envint.2007.06.011.

[34] N. Brouwers and K. Langendoen. Pogo, a middleware for mobile phone sensing. In *Proceedings of the 13th International Middleware Conference*, Middleware '12, pages 21–40, New York, NY, USA, 2012. Springer-Verlag New York, Inc. ISBN 978-3-642-35169-3.

[35] W. Brunette, R. Sodt, R. Chaudhri, M. Goel, M. Falcone, J. Van Orden, and G. Borriello. Open data kit sensors: A Sensor Integration Framework for Android at the Application-Level. In *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*, page 351, New York, New York, USA, June 2012. ACM Press. ISBN 9781450313018. doi: 10.1145/2307636.2307669.

[36] W. Brunette, M. Sundt, N. Dell, R. Chaudhri, N. Breit, and G. Borriello. Open data kit 2.0: Expanding and Refining Information Services for Developing Regions. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications - HotMobile '13*, page 1, New York, New York, USA, Feb. 2013. ACM Press. ISBN 9781450314213. doi: 10.1145/2444776.2444790.

[37] M. Budde, M. Berning, M. Busse, T. Miyaki, and M. Beigl. The TECO Envboard: A mobile sensor platform for accurate urban sensing — And more. In *2012 Ninth International Conference on Networked Sensing (INSS)*, pages 1–2. IEEE, June 2012. ISBN 978-1-4673-1786-3. doi: 10.1109/INSS.2012.6240573.

[38] M. Budde, P. Barbera, R. El Masri, T. Riedel, and M. Beigl. Retrofitting smartphones to be used as particulate matter dosimeters. In *Proceedings of the 17th annual international symposium on International symposium on wearable computers - ISWC '13*, page 139, New York, New York, USA, Sept. 2013. ACM Press. ISBN 9781450321273. doi: 10.1145/2493988.2494342.

[39] M. N. Burns, M. Begale, J. Duffecy, D. Gergle, C. J. Karr, E. Giangrande, and D. C. Mohr. Harnessing context sensing to develop a mobile intervention for depression. *Journal of medical Internet research*, 13(3):e55, Jan. 2011. ISSN 1438-8871. doi: 10.2196/jmir.1838.

[40] O. Canovas, P. Lopez-de Teruel, and A. Ruiz. WiFiBoost: a terminal-based method for detection of indoor/outdoor places. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 352–353. ICST, Dec. 2014. ISBN 978-1-63190-039-6. doi: 10.4108/icst.mobiquitous.2014.258063.

[41] N. A. Capela, E. D. Lemaire, and N. Baddour. Feature Selection for Wearable Smartphone-Based Human Activity Recognition with Able bodied, Elderly, and Stroke Patients. *PloS one*, 10(4):e0124414, Jan. 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0124414.

[42] G. Cardone, A. Cirri, A. Corradi, L. Foschini, and D. Maio. MSF: An Efficient Mobile Phone Sensing Framework. *International Journal of Distributed Sensor Networks*, 2013: 1–9, 2013. ISSN 1550-1329. doi: 10.1155/2013/538937.

[43] A. Carlotto, M. Parodi, C. Bonamico, F. Lavagetto, and M. Valla. Proximity classification for mobile devices using wi-fi environment similarity. In *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments - MELT '08*, page 43, New York, New York, USA, Sept. 2008. ACM Press. ISBN 9781605581897. doi: 10.1145/1410012.1410023.

[44] D. Carlson and A. Schrader. Dynamix: An open plug-and-play context framework for android. In *2012 3rd IEEE International Conference on the Internet of Things*, pages 151–158. IEEE, Oct. 2012. ISBN 978-1-4673-1346-9. doi: 10.1109/IOT.2012.6402317.

[45] C.-C. Chang and C.-J. Lin. Libsvm: a Library for Support Vector Machines. *ACM TIST*, 2 (3):27, 2011.

[46] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical report, Hanover, NH, USA, Nov. 2000.

[47] L. Chen, C. D. Nugent, and H. Wang. A Knowledge-Driven Approach to Activity Recognition in Smart Homes. *IEEE Transactions on Knowledge and Data Engineering*, 24(6): 961–974, June 2012. ISSN 1041-4347. doi: 10.1109/TKDE.2011.51.

[48] 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 P. Dourish and A. Friday, editors, *UbiComp 2006: Ubiquitous Computing SE - 14*, volume

4206 of *Lecture Notes in Computer Science*, pages 225–242. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-39634-5. doi: 10.1007/11853565_14.

[49] Z. Chen, H. Zou, H. Jiang, Q. Zhu, Y. C. Soh, and L. Xie. Fusion of WiFi, smartphone sensors and landmarks using the Kalman filter for indoor localization. *Sensors (Basel, Switzerland)*, 15(1):715–32, Jan. 2015. ISSN 1424-8220. doi: 10.3390/s150100715.

[50] Chenshu Wu, Zheng Yang, Yunhao Liu, and Wei Xi. WILL: Wireless Indoor Localization without Site Survey. *IEEE Transactions on Parallel and Distributed Systems*, 24(4):839–848, Apr. 2013. ISSN 1045-9219. doi: 10.1109/TPDS.2012.179.

[51] A. D. Cheok, S. W. Fong, K. H. Goh, X. Yang, W. Liu, and F. Farzbiz. Human Pacman. In *Proceedings of the 2nd workshop on Network and system support for games - NETGAMES '03*, pages 106–117, New York, New York, USA, May 2003. ACM Press. ISBN 1581137346. doi: 10.1145/963900.963911.

[52] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking - MobiCom '10*, page 173, New York, New York, USA, Sept. 2010. ACM Press. ISBN 9781450301817. doi: 10.1145/1859995.1860016.

[53] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 493–498, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956808.

[54] J. Chon and H. Cha. LifeMap: A smartphone-based context provider for location-based services. *IEEE Pervasive Computing*, 10(2):58–67, 2011. ISSN 15361268. doi: 10.1109/MPRV.2011.13.

[55] Y. Chon, Y. Kim, H. Shin, and H. Cha. Adaptive Duty Cycling for Place-centric Mobility Monitoring using Zero-Cost Information in Smartphone. *IEEE Transactions on Mobile Computing*, PP(99):1–1, 2014. ISSN 1536-1233. doi: 10.1109/TMC.2013.151.

[56] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, P. P. Klasnja, K. Koscher, A. LaMarca, J. A. Landay, L. LeGrand, J. Lester, A. Rahimi, A. Rea, and D. Wyatt. The Mobile Sensing Platform: An Embedded Activity Recognition System. *IEEE Pervasive Computing*, 7(2):32–41, Apr. 2008. ISSN 1536-1268. doi: 10.1109/MPRV.2008.39.

[57] D. Chu and N. D. Lane. Balancing Energy , Latency and Accuracy for Mobile Sensor Data Classification. In *ACM Sensys'11*, volume 50 of *SenSys '11*, pages 54–67. ACM, 2011. ISBN 9781450307185. doi: 10.1145/2070942.2070949.

[58] D. Chu, N. D. Lane, T. T.-T. Lai, C. Pang, X. Meng, Q. Guo, F. Li, and F. Zhao. Balancing energy, latency and accuracy for mobile sensor data classification. In *Proceedings of*

the 9th ACM Conference on Embedded Networked Sensor Systems SenSys 11, volume 50 of *SenSys '11*, page 54. ACM Press, 2011. ISBN 9781450307185. doi: 10.1145/2070942.2070949.

[59] F.-L. Chung, T.-C. Fu, V. Ng, and R. Luk. An Evolutionary Approach to Pattern-Based Time Series Segmentation. *IEEE Transactions on Evolutionary Computation*, 8(5):471–489, Oct. 2004. ISSN 1089-778X. doi: 10.1109/TEVC.2004.832863.

[60] S. Consolvo, R. Libby, I. Smith, J. A. Landay, D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, and L. LeGrand. Activity sensing in the wild. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 1797, New York, New York, USA, Apr. 2008. ACM Press. ISBN 9781605580111. doi: 10.1145/1357054.1357335.

[61] I. Constandache, R. R. Choudhury, and I. Rhee. Towards Mobile Phone Localization without War-Driving. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, Mar. 2010. ISBN 978-1-4244-5836-3. doi: 10.1109/INFCOM.2010.5462058.

[62] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *KDD*, pages 16–22. AAAI Press, 1998. ISBN 1-57735-070-7.

[63] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. PRISM: Platform for Remote Sensing using Smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10*, page 63, New York, New York, USA, June 2010. ACM Press. ISBN 9781605589855. doi: 10.1145/1814433.1814442.

[64] A. de Nazelle, D. A. Rodríguez, and D. Crawford-Brown. The built environment and health: impacts of pedestrian-friendly designs on air pollution exposure. *The Science of the total environment*, 407(8):2525–35, Apr. 2009. ISSN 0048-9697. doi: 10.1016/j.scitotenv.2009.01.006.

[65] A. de Nazelle, E. Seto, D. Donaire-Gonzalez, M. Mendez, J. Matamala, M. J. Nieuwenhuijsen, and M. Jerrett. Improving estimates of air pollution exposure through ubiquitous sensing technologies. *Environmental pollution (Barking, Essex : 1987)*, 176:92–9, May 2013. ISSN 1873-6424. doi: 10.1016/j.envpol.2012.12.032.

[66] M. Demirbas, M. Ali Bayir, C. G. Akcora, Y. S. Yilmaz, and H. Ferhatosmanoglu. Crowd-sourced sensing and collaboration using twitter. In *2010 IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pages 1–9. IEEE, June 2010. ISBN 978-1-4244-7264-2. doi: 10.1109/WOWMOM.2010.5534910.

[67] A. M. Denton, C. A. Besemann, and D. H. Dorr. Pattern-based time-series subsequence clustering using radial distribution functions. *Knowledge and Information Systems*, 18 (1), Mar. 2008. ISSN 0219-1377. doi: 10.1007/s10115-008-0125-7.

[68] A. Deshwal, S. Kohli, and K. P. Chethan. Information as a service based architectural solution for WSN. In *2012 1st IEEE International Conference on Communications in China (ICCC)*, pages 68–73. IEEE, Aug. 2012. ISBN 978-1-4673-2815-9. doi: 10.1109/ICCChina.2012.6356972.

[69] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, and B. Nath. Real-time air quality monitoring through mobile sensing in metropolitan areas. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing - UrbComp '13*, page 1, New York, New York, USA, Aug. 2013. ACM Press. ISBN 9781450323314. doi: 10.1145/2505821.2505834.

[70] A. K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1): 4–7, Feb. 2001. ISSN 1617-4909. doi: 10.1007/s007790170019.

[71] A. K. Dey, K. Wac, D. Ferreira, K. Tassini, J.-H. Hong, and J. Ramos. Getting closer. In *Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11*, page 163, New York, New York, USA, Sept. 2011. ACM Press. ISBN 9781450306300. doi: 10.1145/2030112.2030135.

[72] S. Dhillon and K. Chakrabarty. Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks. 3:1609 –1614 vol.3, 2003. ISSN 1525-3511. doi: 10.1109/WCNC.2003.1200627.

[73] E. D'Hondt, M. Stevens, and A. Jacobs. Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. *Pervasive and Mobile Computing*, 9(5):681–694, Oct. 2013. ISSN 15741192. doi: 10.1016/j.pmcj.2012.09.002.

[74] S. Distefano, G. Merlino, A. Puliafito, and A. Vecchio. A hypervisor for infrastructure-enabled sensing Clouds. In *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 1362–1366. IEEE, June 2013. ISBN 978-1-4673-5753-1. doi: 10.1109/ICCW.2013.6649449.

[75] T. M. T. Do and D. Gatica-Perez. GroupUs: Smartphone Proximity Data and Human Interaction Type Mining. In *2011 15th Annual International Symposium on Wearable Computers*, pages 21–28. IEEE, June 2011. ISBN 978-1-4577-0774-2. doi: 10.1109/ISWC.2011.28.

[76] O. Dousse, J. Eberle, and M. Mertens. Place learning via direct WiFi fingerprint clustering. In *MDM '12 Proceedings of the 2012 IEEE 13th International Conference on Mobile Data Management - Industry track*, 2012.

[77] J. Dunkel, R. Bruns, and S. Stipkovic. Event-based smartphone sensor processing for ambient assisted living. In *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, pages 1–6. IEEE, Mar. 2013. ISBN 978-1-4673-5070-9. doi: 10.1109/ISADS.2013.6513422.

[78] P. Dutta, P. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff. Common Sense: Participatory Urban Sensing Using a Network of Handheld Air Quality Monitors. In *SenSys*, pages 349–350, 2009.

[79] J. Eberle and G. P. Perrucci. Energy measurements campaign for positioning methods on State-of-the-Art smartphones. In *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 937–941. IEEE, Jan. 2011. ISBN 978-1-4244-8789-9. doi: 10.1109/CCNC.2011.5766645.

[80] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. BikeNet. *ACM Transactions on Sensor Networks*, 6(1):1–39, Dec. 2009. ISSN 15504859. doi: 10.1145/1653760.1653766.

[81] A. Farshad, M. K. Marina, and F. J. Garcia. A microscopic look at WiFi fingerprinting for indoor mobile phone localization in diverse environments. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 1–10. IEEE, Oct. 2013. ISBN 978-1-4799-4043-1. doi: 10.1109/IPIN.2013.6817920.

[82] M. Faulkner, M. Olson, R. Chandy, J. Krause, K. M. Chandy, and A. Krause. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *Proceedings of the 10th ACMIEEE International Conference on Information Processing in Sensor Networks*, pages 13–24. IEEE, 2011. ISBN 9781450305129.

[83] S. Feizi, G. Angelopoulos, V. K. Goyal, and M. Medard. Energy-efficient Time-Stampless Adaptive Nonuniform Sampling. In *2011 IEEE SENSORS Proceedings*, pages 912–915. IEEE, Oct. 2011. ISBN 978-1-4244-9289-3. doi: 10.1109/ICSENS.2011.6127202.

[84] B. Ferris, D. Fox, and N. Lawrence. WiFi-SLAM using Gaussian process latent variable models. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2480–2485, Hyderabad, India, 2007. Morgan Kaufmann Publishers Inc.

[85] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick. Online Segmentation of Time Series Based on Polynomial Least-Squares Approximations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2232–2245, 2010.

[86] B. Fulcher and N. Jones. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, Jan. 2014. ISSN 1041-4347. doi: 10.1109/TKDE.2014.2316504.

[87] A. Gaggioli, G. Pioggia, G. Tartarisco, G. Baldus, D. Corda, P. Cipresso, and G. Riva. A mobile data collection platform for mental health research. *Personal and Ubiquitous Computing*, 17(2):241–251, Sept. 2011. ISSN 1617-4909. doi: 10.1007/s00779-011-0465-2.

[88] R. K. Ganti, S. Srinivasan, and A. Gacic. Multisensor Fusion in Smartphones for Lifestyle Monitoring. In *2010 International Conference on Body Sensor Networks*, pages 36–43. IEEE, June 2010. ISBN 978-1-4244-5817-2. doi: 10.1109/BSN.2010.10.

[89] L. Gao, A. K. Bourke, and J. Nelson. Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems. *Medical engineering & physics*, 36(6): 779–85, June 2014. ISSN 1873-4030. doi: 10.1016/j.medengphy.2014.02.012.

[90] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, 7(5):341–351, Oct. 2002. ISSN 1383-469X. doi: 10.1023/A:1016587515822.

[91] B. Girod, V. Chandrasekhar, D. Chen, N.-M. Cheung, R. Grzeszczuk, Y. Reznik, G. Takacs, S. Tsai, and R. Vedantham. Mobile Visual Search. *IEEE Signal Processing Magazine*, 28 (4):61–76, July 2011. ISSN 1053-5888. doi: 10.1109/MSP.2011.940881.

[92] L. Goasduff and J. Rivera. Gartner Says Smartphone Sales Surpassed One Billion Units in 2014, 2015. URL http://www.gartner.com/newsroom/id/2996817~[21.07.2015].

[93] P. Grassberger and I. Procaccia. Estimation of the kolmogorov entropy from a chaotic signal. *Phys. Rev. A*, 28:2591–2593, Oct 1983. doi: 10.1103/PhysRevA.28.2591.

[94] Y. Gu, A. Lo, and I. Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys & Tutorials*, 11(1):13–32, 2009. ISSN 1553-877X. doi: 10.1109/SURV.2009.090103.

[95] J. J. Guiry, P. van de Ven, and J. Nelson. Multi-sensor fusion for enhanced contextual awareness of everyday activities with ubiquitous devices. *Sensors (Basel, Switzerland)*, 14(3):5687–701, Jan. 2014. ISSN 1424-8220. doi: 10.3390/s140305687.

[96] T. Guo, Z. Yan, and K. Aberer. An adaptive approach for online segmentation of multi-dimensional mobile data. In *Proceedings of the 11th International Workshop on Data Engineering for Wireless and Mobile Access*, pages 7–14, New York, USA, May 2012. ACM. ISBN 9781450314428. doi: 10.1145/2258056.2258059.

[97] A. Gupta, A. Kalra, D. Boston, and C. Borcea. MobiSoC: a middleware for mobile social computing applications. *Mobile Networks and Applications*, 14(1):35–52, Nov. 2008. ISSN 1383-469X. doi: 10.1007/s11036-008-0114-9.

[98] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software. *ACM SIGKDD Explorations Newsletter*, 11(1):10, Nov. 2009. ISSN 19310145. doi: 10.1145/1656274.1656278. URL http://portal.acm.org/citation.cfm?doid=1656274.1656278.

[99] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, Jan. 2007. ISSN 1384-5810. doi: 10.1007/s10618-006-0059-1.

[100] S.-J. Han and S.-B. Cho. A hybrid personal assistant based on Bayesian networks and a rule-based system inside a smartphone. *International Journal of Hybrid Intelligent Systems*, 2(3):221–234, Aug. 2005. ISSN 1448-5869.

[101] T. Hao, G. Xing, and G. Zhou. iSleep: Unobtrusive Sleep Quality Monitoring using Smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems - SenSys '13*, pages 1–14, New York, New York, USA, Nov. 2013. ACM Press. ISBN 9781450320276. doi: 10.1145/2517351.2517359.

[102] R. Harle. A survey of indoor inertial positioning systems for pedestrians. *IEEE Communications Surveys and Tutorials*, 15(3):1281–1293, 2013. ISSN 1553877X. doi: 10.1109/SURV.2012.121912.00075.

[103] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. Participatory Air Pollution Monitoring Using Smartphones. In *1st International Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, 2012.

[104] D. Hasenfratz, O. Saukh, and L. Thiele. On-the-fly calibration of low-cost gas sensors. In *Wireless Sensor Networks*, pages 228–244. Springer, 2012.

[105] M. Hazas, C. Kray, H. Gellersen, H. Agbota, G. Kortuem, and A. Krohn. A relative positioning system for co-located mobile devices. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys '05*, page 177, New York, New York, USA, June 2005. ACM Press. ISBN 1931971315. doi: 10.1145/1067170.1067190.

[106] J. Heer, N. Kong, and M. Agrawala. Sizing the horizon. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, New York, USA, Apr. 2009. ACM. ISBN 9781605582467. doi: 10.1145/1518701.1518897.

[107] J. Hightower, C. Vakili, G. Borriello, and R. Want. Design and Calibration of the SpotON Ad-Hoc Location Sensing System. Technical report, University of Washington, Department of Computer Science and Engineering, Seattle, WA, 2001.

[108] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. Toivonen. Time Series Segmentation for Context Recognition in Mobile Devices. In *ICDM*, pages 203–210, 2001.

[109] J. Hoey, Xiao Yang, E. Quintana, and J. Favela. LaCasa: Location and context-aware safety assistant. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pages 171–174, 2012.

[110] L. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl5, and H.-W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In G. Abowd, B. Brumitt, and S. Shafer, editors, *Ubicomp 2001: Ubiquitous Computing SE - 10*, volume 2201 of *Lecture Notes in Computer Science*, pages 116–122. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42614-1. doi: 10.1007/3-540-45427-6_10.

[111] F. Hong, Y. Zhang, Z. Zhang, M. Wei, Y. Feng, and Z. Guo. WaP: Indoor localization and tracking using WiFi-Assisted Particle filter. In *39th Annual IEEE Conference on Local*

Computer Networks, pages 210–217. IEEE, Sept. 2014. ISBN 978-1-4799-3780-6. doi: 10.1109/LCN.2014.6925774.

[112] J.-y. Hong, E.-h. Suh, and S.-J. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522, May 2009. ISSN 09574174. doi: 10.1016/j.eswa.2008.10.071.

[113] R. Honicky, E. A. Brewer, E. Paulos, and R. White. N-smarts. In *Proceedings of the second ACM SIGCOMM workshop on Networked systems for developing regions - NSDR '08*, page 25, New York, New York, USA, Aug. 2008. ACM Press. ISBN 9781605581804. doi: 10.1145/1397705.1397713.

[114] F. Höppner. Discovery of temporal patterns. Learning rules about the qualitative behaviour of time series. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '01, pages 192–203, London, UK, 2001. Springer-Verlag. ISBN 3-540-42534-9.

[115] S. A. Hoseini-Tabatabaei, A. Gluhak, and R. Tafazolli. A survey on smartphone-based systems for opportunistic user context recognition. *ACM Computing Surveys*, 45(3): 1–51, June 2013. ISSN 03600300. doi: 10.1145/2480741.2480744.

[116] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal. Efficient, generalized indoor WiFi GraphSLAM. In *2011 IEEE International Conference on Robotics and Automation*, pages 1038–1043. IEEE, May 2011. ISBN 978-1-61284-386-5. doi: 10.1109/ICRA.2011.5979643.

[117] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. SensorScope: Application-Specific Sensor Network for Environmental Monitoring. *TOSN*, 6 (2), 2010.

[118] R. Istepanian, E. Jovanov, and Y. Zhang. Guest Editorial Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity. *IEEE Transactions on Information Technology in Biomedicine*, 8(4):405–414, Dec. 2004. ISSN 1089-7771. doi: 10.1109/TITB.2004.840019.

[119] R. S. H. Istepanian, S. Hu, N. Y. Philip, and A. Sungoor. The potential of Internet of m-health Things "m-IoT" for non-invasive glucose level sensing. *Proceeding of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, 2011: 5264–6, Jan. 2011. ISSN 1557-170X. doi: 10.1109/IEMBS.2011.6091302.

[120] R. Jain and L. Jalali. Objective Self. *IEEE MultiMedia*, 21(4):100–110, Oct. 2014. ISSN 1070-986X. doi: 10.1109/MMUL.2014.63.

[121] R. Jain, D.-M. Chiu, and W. Hawe. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. Technical report, Digital Equipment Corp., Jan. 1984.

[122] P. Jayaraman, C. Perera, D. Georgakopoulos, and A. Zaslavsky. Efficient opportunistic sensing using mobile collaborative platform MOSDEN. In *9th International Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*, pages 77–86, Austin, TX, 2013.

[123] R. H. John Krumm. TempIO: Inside/Outside Classification with Temperature. In *Second International Workshop on Man-Machine Symbiotic Systems*, 2004.

[124] T. Johnson. A guide to selected algorithms, distributions, and databases used in exposure models developed by the office of air quality planning and standards. *Research Triangle Park, NC, US Environmental Protection Agency, Office of Research and Development*, 2002.

[125] T. Johnson, G. Mihlan, J. LaPointe, K. Fletcher, T. Environmental, A. Rosenbaum, J. Cohen, P. Stiefer, and H. Richmond. Estimation of carbon monoxide exposures and associated carboxyhemoglobin levels for residents of denver and los angeles using pnem/co (version 2.1). 2000.

[126] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song. SeeMon. In *Proceeding of the 6th international conference on Mobile systems, applications, and services - MobiSys '08*, page 267, New York, New York, USA, June 2008. ACM Press. ISBN 9781605581392. doi: 10.1145/1378600.1378630.

[127] E. Kanjo. NoiseSPY: A Real-Time Mobile Phone Platform for Urban Noise Monitoring and Mapping. *Mobile Networks and Applications*, 15(4):562–574, Nov. 2009. ISSN 1383-469X. doi: 10.1007/s11036-009-0217-y.

[128] P. G. Kannan, S. P. Venkatagiri, M. C. Chan, A. L. Ananda, and L.-S. Peh. Low cost crowd counting using audio tones. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems - SenSys '12*, page 155, New York, New York, USA, Nov. 2012. ACM Press. ISBN 9781450311694. doi: 10.1145/2426656.2426673.

[129] A. Kansal, S. Saponas, A. B. Brush, K. S. McKinley, T. Mytkowicz, and R. Ziola. The latency, accuracy, and battery (LAB) abstraction. In *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications - OOPSLA '13*, volume 48, pages 661–676, New York, New York, USA, Oct. 2013. ACM Press. ISBN 9781450323741. doi: 10.1145/2509136.2509541.

[130] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3(3):263–286, Aug. 2001. ISSN 0219-1377. doi: 10.1007/PL00011669.

[131] E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: implications for previous and future research. In *Third IEEE International Conference on Data Mining (ICDM)*, pages 115–122, 2003. doi: 10.1109/ICDM.2003.1250910.

[132] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting Time Series: A Survey and Novel Approach. In *Data mining in Time Series Databases. Published by World Scientific*, pages 1–22. Publishing Company, 2004.

[133] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An Online Algorithm for Segmenting Time Series. In *ICDM*, pages 289–296, 2001.

[134] S. A. Kharidia, Q. Ye, S. Sampalli, J. Cheng, H. Du, and L. Wang. HILL: A Hybrid Indoor Localization Scheme. In *2014 10th International Conference on Mobile Ad-hoc and Sensor Networks*, pages 201–206. IEEE, Dec. 2014. ISBN 978-1-4799-7394-1. doi: 10.1109/MSN.2014.34.

[135] N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-perez, and J. Laurila. Towards rich mobile phone datasets : Lausanne data collection campaign. *Proc ICPS Berlin*, 2002.

[136] M. B. Kjæ rgaard, S. Bhattacharya, H. Blunck, and P. Nurmi. Energy-efficient trajectory tracking for mobile devices. In *MobiSys '11*, pages 307–320, June 2011. ISBN 978-1-4503-0643-0. doi: 10.1145/1999995.2000025.

[137] P. Klasnja and W. Pratt. Healthcare in the pocket: mapping the space of mobile-phone health interventions. *Journal of biomedical informatics*, 45(1):184–98, Feb. 2012. ISSN 1532-0480. doi: 10.1016/j.jbi.2011.08.017.

[138] P. Klasnja, S. Consolvo, D. W. McDonald, J. A. Landay, and W. Pratt. Using mobile & personal sensing technologies to support health behavior change in everyday life: lessons learned. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, 2009:338–42, Jan. 2009. ISSN 1942-597X.

[139] J. Kohlmorgen and S. Lemm. An on-line method for segmentation and identification of non-stationary time series. In *Neural Networks for Signal Processing XI, 2001. Proceedings of the 2001 IEEE Signal Processing Society Workshop*, pages 113–122, 2001. doi: 10.1109/NNSP.2001.943116.

[140] J. Z. Kolter and M. J. Johnson. REDD: A public data set for energy disaggregation research. In *Proceedings of the SustKDD workshop on Data Mining Applications in Sustainability*, 2011.

[141] A. Krause and C. Guestrin. Optimizing Sensing: From Water to the Web. *Computer*, 42 (8):38–45, Aug. 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.265.

[142] A. Krause, A. P. Singh, and C. Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[143] A. Krause, R. Rajagopal, A. Gupta, and C. Guestrin. Simultaneous Optimization of Sensor Placements and Balanced Schedules. *IEEE Transactions on Automatic Control*, 56(10): 2390–2405, Oct. 2011. ISSN 0018-9286. doi: 10.1109/TAC.2011.2164010.

[144] J. Krosche, A. Jakl, D. Gusenbauer, D. Rothbauer, and B. Ehringer. Managing Context on a Sensor Enabled Mobile Device - The mSense Approach. In *2009 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 135–140. IEEE, Oct. 2009. ISBN 978-0-7695-3841-9. doi: 10.1109/WiMob.2009.32.

[145] J. Krumm and K. Hinckley. The NearMe Wireless Proximity Server. In N. Davies, E. Mynatt, and I. Siio, editors, *UbiComp 2004: Ubiquitous Computing SE - 17*, volume 3205 of *Lecture Notes in Computer Science*, pages 283–300. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22955-1. doi: 10.1007/978-3-540-30119-6_17.

[146] S. Kumar, W. J. Nilsen, A. Abernethy, A. Atienza, K. Patrick, M. Pavel, W. T. Riley, A. Shar, B. Spring, D. Spruijt-Metz, D. Hedeker, V. Honavar, R. Kravitz, R. C. Lefebvre, D. C. Mohr, S. A. Murphy, C. Quinn, V. Shusterman, and D. Swendeman. Mobile health technology evaluation: the mHealth evidence workshop. *American journal of preventive medicine*, 45(2):228–36, Aug. 2013. ISSN 1873-2607. doi: 10.1016/j.amepre.2013.03.017.

[147] Y.-S. Kuo, P. Pannuto, K.-J. Hsiao, and P. Dutta. Luxapose: indoor positioning with mobile phones and visible light. In *Proceedings of the 20th annual international conference on Mobile computing and networking - MobiCom '14*, pages 447–458, New York, New York, USA, Sept. 2014. ACM Press. ISBN 9781450327831. doi: 10.1145/2639108.2639109.

[148] C.-P. Lai, P.-C. Chung, and V. S. Tseng. A novel two-level clustering method for time series data analysis. *Expert Systems with Applications*, 37:6319–6326, 2010. ISSN 09574174. doi: 10.1016/j.eswa.2010.02.089.

[149] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, Sept. 2010. ISSN 0163-6804. doi: 10.1109/MCOM.2010.5560598.

[150] N. D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. T. Campbell. Bewell: A smartphone application to monitor, model and promote wellbeing. In *in Intl. ICST Conf. on Pervasive Computing Technologies for Healthcare*, 2011.

[151] N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11*, page 355, New York, New York, USA, Sept. 2011. ACM Press. ISBN 9781450306300. doi: 10.1145/2030112.2030160.

[152] O. D. Lara and M. A. Labrador. A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013. ISSN 1553-877X. doi: 10.1109/SURV.2012.110112.00192.

[153] O. D. Lara, A. J. Pérez, M. A. Labrador, and J. D. Posada. Centinela: A human activity recognition system based on acceleration and vital sign data. *Pervasive and Mobile Computing*, 8(5):717–729, Oct. 2012. ISSN 15741192. doi: 10.1016/j.pmcj.2011.06.004.

[154] E. C. Larson, T. Lee, S. Liu, M. Rosenfeld, and S. N. Patel. Accurate and privacy preserving cough sensing using a low-cost microphone. In *Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11*, page 375, New York, New York, USA, Sept. 2011. ACM Press. ISBN 9781450306300. doi: 10.1145/2030112.2030163.

[155] J. K. Laurila, D. Gatica-Perez, I. Aad, B. J., O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. Miettinen. The Mobile Data Challenge: Big Data for Mobile Computing Research. In *Pervasive Computing*, 2012.

[156] D. T. Lee and A. K. Lin. Computational Complexity of Art Gallery Problems. *IEEE Trans. on Information Theory*, 32(2):276–282, 1986.

[157] P. Leijdekkers and V. Gay. A Self-Test to Detect a Heart Attack Using a Mobile Phone and Wearable Sensors. In *2008 21st IEEE International Symposium on Computer-Based Medical Systems*, pages 93–98. IEEE, June 2008. ISBN 978-0-7695-3165-6. doi: 10.1109/CBMS.2008.59.

[158] D. Lemire. A better alternative to piecewise linear time series segmentation. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 545–550. doi: 10.1137/1.9781611972771.59.

[159] K. A. Li, T. Y. Sohn, S. Huang, and W. G. Griswold. Peopletones. In *Proceeding of the 6th international conference on Mobile systems, applications, and services - MobiSys '08*, page 160, New York, New York, USA, June 2008. ACM Press. ISBN 9781605581392. doi: 10.1145/1378600.1378619.

[160] X. Li, H. Cao, E. Chen, and J. Tian. Learning to Infer the Status of Heavy-Duty Sensors for Energy-Efficient Context-Sensing. *ACM Transactions on Intelligent Systems and Technology*, 3(2):1–23, Feb. 2012. ISSN 21576904. doi: 10.1145/2089094.2089111.

[161] L. Liao, D. Fox, and H. Kautz. Hierarchical conditional random fields for gps-based activity recognition. In S. Thrun, R. Brooks, and H. Durrant-Whyte, editors, *Robotics Research*, volume 28 of *Springer Tracts in Advanced Robotics*, pages 487–506. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-48110-2. doi: 10.1007/978-3-540-48113-3_41.

[162] R. LiKamWa, Y. Liu, N. D. Lane, and L. Zhong. MoodScope: Building a Mood Sensor from Smartphone Usage Patterns. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services - MobiSys '13*, page 389, New York, New York, USA, June 2013. ACM Press. ISBN 9781450316729. doi: 10.1145/2462456.2464449.

[163] J. H. Lim, A. Zhan, E. Goldschmidt, J. Ko, M. Chang, and A. Terzis. HealthOS: A Platform for Pervasive Health Applications. In *Proceedings of the Second ACM Workshop on Mobile Systems, Applications, and Services for HealthCare - mHealthSys '12*, page 1, New York, New York, USA, Nov. 2012. ACM Press. ISBN 9781450317641. doi: 10.1145/2396276.2396281.

[164] F. Lin and P. Chiu. A Near-Optimal Sensor Placement Algorithm to Achieve Complete Coverage-Discrimination in Sensor Networks. *IEEE Communications Letters*, 9(1):43 – 45, jan. 2005. ISSN 1089-7798. doi: 10.1109/LCOMM.2005.01027.

[165] F. X. Lin, A. Rahmati, and L. Zhong. Dandelion: A Framework for Transparently Programming Phone-Centered Wireless Body Sensor Applications for Health. In *Wireless Health 2010 on - WH '10*, page 74, New York, New York, USA, Oct. 2010. ACM Press. ISBN 9781605589893. doi: 10.1145/1921081.1921091.

[166] U. Lipowezky and I. Vol. Indoor-outdoor detector for mobile phone cameras using gentle boosting. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 31–38. IEEE, June 2010. ISBN 978-1-4244-7029-7. doi: 10.1109/CVPRW.2010.5543754.

[167] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of Wireless Indoor Positioning Techniques and Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, Nov. 2007. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.905750.

[168] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye. Push the limit of WiFi based localization for smartphones. In *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, page 305, New York, New York, USA, Aug. 2012. ACM Press. ISBN 9781450311595. doi: 10.1145/2348543.2348581.

[169] M. Liu. A study of mobile sensing using smartphones. *International Journal of Distributed Sensor Networks*, 2013, 2013. ISSN 15501329. doi: 10.1155/2013/272916.

[170] Q. LIU, H. MA, E. CHEN, and H. XIONG. A SURVEY OF CONTEXT-AWARE MOBILE RECOMMENDATIONS. *International Journal of Information Technology & Decision Making*, 12(01):139–172, Jan. 2013. ISSN 0219-6220. doi: 10.1142/S0219622013500077.

[171] S. Liu, R. X. Gao, D. John, J. W. Staudenmayer, and P. S. Freedson. Multisensor data fusion for physical activity assessment. *IEEE transactions on bio-medical engineering*, 59(3):687–96, Mar. 2012. ISSN 1558-2531. doi: 10.1109/TBME.2011.2178070.

[172] X. Liu, Z. Lin, and H. Wang. Novel Online Methods for Time Series Segmentation. *IEEE Trans. Knowl. Data Eng.*, 20(12):1616–1626, 2008.

[173] Y. Liu, Q. Wang, J. Liu, and T. Wark. MCMC-based indoor localization with a smart phone and sparse WiFi access points. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 247–252. IEEE, Mar. 2012. ISBN 978-1-4673-0907-3. doi: 10.1109/PerComW.2012.6197488.

[174] J. W. Lockhart, T. Pulickal, and G. M. Weiss. Applications of mobile activity recognition. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, page 1054, New York, New York, USA, Sept. 2012. ACM Press. ISBN 9781450312240. doi: 10.1145/2370216.2370441.

[175] B. Logan, J. Healey, M. Philipose, E. Tapia, and S. Intille. A long-term evaluation of sensing modalities for activity recognition. In J. Krumm, G. Abowd, A. Seneviratne, and T. Strang, editors, *UbiComp 2007: Ubiquitous Computing*, volume 4717 of *Lecture Notes in Computer Science*, pages 483–500. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74852-6. doi: 10.1007/978-3-540-74853-3_28.

[176] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The Jigsaw continuous sensing engine for mobile phone applications. pages 71–84, Nov. 2010. doi: 10.1145/1869983.1869992.

[177] H. Lu, A. J. Bernheim Brush, B. Priyantha, A. Karlson, and J. Liu. SpeakerSense: Energy Efficient Unobtrusive Speaker Identification on Mobile Phones. In K. Lyons, J. Hightower, and E. Huang, editors, *Pervasive Computing SE - 12*, volume 6696 of *Lecture Notes in Computer Science*, pages 188–205. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-21725-8. doi: 10.1007/978-3-642-21726-5_12.

[178] H. Lu, D. Frauendorfer, M. Rabbi, M. S. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, and T. Choudhury. StressSense: Detecting Stress in Unconstrained Acoustic Environments using Smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, page 351, New York, New York, USA, Sept. 2012. ACM Press. ISBN 9781450312240. doi: 10.1145/2370216.2370270.

[179] Y. Ma, B. Xu, Y. Bai, G. Sun, and R. Zhu. Daily Mood Assessment Based on Mobile Phone Sensing. In *2012 Ninth International Conference on Wearable and Implantable Body Sensor Networks*, pages 142–147. IEEE, May 2012. ISBN 978-0-7695-4698-8. doi: 10.1109/BSN.2012.3.

[180] E. Macias, A. Suarez, and J. Lloret. Mobile Sensing Systems. *Sensors*, 13(12):17292–17321, Dec. 2013. ISSN 1424-8220. doi: 10.3390/s131217292.

[181] R. Madarshahian and J. Caicedo. Human activity recognition using multinomial logistic regression. In H. S. Atamturktur, B. Moaveni, C. Papadimitriou, and T. Schoenherr, editors, *Model Validation and Uncertainty Quantification, Volume 3*, Conference Proceedings of the Society for Experimental Mechanics Series, pages 363–372. Springer International Publishing, 2015. ISBN 978-3-319-15223-3. doi: 10.1007/978-3-319-15224-0_38.

[182] N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, and L. Steels. Citizen noise pollution monitoring. In *Proceedings of the 10th Annual International Conference on Digital Government Research: Social Networks: Making Connections Between Citizens, Data and Government*, pages 96–103, Puebla, Mexico, May 2009. Digital Government Society of North America. ISBN 978-1-60558-535-2.

[183] N. Maisonneuve, M. Stevens, and B. Ochab. Participatory noise pollution monitoring using mobile phones. *Information Polity*, 15(1-2):51–71, 2010. ISSN 15701255. doi: 10.3233/IP-2010-0200.

[184] D. Majoe, P. Bonhof, T. Kaegi-Trachsel, J. Gutknecht, and L. Widmer. Stress and sleep quality estimation from a smart wearable sensor. In *5th International Conference on Pervasive Computing and Applications*, pages 14–19. IEEE, Dec. 2010. ISBN 978-1-4244-9144-5. doi: 10.1109/ICPCA.2010.5704068.

[185] Y. Man and E. C.-H. Ngai. Energy-efficient automatic location-triggered applications on smartphones. *Computer Communications*, Mar. 2014. ISSN 01403664. doi: 10.1016/j.comcom.2014.03.023.

[186] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy. Precise indoor localization using smart phones. In *Proceedings of the international conference on Multimedia - MM '10*, page 787, New York, New York, USA, Oct. 2010. ACM Press. ISBN 9781605589336. doi: 10.1145/1873951.1874078.

[187] F. Martinez Alvarez, A. Troncoso, J. Riquelme, and J. Aguilar Ruiz. Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1230–1243, 2011. ISSN 1041-4347. doi: 10.1109/TKDE.2010.227.

[188] A. Matic, V. Osmani, and O. Mayora-Ibarra. Analysis of Social Interactions Through Mobile Phones. *Mobile Networks and Applications*, 17(6):808–819, 2012. ISSN 1383-469X. doi: 10.1007/s11036-012-0400-4.

[189] U. Maurer, A. Smailagic, D. Siewiorek, and M. Deisher. Activity Recognition and Monitoring Using Multiple Sensors on Different Body Positions. In *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, pages 113–116. IEEE, 2006. ISBN 0-7695-2547-4. doi: 10.1109/BSN.2006.6.

[190] 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 '08*, page 337, New York, New York, USA, Nov. 2008. ACM Press. ISBN 9781595939906. doi: 10.1145/1460412.1460445.

[191] A. Mishra, D. Irwin, P. Shenoy, and T. Zhu. Scaling distributed energy storage for grid peak reduction. In *Proceedings of the Fourth International Conference on Future Energy Systems*, e-Energy '13, pages 3–14, New York, USA, 2013. ACM. ISBN 978-1-4503-2052-8. doi: 10.1145/2487166.2487168.

[192] A. Morabia, P. N. Amstislavski, F. E. Mirer, T. M. Amstislavski, H. Eisl, M. S. Wolff, and S. B. Markowitz. Air pollution and activity during transportation by car, subway, and walking. *American journal of preventive medicine*, 37(1):72–7, July 2009. ISSN 1873-2607. doi: 10.1016/j.amepre.2009.03.014.

[193] M. Morris and F. Guilak. Mobile Heart Health: Project Highlight. *IEEE Pervasive Computing*, 8(2):57–61, Apr. 2009. ISSN 1536-1268. doi: 10.1109/MPRV.2009.31.

[194] B. Motnikar, D. Čepar, P. Žunko, M. Ribarič, and B. Vovk. Time series forecasting by imitation of preceding patterns. In P. Gritzmann, R. Hettich, R. Horst, and E. Sachs, editors, *Operations Research'91*, pages 272–275. Physica-Verlag HD, 1992. ISBN 978-3-7908-0608-3. doi: 10.1007/978-3-642-48417-9_75.

[195] A. Muaremi, B. Arnrich, and G. Tröster. A Survey on Measuring Happiness with Smart Phones. In *6th International Workshop on Ubiquitous Health and Wellness (UbiHealth)*, 2012.

[196] A. Mueen and E. Keogh. Online discovery and maintenance of time series motifs. In *Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 1089–1098, New York, USA, 2010. ACM. ISBN 978-1-4503-0055-1. doi: 10.1145/1835804.1835941.

[197] M. Mun, P. Boda, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, and R. West. 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 '09*, page 55, New York, New York, USA, June 2009. ACM Press. ISBN 9781605585666. doi: 10.1145/1555816.1555823.

[198] F. M. Naini, O. Dousse, P. Thiran, and M. Vetterli. Population size estimation using a few individuals as agents. In *2011 IEEE International Symposium on Information Theory Proceedings*, pages 2499–2503. IEEE, July 2011. ISBN 978-1-4577-0596-0. doi: 10.1109/ISIT.2011.6034016.

[199] T. Nakakura, Y. Sumi, and T. Nishida. Neary. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications - HotMobile '09*, pages 1–6, New York, New York, USA, Feb. 2009. ACM Press. ISBN 9781605582832. doi: 10.1145/1514411.1514423.

[200] S. Nath. ACE: Exploiting Correlation for Energy-Efficient and Continuous Context Sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*, page 29, New York, New York, USA, June 2012. ACM Press. ISBN 9781450313018. doi: 10.1145/2307636.2307640.

[201] E. C.-H. Ngai and J. Xiong. Adaptive collaborative sensing using mobile phones and stationary sensors. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 280–285. IEEE, June 2011. ISBN 978-1-4577-0374-4. doi: 10.1109/DSNW.2011.5958782.

[202] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. LANDMARC: Indoor Location Sensing Using Active RFID. *Wireless Networks*, 10(6):701–710, Nov. 2004. ISSN 1022-0038. doi: 10.1023/B:WINE.0000044029.06344.dd.

[203] E. Niforatos, P. Campos, A. Vourvopoulos, A. Doria, and M. Langheinrich. Atmos. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct*, pages 135–138, New York, New York, USA, Sept. 2014. ACM Press. ISBN 9781450330473. doi: 10.1145/2638728.2638780.

[204] S. Nirjon, F. Zhao, R. F. Dickerson, Q. Li, P. Asare, J. A. Stankovic, D. Hong, B. Zhang, X. Jiang, and G. Shen. MusicalHeart: A Hearty Way of Listening to Music. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems - SenSys '12*, page 43, New York, New York, USA, Nov. 2012. ACM Press. ISBN 9781450311694. doi: 10.1145/2426656.2426662.

[205] N. Oliver and F. Flores-Mangas. MPTrain: A Mobile, Music and Physiology-Based Personal Trainer. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services - MobileHCI '06*, page 21, New York, New York, USA, Sept. 2006. ACM Press. ISBN 1595933905. doi: 10.1145/1152215.1152221.

[206] N. Oliver and F. Flores-Mangas. HealthGear: Automatic Sleep Apnea Detection and Monitoring with a Mobile Phone. *Journal of Communications*, 2(2):1–9, Mar. 2007. ISSN 1796-2021. doi: 10.4304/jcm.2.2.1-9.

[207] J. J. Oresko, H. Duschl, and A. C. Cheng. A wearable smartphone-based platform for real-time cardiovascular disease detection via electrocardiogram processing. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 14(3):734–40, May 2010. ISSN 1558-0032. doi: 10.1109/TITB.2010.2047865.

[208] A. Overeem, J. C. R. Robinson, H. Leijnse, G. J. Steeneveld, B. K. P. Horn, and R. Uijlenhoet. Crowdsourcing urban air temperatures from smartphone battery temperatures. *Geophysical Research Letters*, 40(15):4081–4085, Aug. 2013. ISSN 00948276. doi: 10.1002/grl.50786.

[209] 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 '10*, page 299, New York, New York, USA, June 2010. ACM Press. ISBN 9781605589855. doi: 10.1145/1814433.1814463.

[210] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter. Mobile crowd sensing in space weather monitoring: the mahali project. *IEEE Communications Magazine*, 52(8): 22–28, Aug. 2014. ISSN 0163-6804. doi: 10.1109/MCOM.2014.6871665.

[211] A. Pantelopoulos and N. Bourbakis. A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):1–12, Jan. 2010. ISSN 1094-6977. doi: 10.1109/TSMCC.2009.2032660.

[212] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 697–708, Trondheim, Norway, Aug. 2005. VLDB Endowment. ISBN 1-59593-154-6.

[213] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen. Activity Classification Using Realistic Data From Wearable Sensors. *IEEE Transactions on In-*

formation Technology in Biomedicine, 10(1):119–128, Jan. 2006. ISSN 1089-7771. doi: 10.1109/TITB.2005.856863.

[214] S. Patel, J. Kientz, G. Hayes, S. Bhat, and G. Abowd. Farther Than You May Think: An Empirical Investigation of the Proximity of Users to Their Mobile Phones. In P. Dourish and A. Friday, editors, *UbiComp 2006: Ubiquitous Computing SE - 8*, volume 4206 of *Lecture Notes in Computer Science*, pages 123–140. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-39634-5. doi: 10.1007/11853565_8.

[215] N. Pathak, M. A. A. H. Khan, and N. Roy. Acoustic Based Appliance State Identifications for Fine Grained Energy Analytics. In *2015 IEEE International Conference on Pervasive Computing and Communications*, pages 63–71, 2015.

[216] A. Payne and S. Singh. Indoor vs. outdoor scene classification in digital photographs. *Pattern Recognition*, 38(10):1533–1545, Oct. 2005. ISSN 00313203. doi: 10.1016/j.patcog.2004.12.014.

[217] C. G. Pendao, A. C. Moreira, and H. Rodrigues. Energy consumption in personal mobile devices sensing applications. In *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 1–8. IEEE, May 2014. ISBN 978-1-4799-3060-9. doi: 10.1109/WMNC.2014.6878874.

[218] C. Peng, G. Shen, and Y. Zhang. BeepBeep. *ACM Transactions on Embedded Computing Systems*, 11(1):1–29, Mar. 2012. ISSN 15399087. doi: 10.1145/2146417.2146421.

[219] H. Peng, F. Long, and C. Ding. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226 –1238, aug. 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.159.

[220] D. E. Phillips, R. Tan, M.-M. Moazzami, G. Xing, J. Chen, and D. K. Yau. Supero: A sensor system for unsupervised residential power usage monitoring. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 66–75. IEEE, 2013.

[221] K. Plarre, A. Raij, S. Hossain, A. Ali, M. Nakajima, M. Al'Absi, E. Ertin, T. Kamarck, S. Kumar, M. Scott, D. Siewiorek, A. Smailagic, and L. Wittmers. Continuous inference of psychological stress from sensory measurements collected in the natural environment. In *10th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 97–108, 2011.

[222] I. Podnar Zarko, A. Antonic, and K. Pripužic. Publish/subscribe middleware for energy-efficient mobile crowdsensing. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication - UbiComp '13 Adjunct*, pages 1099–1110, New York, New York, USA, Sept. 2013. ACM Press. ISBN 9781450322157. doi: 10.1145/2494091.2499577.

[223] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, June 2010. ISSN 02628856. doi: 10.1016/j.imavis.2009.11.014.

[224] A. Poshtkar, V. Elangovan, A. Shirkhodaie, A. Chan, and S. Hu. Physical environment virtualization for human activities recognition. In E. J. Kelmelis, editor, *SPIE Defense + Security*, page 94780I. International Society for Optics and Photonics, May 2015. doi: 10.1117/12.2178547.

[225] O. Postolache, P. S. Girao, M. Ribeiro, M. Guerra, J. Pincho, F. Santiago, and A. Pena. Enabling telecare assessment with pervasive sensing and Android OS smartphone. In *2011 IEEE International Symposium on Medical Measurements and Applications*, pages 288–293. IEEE, May 2011. ISBN 978-1-4244-9336-4. doi: 10.1109/MeMeA.2011.5966761.

[226] B. Predic, Z. Yan, J. Eberle, D. Stojanovic, and K. Aberer. ExposureSense: Integrating Daily Activities with Air Quality using Mobile Participatory Sensing. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 303–305, March 2013. doi: 10.1109/PerComW.2013.6529500.

[227] B. Priyantha, D. Lymberopoulos, and Jie Liu. LittleRock: Enabling Energy-Efficient Continuous Sensing on Mobile Phones. *IEEE Pervasive Computing*, 10(2):12–15, Apr. 2011. ISSN 1536-1268. doi: 10.1109/MPRV.2011.28.

[228] K. Rachuri, C. Mascolo, and M. Musolesi. Energy-Accuracy Trade-offs of Sensor Sampling in Smart Phone Based Sensing Systems. In T. Lovett and E. O'Neill, editors, *Mobile Context Awareness SE - 3*, pages 65–76. Springer London, 2012. ISBN 978-0-85729-624-5. doi: 10.1007/978-0-85729-625-2_3.

[229] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. EmotionSense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research. In *Proceedings of the 12th ACM international conference on Ubiquitous computing - Ubicomp '10*, page 281, New York, New York, USA, Sept. 2010. ACM Press. ISBN 9781605588438. doi: 10.1145/1864349.1864393.

[230] K. K. Rachuri, C. Mascolo, and P. J. Rentfrow. SociableSense : Exploring the Trade-offs of Adaptive Sampling and Computation Offloading for Social Sensing Categories and Subject Descriptors. In *MobiCom '11*, MobiCom '11, pages 73–84. ACM, 2011. ISBN 9781450304924. doi: 10.1145/2030613.2030623.

[231] K. K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P. J. Rentfrow. METIS: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 85–93. IEEE, Mar. 2013. ISBN 978-1-4673-4575-0. doi: 10.1109/PerCom.2013.6526718.

[232] I. Rafael, L. Duarte, L. Carriço, and T. Guerreiro. Towards ubiquitous awareness tools for blind people. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*, pages 38:1–38:5, London, UK, Sept. 2013. British Computer Society.

[233] T. Rahman, A. T. Adams, M. Zhang, E. Cherry, B. Zhou, H. Peng, and T. Choudhury. BodyBeat: A Mobile System for Sensing Non-Speech Body Sounds. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys '14*, pages 2–13, New York, New York, USA, June 2014. ACM Press. ISBN 9781450327930. doi: 10.1145/2594368.2594386.

[234] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, page 293, New York, New York, USA, Aug. 2012. ACM Press. ISBN 9781450311595. doi: 10.1145/2348543.2348580.

[235] T. Rakthanmanon, E. Keogh, S. Lonardi, and S. Evans. Time series epenthesis: Clustering time series streams requires ignoring some data. In *IEEE 11th International Conference on Data Mining (ICDM)*, pages 547–556, Dec 2011. doi: 10.1109/ICDM.2011.146.

[236] M. Ramoni, P. Sebastiani, and P. Cohen. Bayesian Clustering by Dynamics. *Machine Learning*, 47(1):91–121, Apr. 2002. ISSN 1573-0565. doi: 10.1023/A:1013635829250.

[237] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Ear-phone. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks - IPSN '10*, page 105, New York, New York, USA, Apr. 2010. ACM Press. ISBN 9781605589886. doi: 10.1145/1791212.1791226.

[238] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Ear-Phone: an End-To-End Participatory Urban Noise Mapping System. In *IPSN*, pages 105–116, 2010.

[239] R. Raskar, P. Beardsley, J. van Baar, Y. Wang, P. Dietz, J. Lee, D. Leigh, and T. Willwacher. RFIG lamps: interacting with a self-describing world via photosensing wireless tags and projectors. *ACM Transactions on Graphics*, 23(3):406, Aug. 2004. ISSN 07300301. doi: 10.1145/1015706.1015738.

[240] L. Ravindranath, A. Thiagarajan, H. Balakrishnan, and S. Madden. Code in the air: Simplifying Sensing and Coordination Tasks on Smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications - HotMobile '12*, page 1, New York, New York, USA, Feb. 2012. ACM Press. ISBN 9781450312073. doi: 10.1145/2162081.2162087.

[241] R. Rawassizadeh, M. Tomitsch, K. Wac, and A. M. Tjoa. UbiqLog: a generic mobile phone-based life-log framework. *Personal and Ubiquitous Computing*, 17(4):621–637, Apr. 2012. ISSN 1617-4909. doi: 10.1007/s00779-012-0511-8.

[242] 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*, 6 (2):1–27, Feb. 2010. ISSN 15504859. doi: 10.1145/1689239.1689243.

[243] M. Riahi, T. G. Papaioannou, I. Trummer, and K. Aberer. Utility-driven data acquisition in participatory sensing. In *Proceedings of the 16th International Conference on Extending*

Database Technology - EDBT '13, page 251, New York, New York, USA, Mar. 2013. ACM Press. ISBN 9781450315975. doi: 10.1145/2452376.2452407.

[244] D. Riboni and C. Bettini. COSAR: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 15(3):271–289, Aug. 2010. ISSN 1617-4909. doi: 10.1007/s00779-010-0331-7.

[245] J. S. Richman and J. R. Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *Am J Physiol Heart Circ Physiol*, 278(6):H2039–2049, June 2000.

[246] M. Rohs and P. Zweifel. A Conceptual Framework for Camera Phone-Based Interaction Techniques. In H.-W. Gellersen, R. Want, and A. Schmidt, editors, *Pervasive Computing SE - 11*, volume 3468 of *Lecture Notes in Computer Science*, pages 171–189. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26008-0. doi: 10.1007/11428572_11.

[247] N. Roy, H. Wang, and R. Roy Choudhury. I am a smartphone and i can tell my user's walking direction. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys '14*, pages 329–342, New York, New York, USA, June 2014. ACM Press. ISBN 9781450327930. doi: 10.1145/2594368.2594392.

[248] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh. Using mobile phone barometer for low-power transportation context detection. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems - SenSys '14*, pages 191–205, New York, New York, USA, Nov. 2014. ACM Press. ISBN 9781450331432. doi: 10.1145/2668332.2668343.

[249] A. Sano and R. W. Picard. Stress Recognition Using Wearable Sensors and Mobile Phones. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 671–676. IEEE, Sept. 2013. ISBN 978-0-7695-5048-0. doi: 10.1109/ACII.2013.117.

[250] S. Santini, B. Ostermaier, and R. Adelmann. On the use of sensor nodes and mobile phones for the assessment of noise pollution levels in urban environments. In *2009 Sixth International Conference on Networked Sensing Systems (INSS)*, pages 1–8. IEEE, June 2009. ISBN 978-1-4244-6313-8. doi: 10.1109/INSS.2009.5409957.

[251] O. Saukh, D. Hasenfratz, A. Noori, T. Ulrich, and L. Thiele. Route selection for mobile sensors with checkpointing constraints. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 266–271. IEEE, Mar. 2012. ISBN 978-1-4673-0907-3. doi: 10.1109/PerComW.2012.6197492.

[252] M. Schirmer and H. Höpfner. SenST*: Approaches for Reducing the Energy Consumption of Smartphone-Based Context Recognition. In M. Beigl, H. Christiansen, T. Roth-Berghofer, A. Kofod-Petersen, K. Coventry, and H. Schmidtke, editors, *Modeling and Using Context SE - 27*, volume 6967 of *Lecture Notes in Computer Science*,

pages 250–263. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24278-6. doi: 10.1007/978-3-642-24279-3_27.

[253] D. Schuster, A. Rosi, M. Mamei, T. Springer, M. Endler, and F. Zambonelli. Pervasive social context. *ACM Transactions on Intelligent Systems and Technology*, 4(3):1, June 2013. ISSN 21576904. doi: 10.1145/2483669.2483679.

[254] C. G. Scully, J. Lee, J. Meyer, A. M. Gorbach, D. Granquist-Fraser, Y. Mendelson, and K. H. Chon. Physiological parameter monitoring from optical recordings with a mobile phone. *IEEE transactions on bio-medical engineering*, 59(2):303–6, Mar. 2012. ISSN 1558-2531. doi: 10.1109/TBME.2011.2163157.

[255] C. Seeger, A. Buchmann, and K. Van Laerhoven. An event-based BSN middleware that supports seamless switching between sensor configurations. In *Proceedings of the 2nd ACM SIGHIT symposium on International health informatics - IHI '12*, page 503, New York, New York, USA, Jan. 2012. ACM Press. ISBN 9781450307819. doi: 10.1145/2110363.2110420.

[256] J. Seitz, T. Vaupel, J. Jahn, S. Meyer, J. Boronat, and J. Thielecke. A Hidden Markov Model for urban navigation based on fingerprinting and pedestrian dead reckoning. *Information Fusion (FUSION), 2010 13th Conference on*, pages 1–8, 2010.

[257] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka. You are facing the Mona Lisa. In *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*, page 183, New York, New York, USA, June 2012. ACM Press. ISBN 9781450313018. doi: 10.1145/2307636.2307654.

[258] S. Sen, J. Lee, K.-H. Kim, and P. Congdon. Avoiding multipath to revive inbuilding WiFi localization. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services - MobiSys '13*, page 249, New York, New York, USA, June 2013. ACM Press. ISBN 9781450316729. doi: 10.1145/2462456.2464463.

[259] SGXSensortech. Datasheet MiCS-2614, 2014. URL http://www.sgxsensortech.com/content/uploads/2014/07/1087_Datasheet-MiCS-2614-rev-6.pdf~[21.07.2015].

[260] P. Sharma, D. Chakraborty, N. Banerjee, D. Banerjee, S. K. Agarwal, and S. Mittal. KARMA: Improving WiFi-based indoor localization with dynamic causality calibration. In *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 90–98. IEEE, June 2014. ISBN 978-1-4799-4657-0. doi: 10.1109/SAHCN.2014.6990331.

[261] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 428–439. Morgan Kaufmann Publishers Inc., Aug. 1998. ISBN 1-55860-566-5.

[262] W. Shen, V. Babushkin, Z. Aung, and W. L. Woon. An ensemble model for day-ahead electricity demand time series forecasting. In *Proceedings of the fourth International Conference on Future Energy Systems*, e-Energy '13, pages 51–62, New York, USA, 2013. ACM. ISBN 978-1-4503-2052-8. doi: 10.1145/2487166.2487173.

[263] X. Sheng and J. Tang. Energy-Efficient Collaborative Sensing with Mobile Phones. In *INFOCOM*, pages 1916–1924, 2012. ISBN 9781467307758.

[264] X. Sheng, X. Xiao, J. Tang, and G. Xue. Sensing as a service: A cloud computing system for mobile phone sensing. In *2012 IEEE Sensors*, pages 1–4. IEEE, Oct. 2012. ISBN 978-1-4577-1767-3. doi: 10.1109/ICSENS.2012.6411516.

[265] K. Shimizu, M. Iwai, and K. Sezaki. Social Link Analysis Using Wireless Beaconing and Accelerometer. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 33–38. IEEE, Mar. 2013. ISBN 978-1-4673-6239-9. doi: 10.1109/WAINA.2013.141.

[266] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and Fei Lung Wong. SenSay: a context-aware mobile phone. In *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.*, pages 248–249. IEEE, Oct. 2003. ISBN 0-7695-2034-0. doi: 10.1109/ISWC.2003.1241422.

[267] B. Sig. BLUETOOTH SPECIFICATION Version 4.0 [Vol 3]. Specification p.330, Bluetooth SIG, 2010. URL https://www.bluetooth.org/en-us/specification/adopted-specifications~[21.07.2015].

[268] S. Sigg, U. Blanke, and G. Troster. The telepathic phone: Frictionless activity recognition from WiFi-RSSI. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 148–155. IEEE, Mar. 2014. ISBN 978-1-4799-3445-4. doi: 10.1109/PerCom.2014.6813955.

[269] S. Sigg, M. Scholz, and M. Beigl. RF-Sensing of Activities from Non-Cooperative Subjects in Device-Free Recognition Systems Using Ambient and Local Signals. *IEEE Transactions on Mobile Computing*, 13(4):907–920, Apr. 2014. ISSN 1536-1233. doi: 10.1109/TMC.2013.28.

[270] B. W. Silverman. *Density estimation for statistics and data analysis.* Chapman and Hall, London New York, 1986. ISBN 0412246201.

[271] R. Srinivasan, C. Wang, W. Ho, and K. Lim. Context-based recognition of process states using neural networks. *Chemical Engineering Science*, 60(4):935–949, Feb. 2005. ISSN 00092509. doi: 10.1016/j.ces.2004.09.061.

[272] S. Steinle, S. Reis, and C. E. Sabel. Quantifying human exposure to air pollution–moving from static monitoring to spatio-temporally resolved personal exposure assessment. *The Science of the total environment*, 443:184–93, Jan. 2013. ISSN 1879-1026. doi: 10.1016/j.scitotenv.2012.10.098.

[273] M. Swan. Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0. *Journal of Sensor and Actuator Networks*, 1(3):217–253, Nov. 2012. ISSN 2224-2708. doi: 10.3390/jsan1030217.

[274] M. Szummer and R. Picard. Indoor-outdoor image classification. In *Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database*, pages 42–51. IEEE Comput. Soc, 1998. ISBN 0-8186-8329-5. doi: 10.1109/CAIVD.1998.646032.

[275] J. S. Tan, Z. F. Kuang, and G. G. Yang. An Efficient Frequent Patterns Mining Algorithm over Data Streams Based on FPD-Graph. *Advanced Materials Research*, 433-440:4457–4462, Feb. 2012. ISSN 1662-8985.

[276] J. Tang and W. Zhang. Energy-efficient collaborative sensing with mobile phones. In *2012 Proceedings IEEE INFOCOM*, pages 1916–1924. IEEE, Mar. 2012. ISBN 978-1-4673-0775-8. doi: 10.1109/INFCOM.2012.6195568.

[277] N. Thepvilojanapong, S. Konomi, Y. Tobe, Y. Ohta, M. Iwai, and K. Sezaki. Opportunistic collaboration in participatory sensing environments. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture - MobiArch '10*, page 39, New York, New York, USA, Sept. 2010. ACM Press. ISBN 9781450301435. doi: 10.1145/1859983.1859994.

[278] A. Tinka, I. Strub, Q. Wu, and A. M. Bayen. Quadratic Programming based data assimilation with passive drifting sensors for shallow water flows. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 7614–7620. IEEE, Dec. 2009. ISBN 978-1-4244-3871-6. doi: 10.1109/CDC.2009.5399663.

[279] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In J. Redi, H. Balakrishnan, and F. Zhao, editors, *Proceedings of the 3rd international conference on Embedded networked sensor systems SenSys 05*, volume 3 of *SenSys '05*, pages 51–63. ACM Press, 2005. ISBN 159593054X. doi: 10.1145/1098918.1098925.

[280] E. Toye, R. Sharp, A. Madhavapeddy, D. Scott, E. Upton, and A. Blackwell. Interacting with mobile services: an evaluation of camera-phones and visual tags. *Personal and Ubiquitous Computing*, 11(2):97–106, Feb. 2006. ISSN 1617-4909. doi: 10.1007/s00779-006-0064-9.

[281] T. Tran, H. Bui, and S. Venkatesh. Human Activity Learning and Segmentation using Partially Hidden Discriminative Models. In *HAREM 2005 : Proceedings of the International Workshop on Human Activity Recognition and Modelling*, pages 87–95, Oxford, U.K., Aug. 2005. The Conference, HAREM 2005 in conjunction with BMVC 2005. URL http://arxiv.org/abs/1408.3081.

[282] V. Trifa, S. Wiel, D. Guinard, and T. Bohnert. Design and Implementation of a Gateway for Web-based Interaction and Management of Embedded Devices. In *Proceedings of the 2nd International Workshop on Sensor Network Engineering (IWSNE 09)*, pages 1–14, Marina del Rey, CA, USA, 2009.

[283] D. Trossen and D. Pavel. Airs: A mobile sensing platform for lifestyle management research and applications. In C. Borcea, P. Bellavista, C. Giannelli, T. Magedanz, and F. Schreiner, editors, *Mobile Wireless Middleware, Operating Systems, and Applications*, volume 65 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 1–15. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-36659-8. doi: 10.1007/978-3-642-36660-4_1.

[284] D.-H. Tsai, N. Amyai, P. Marques-Vidal, J.-L. Wang, M. Riediker, V. Mooser, F. Paccaud, G. Waeber, P. Vollenweider, and M. Bochud. Effects of particulate matter on inflammatory markers in the general adult population. *Particle and fibre toxicology*, 9(1):24, Jan. 2012. ISSN 1743-8977. doi: 10.1186/1743-8977-9-24.

[285] D.-H. Tsai, M. Riediker, G. Wuerzner, M. Maillard, P. Marques-Vidal, F. Paccaud, P. Vollenweider, M. Burnier, and M. Bochud. Short-term increase in particulate matter blunts nocturnal blood pressure dipping and daytime urinary sodium excretion. *Hypertension*, 60 (4):1061–9, Oct. 2012. ISSN 1524-4563. doi: 10.1161/HYPERTENSIONAHA.112.195370.

[286] N. Vallina-Rodriguez and J. Crowcroft. The Case for Context-Aware Resources Management in Mobile Operating Systems. In T. Lovett and E. O'Neill, editors, *Mobile Context Awareness SE - 6*, pages 97–113. Springer London, 2012. ISBN 978-0-85729-624-5. doi: 10.1007/978-0-85729-625-2_6.

[287] W. Van den Broeck, C. Cattuto, A. Barrat, M. Szomszor, G. Correndo, and H. Alani. The Live Social Semantics application: a platform for integrating face-to-face presence with on-line social networking. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 226–231. IEEE, Mar. 2010. ISBN 978-1-4244-6605-4. doi: 10.1109/PERCOMW.2010.5470665.

[288] H. Vogt. Multiple object identification with passive RFID tags. In *IEEE International Conference on Systems, Man and Cybernetics*, volume vol.3, page 6. IEEE, 2002. ISBN 0-7803-7437-1. doi: 10.1109/ICSMC.2002.1176119.

[289] P. Vytelingum, T. D. Voice, S. D. Ramchurn, A. Rogers, and N. R. Jennings. Agent-based micro-storage management for the smart grid. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10, pages 39–46, 2010. ISBN 978-0-9826571-1-9.

[290] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No need to war-drive. In *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*, page 197, New York, New York, USA, June 2012. ACM Press. ISBN 9781450313018. doi: 10.1145/2307636.2307655.

[291] K.-C. Wang, P. Ramanathan, V. Prasanna, S. Iyengar, P. Spirakis, and M. Welsh. *Distributed Computing in Sensor Systems - Collaborative Sensing Using Sensors of Uncoordinated Mobility*, volume 3560 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-26422-4. doi: 10.1007/b137498.

[292] P. Wang, H. Wang, and W. Wang. Finding semantics in time series. In *Proceedings of the International Conference on Management of Mata - SIGMOD '11*, pages 385–396, New York, USA, June 2011. ACM Press. ISBN 9781450306614. doi: 10.1145/1989323.1989364.

[293] W. Wang, L. Yu, H. Liu, and F. Sun. Extreme learning machine for linear dynamical systems classification: Application to human activity recognition. In J. Cao, K. Mao, E. Cambria, Z. Man, and K.-A. Toh, editors, *Proceedings of ELM-2014 Volume 2*, volume 4 of *Proceedings in Adaptation, Learning and Optimization*, pages 11–20. Springer International Publishing, 2015. ISBN 978-3-319-14065-0. doi: 10.1007/978-3-319-14066-7_2.

[294] X. Wang, X. Wang, and J. Zhao. Impact of Mobility and Heterogeneity on Coverage and Energy Consumption in Wireless Sensor Networks. In *2011 31st International Conference on Distributed Computing Systems*, pages 477–487. IEEE, 2011. ISBN 9780769543642. doi: 10.1109/ICDCS.2011.17.

[295] 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 '09*, page 179, New York, New York, USA, June 2009. ACM Press. ISBN 9781605585666. doi: 10.1145/1555816.1555835.

[296] Y. Wang, B. Krishnamachari, Q. Zhao, and M. Annavaram. Markov-optimal sensing policy for user state estimation in mobile devices. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks - IPSN '10*, page 268, New York, New York, USA, Apr. 2010. ACM Press. ISBN 9781605589886. doi: 10.1145/1791212.1791244.

[297] D. Wei, M. Uchida, S. Ding, and M. Cohen. A mobile phone-based wearable vital signs monitoring system. In *The Fifth International Conference on Computer and Information Technology (CIT'05)*, pages 950–955. IEEE, 2005. ISBN 0-7695-2432-X. doi: 10.1109/CIT.2005.19.

[298] H. Weinschrott, F. Durr, and K. Rothermel. StreamShaper: Coordination algorithms for participatory mobile urban sensing. In *The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)*, pages 195–204. IEEE, Nov. 2010. ISBN 978-1-4244-7488-2. doi: 10.1109/MASS.2010.5663996.

[299] M. Werner, M. Kessel, and C. Marouane. Indoor positioning using smartphone camera. In *2011 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–6. IEEE, Sept. 2011. ISBN 978-1-4577-1805-2. doi: 10.1109/IPIN.2011.6071954.

[300] J. Wiese, T. S. Saponas, and A. B. Brush. Phoneprioception. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*, page 2157, New York, New York, USA, Apr. 2013. ACM Press. ISBN 9781450318990. doi: 10.1145/2470654.2481296.

[301] T. K. Wijaya, J. Eberle, and K. Aberer. Symbolic representation of smart meter data. In *Proceedings of the Joint EDBT/ICDT Workshops*, EDBT '13, pages 242–248, New York, USA, 2013. ACM. ISBN 978-1-4503-1599-9. doi: 10.1145/2457317.2457357.

[302] T. K. Wijaya, T. Ganu, D. Chakraborty, K. Aberer, and D. P. Seetharam. Consumer segmentation and knowledge extraction from smart meter and survey data. In *SIAM International Conference on Data Mining (SDM14)*, 2014.

[303] T. K. Wijaya, M. Vasirani, and K. Aberer. When bias matters: An economic assessment of demand response baselines for residential customers. *IEEE Transactions on Smart Grid*, 5(4):1755–1763, July 2014. ISSN 1949-3053. doi: 10.1109/TSG.2014.2309053.

[304] P. Wu, J. Zhu, and J. Y. Zhang. MobiSens: A Versatile Mobile Sensing Platform for Real-World Applications. *Mobile Networks and Applications*, 18(1):60–80, Nov. 2012. ISSN 1383-469X. doi: 10.1007/s11036-012-0422-y.

[305] D. Wyatt, T. Choudhury, J. Bilmes, and J. A. Kitts. Inferring colocation and conversation networks from privacy-sensitive audio with implications for computational social science. *ACM Transactions on Intelligent Systems and Technology*, 2(1):1–41, Jan. 2011. ISSN 21576904. doi: 10.1145/1889681.1889688.

[306] C. Xu, S. Li, G. Liu, and Y. Zhang. Crowd ++: Unsupervised Speaker Count with Smartphones. In *Ubicomp'13*, pages 43–52, 2013. ISBN 9781450317702. doi: 10.1145/2493432.2493435.

[307] W. Xu, R. Chen, T. Chu, L. Kuang, Y. Yang, X. Li, J. Liu, and Y. Chen. A context detection approach using GPS module and emerging sensors in smartphone platform. In *2014 Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, pages 156–163. IEEE, Nov. 2014. ISBN 978-1-4799-6004-0. doi: 10.1109/UPINLBS.2014.7033723.

[308] T. Yamabe and T. Nakajima. Possibilities and Limitations of Context Extraction in Mobile Devices: Experiments with a Multi-sensory Personal Device. *Int. J. Multimed. Ubiquitous Eng.*, 4:37–52, 2009.

[309] Z. Yan, J. Eberle, and K. Aberer. Optimos: Optimal sensing for mobile sensors. In *IEEE 13th International Conference on Mobile Data Management (MDM)*, pages 105–114, July 2012. doi: 10.1109/MDM.2012.43.

[310] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer. Energy-Efficient Continuous Activity Recognition on Mobile Phones: An Activity-Adaptive Approach. In *2012 16th International Symposium on Wearable Computers*, pages 17–24. IEEE, June 2012. ISBN 978-0-7695-4697-1. doi: 10.1109/ISWC.2012.23.

[311] Z. Yang, C. Wu, and Y. Liu. Locating in fingerprint space: Wireless Indoor Localization with Little Human Intervention. In *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, page 269, New York, New York, USA, Aug. 2012. ACM Press. ISBN 9781450311595. doi: 10.1145/2348543.2348578.

[312] Z. Yang, L. Shangguan, W. Gu, Z. Zhou, C. Wu, and Y. Liu. Sherlock: Micro-Environment Sensing for Smartphones. *IEEE Transactions on Parallel and Distributed Systems*, 25(12): 3295–3305, Dec. 2014. ISSN 1045-9219. doi: 10.1109/TPDS.2013.2297309.

[313] K. Yatani and K. N. Truong. BodyScope: A Wearable Acoustic Sensor for Activity Recognition. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, page 341, New York, New York, USA, Sept. 2012. ACM Press. ISBN 9781450312240. doi: 10.1145/2370216.2370269.

[314] F. Ye, R. Ganti, R. Dimaghani, K. Grueneberg, and S. Calo. MECA: Mobile Edge Capture and Analysis Middleware for Social Sensing Applications. In *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, page 699, New York, New York, USA, Apr. 2012. ACM Press. ISBN 9781450312301. doi: 10.1145/2187980.2188184.

[315] M. Youssef and A. Agrawala. The Horus WLAN location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys '05*, page 205, New York, New York, USA, June 2005. ACM Press. ISBN 1931971315. doi: 10.1145/1067170.1067193.

[316] F. Yu, M. Jiang, J. Liang, X. Qin, M. Hu, T. Peng, and X. Hu. Expansion RSS-based Indoor Localization Using 5G WiFi Signal. In *2014 International Conference on Computational Intelligence and Communication Networks*, pages 510–514. IEEE, Nov. 2014. ISBN 978-1-4799-6929-6. doi: 10.1109/CICN.2014.117.

[317] W. Zhang, X. Meng, Q. Lu, Y. Rao, and J. Zhou. A Hybrid Emotion Recognition on Android Smart Phones. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 1313–1318. IEEE, Aug. 2013. ISBN 978-0-7695-5046-6. doi: 10.1109/GreenCom-iThings-CPSCom.2013.228.

[318] X. Zhao, Z. Xiao, A. Markham, N. Trigoni, and Y. Ren. Does BTLE measure up against WiFi? A comparison of indoor location performance. In *Proceedings of the 20th European Wireless Conference*, pages 263–268. VDE VERLAG GMBH, 2014. ISBN 978-3-8007-3621-8.

[319] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao. Travi-Navi. In *Proceedings of the 20th annual international conference on Mobile computing and networking - MobiCom '14*, pages 471–482, New York, New York, USA, Sept. 2014. ACM Press. ISBN 9781450327831. doi: 10.1145/2639108.2639124.

[320] L. Zhong, M. Sinclair, and R. Bittner. A Phone-Centered Body Sensor Network Platform: Cost, Energy Efficiency & User Interface. In *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, pages 179–182. IEEE, 2006. ISBN 0-7695-2547-4. doi: 10.1109/BSN.2006.4.

[321] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen. IODetector: A Generic Service for Indoor Outdoor Detection. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems - SenSys '12*, page 113, New York, New York, USA, Nov. 2012. ACM Press. ISBN 9781450311694. doi: 10.1145/2426656.2426668.

[322] 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 '10*, page 315, New York, New York, USA, June 2010. ACM Press. ISBN 9781605589855. doi: 10.1145/1814433.1814464.

Julien Eberle

Education

2011-2015 **PhD**, *EPFL-LSIR*, Lausanne.
1-year as research intern at Nokia Research Center, Lausanne

2009-2010 **Master in Computer Science**, *EPFL*, Lausanne.

2005-2010 **Computer Science studies**, *EPFL*, Lausanne.
1-year exchange at RWTH Aachen University, Germany

2001-2005 **Maturité Fédérale**, *Collège St-Michel*, Fribourg.
option Maths/Physique, Chimie

Experience

Internships

2010–2012 **Research Intern**, *Nokia Research Center*, Lausanne.
Master Thesis : Balancing power consumption and location precision for the "Universal Location Awarness". The prototype developed in python was able to save up to 20% energy compared to previous ones, with a continuous positioning. Internship: Design and implementation of an algorithm for identifying relevant places from the data collected on smartphones using this algorithm during a field test.

2008 **Semester Project**, *Fraunhofer Institute for Applied Information Technology*, Sankt-Augustin, Germany.
Designing, implementing and evaluating several methods to improve the accuracy of mobile phone's GPS for pedestrian positioning.

September 2007 **Internship**, *CERN*, Geneva.
Coding the acquisition module for SMAC (Smart Multimedia Archive for Conferences) using the DirectShow filters from MS DirectX, in C++.

Projects

2014– **Global Sensor Networks (GSN)**, *LSIR-EPFL*.
Managing the open-source middleware for distributed stream processing.

2009 **Design of an exertion game**, *Media Computing Group*, RWTH Aachen.
Imagining and prototyping a webcam controller for the Tetris game. The gesture recognition was done using OpenCV in a Objective-C project running on MacOS.

2007–2008 **Software Engeneering**, *EPFL*.
One year group project for the software engineering course, in collaboration with the "Ecole d'ingénieur et d'architecture de Fribourg". The event manger was running on J2EE server and a XML database, communicating with mobile devices running the compact dotNET framework.

Granges-des-Bois 5 – 1482 Cugy FR
📱 *+41 79 728 54 92* • ☎ *+41 21 693 67 87* • ✉ *julien.eberle@epfl.ch*
🖱 *http://people.epfl.ch/julien.eberle*

Extra-curricular Activities

2009–2010 **IT Manager**, *AGEPoly*, EPFL, Lausanne.
Managing the servers and workstations of the EPFL student union and member of the executive committee. Leader of the "myJob.epfl.ch" project centralizing all student job offers for the whole EPFL and external companies, in production since september 2010.

2009–2014 **Member of the organisation committee**, *PolyLAN*, EPFL, Lausanne.
President 2009–2010, Head of Sponsoring 2011–2014; Organizing the greatest LAN-Party in Switzerland, twice a year at the EPFL for 1'000 gamers.

2006–2008 **Teaching Assistant**, EPFL, Lausanne.
for "Programmation Avancée" and "Programmation I et II" courses, resp. with Prof. M. Odersky and Prof. J. Sam

Languages

French	**Native language**	
English	**Fluent (C1)**	
German	**Fluent (B2-C1)**	*1 year exchange at the RWTH Aachen University, Germany*

Personal Information

29, single, Swiss citizenship. Discharged from military service

hobbies Ski, Playing clarinet, Photography

Publications

Julien Eberle, Tri Kurniawan Wijaya, and Karl Aberer. Online unsupervised state recognition in sensor data. In *PerCom*, 2015.

Julien Eberle, Zhixian Yan, and Karl Aberer. Energy-efficient opportunistic collaborative sensing. In *IEEE MASS*, 2013.

Yongsung Kim, Julien Eberle, Riikka Hanninen, Erol Can Un, and Karl Aberer. Mobile observatory: an exploratory study of mobile air quality monitoring application. In *UbiComp '13 Adjunct*, 2013.

Erol Can Un, Julien Eberle, Yongsung Kim, and Karl Aberer. A model-based back-end for air quality data management. In *UbiComp '13 Adjunct*, 2013.

Bratislav Predic, Zhixian Yan, Julien Eberle, Dragan Stojanovic, and Karl Aberer. ExposureSense: Integrating Daily Activities with Air Quality using Mobile Participatory Sensing. In *IEEE Percom*, 2013.

Tri Kurniawan Wijaya, Julien Eberle, and Karl Aberer. Symbolic representation of smart meter data. In *EDBT '13 Workshops*, 2013.

Zhixian Yan, Julien Eberle, and Karl Aberer. OptiMoS: Optimal Sensing for Mobile Sensors. In *MDM '12*, 2012.

Olivier Dousse, Julien Eberle, and Matthias Mertens. Place learning via direct WiFi fingerprint clustering. In *MDM '12 - Industry track*, 2012.

J. Eberle and G.P. Perrucci. Energy measurements campaign for positioning methods on state-of-the-art smartphones. In *IEEE CCNC*, 2011.