SLS: Smart Localization Service

Human Mobility Models and Machine Learning Enhancements for Mobile Phone's Localization

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Michela Papandrea

under the supervision of
Marc Langheinrich and Silvia Giordano

March 2015

Dissertation Committee

Andrew T. Campbell	Dartmouth College, Hanover (NH), USA
Archan Misra	Singapore Management University, Singapore
Mehdi Jazayeri	University of Lugano, Switzerland
Laura Pozzi	University of Lugano, Lugano, Switzerland

Dissertation accepted on 17 March 2015

Research Advisor

Marc Langheinrich

Co-Advisor

Silvia Giordano

PhD Program Directors

Igor Pivkin and Stefan Wolf

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Michela Papandrea
Lugano, 17 March 2015

To Manuel

Abstract

In recent years we are witnessing a noticeable increment in the usage of new generation smartphones, as well as the growth of mobile application development. Today, there is an app for almost everything we need. We are surrounded by a huge number of proactive applications, which automatically provide relevant information and services when and where we need them. This switch from the previous generation of passive applications to the new one of proactive applications has been enabled by the exploitation of context information. One of the most important and most widely used pieces of context information is *location data*. For this reason, new generation devices include a *localization engine* that exploits various embedded technologies (e.g., GPS, WiFi, GSM) to retrieve location information. Consequently, the key issue in localization is now the efficient use of the mobile localization engine, where *efficient* means lightweight on device resource consumption, responsive, accurate and safe in terms of privacy. In fact, since the device resources are limited, all the services running on it have to manage their trade-off between consumption and reliability to prevent a premature depletion of the phone's battery. In turn, localization is one of the most demanding services in terms of resource consumption.

In this dissertation I present an efficient localization solution that includes, in addition to the standard location tracking techniques, the support of other technologies already available on smartphones (e.g., embedded sensors), as well as the integration of both Human Mobility Modelling (HMM) and Machine Learning (ML) techniques. The main goal of the proposed solution is the provision of a continuous tracking service while achieving a sizeable reduction of the energy impact of the localization with respect to standard solutions, as well as the preservation of user privacy by avoiding the use of a back-end server. This results in a Smart Localization Service (SLS), which outperforms current solutions implemented on smartphones in terms of energy consumption (and, therefore, mobile device lifetime), availability of location information, and network traffic volume.

Acknowledgements

Firstly I would like to express my gratitude to my advisors Silvia Giordano and Marc Langheinrich. I would like to thank Silvia for providing me the opportunity to start my PhD and for introducing me into the research area of mobile sensing and computing. She always supported and motivated me and she gave me important hints for advancing during my research. I would like to thank Marc for the interesting and fruitful discussions we had, and for his constructive comments to my work, which helped me to keep focus on my research, when I was starting to diverge.

I would like to thank also my former research advisor Mariagiovanna Sami and my former academic advisor Matthias Hauswirth, which have been for me an important guidance at the beginning of my PhD.

A special thank goes to all the members of the dissertation committee. And in particular I would like to thank Andrew Campbell for giving me the opportunity to join his lab and to work with his group: my visiting period at Dartmouth and the meaningful discussions I had with Andrew helped me to find my path in research and in my professional life. And I would also like to thank Archan Misra for the helpful and interesting discussions I had with him.

A big hug goes to Alessandro Puiatti and Anna Förster for their support during my PhD, and more than everything for their friendship. A very big thank goes to Emiliano Miluzzo and Nicholas Lane for the time they dedicated to me and to the work done together. I would also like to thank Matteo Zignani, Sabrina Gaito e Gian Paolo Rossi for the joint work on human mobility, which is part of this thesis.

Further I would like to thank my colleagues at SUPSI for their support and, above all, for their friendship. A special thanks goes to Daniele Puccinelli, Alan Ferrari, Andrea Bernaschina, Kamini Garg, Steven Mudda, Roberto Guidi, Andrea Baldassari and to the head of the Institute for Information Systems and Networking at SUPSI, Roberto Mastropietro. A particular thanks goes to all the colleagues and friends which helped me with the numerous data collection campaigns I organized during my PhD.

Last but not least, I am very grateful to my parents Melina and Giorgio and to my brother Raffaele, for always supporting and encouraging me during my life as PhD student and mother. A special thank goes to my little son Manuel, which unknowingly gave me the strength, day after day, to continue and to always give the best of me, in my work and in my life.

Contents

Contents **ix**

1 Introduction **1**
1.1 Problem Statement 2
1.2 Motivation: the new generation of Location-Based Applications . 3
1.3 State of the Art 6
1.4 Contributions 10
1.5 The Goal: Solving the Efficient Localization Problem 11
1.6 Organization of the thesis 12

2 System Overview **15**
2.1 Introduction 15
2.2 System Overview: the SLS architecture and approach 17
2.2.1 Learning and Prediction Modules 20
2.2.2 Inference Module 22
2.3 Dataset 23
2.3.1 Trajectories Dataset 24
2.3.2 Continuous Dataset 25
2.4 Conclusion 26

3 Analysis of Battery Consumption **27**
3.1 Introduction 27
3.2 Differentiation of localization procedures 28
3.3 Battery consumption measurement 31
3.4 Comparison of the battery consumption for different tasks 35
3.5 Conclusions 37

4 Activity Inference **39**
4.1 Introduction 39
4.2 State of the Art 40

4.2.1 Context Recognition 40
4.2.2 Activity Inference 42
4.3 Experimental data collection 47
4.3.1 First data collection campaign: feasibility study 47
4.3.2 Final data collection campaign 47
4.4 Raw data collection 48
4.4.1 Sensors 48
4.4.2 Sampling rate 50
4.5 Inference Window Selection 52
4.6 Activity Classification 55
4.6.1 Activities recognized 55
4.6.2 Features Extraction 55
4.6.3 Features Selection 60
4.6.4 On-Mobile features calculation 65
4.6.5 Classification Algorithm 66
4.7 Experiments and Evaluation 66
4.8 Conclusion 69

5 Learning User Mobility **71**
5.1 Introduction 71
5.2 State of the Art 72
5.3 Point of Interest 73
5.3.1 PoIs Identification 74
5.4 User Mobility Learning Algorithm 76
5.4.1 Location based Learning Algorithm 76
5.4.2 Location and Time based Learning Algorithm 79
5.5 PoI Classification 81
5.5.1 Relevance 82
5.5.2 Finding classes of relevance 83
5.5.3 Data pre-processing 84
5.5.4 Experiments and Results 88
5.6 Key Features in Human Mobility 90
5.6.1 Feature 1: Relevance 92
5.6.2 Feature 2: Geographical Distance 93
5.6.3 Feature 3: Transfer Time 93
5.6.4 Time Transfer and Geographical Distance Correlation . . . 94
5.6.5 Transition rules 97
5.7 Conclusion 98
5.8 Remarks 99

6 Mobility Prediction **101**
6.1 Introduction . . . 101
6.2 Location based model . . . 104
6.2.1 Experiment: next movement prediction . . . 105
6.3 Location and Time based model . . . 113
6.3.1 Experiments and Results . . . 114
6.3.2 Discussion . . . 120
6.4 Conclusions . . . 122

7 SLS Validation **123**
7.1 Presentation of the SLS . . . 123
7.1.1 Learning the user's mobility . . . 127
7.1.2 Location and Time based Prediction . . . 130
7.1.3 SLS's working scenario . . . 135
7.2 Validation . . . 135
7.2.1 Availability of Location Information . . . 138
7.2.2 Mobile device life-time . . . 140
7.2.3 Amount of network traffic exchanged . . . 142
7.2.4 Amount of predicted locations . . . 143
7.2.5 Visit and Activity Inference . . . 147
7.2.6 PoIs identification . . . 151
7.3 Comparative Study . . . 155
7.4 Conclusion . . . 157

8 Conclusion and outlook **159**
8.1 Summary and conclusions . . . 159
8.1.1 Human Mobility . . . 159
8.1.2 Mobility Prediction . . . 160
8.1.3 Activity Inference . . . 160
8.2 Directions for future research . . . 161
8.3 Peer-reviewed Articles . . . 163
8.4 Other Relevant Publications . . . 164
8.4.1 Short Papers . . . 164
8.4.2 Technical Reports . . . 164
8.4.3 Demo . . . 164

Publication List **164**

A On-Mobile Feature Calculation **165**

Bibliography **169**

Chapter 1

Introduction

The high potential of smartphones in terms of resources (e.g., battery, memory, computational power) and availability of embedded sensors and radio interfaces, has opened new perspectives and fueled the creativity of developers, who came out with an extremely high number of applications. This dissertation focuses on location-based applications, a class of applications of paramount importance that leverages location data either directly or indirectly. Examples of these applications range from navigation (e.g., Google Maps) to social (e.g., Facebook) applications, and from find-nearby-friends/places/offers to e-Health applications. The main goal of this dissertation is to investigate a smart and efficient localization solution for mobile phones that outperforms the state of the art, improving the trade-off between availability and reliability of the localization service while reducing resource consumption (mainly battery usage[1]). The novel contribution of the proposed solution consists of combining human mobility models and signals from embedded sensors (i.e., accelerometer) to minimize the need to invoke the localization manager on smartphones, radically reducing the cost of acquiring location information continuously.

This thesis presents the Smart Localization Service (SLS): an efficient localization strategy for mobile phones. The SLS offers a novel approach to the localization problem, by combining the smartness of the mobile devices with both Human Mobility Modelling (HMM) and Machine Learning (ML) techniques. The SLS approach is based on personalized modelling and optimal location reading.

1. *Personalized Modelling*: location predictions are performed by modelling the movements of each specific user among her/his most relevant visited locations.

[1]Other resource consumption are taken under consideration for the optimization of the trade-off with the reliability and accuracy of the service.

2. *Optimal Location Reading*: the SLS performs an activity analysis to understand whether a location reading is actually required.

The SLS is self adaptive, i.e., it dynamically adapts to the mobility behavior of the carrying user and the visited environment.

1.1 Problem Statement

In this dissertation, three main problems related to the localization for mobile devices, will be addressed: accuracy, resource consumption and privacy issues.
Localization accuracy and availability. The possibility for mobile applications to integrate location information raises indeed both technological and privacy issues. Location accuracy is one of the most important problems and also the most thoroughly investigated in the localization area of research. Each localization methodology has a target environment (e.g., generalizing: GPS for outdoor, WiFi for indoor), but mobile devices incorporate a variety of technologies, giving the possibility to deal with many environments and, at the same time, relying on a unique device. As I aim to demonstrate within this work, the accuracy of those different localization methodologies can be enhanced by merging the technologies themselves, and by involving other mechanisms in the localization process. The usage of embedded sensors, the collection and handling of sensed data and the introduction of a mobility model can enhance the trade-off between localization accuracy and battery consumption.
Resource consumption. Many restrictions (e.g., memory, battery, processing power) are imposed by mobile device's resources. While new generation smartphones are powerful enough to perform heavy operations in terms of memory and processing power, battery consumption is still a concern. Most operating systems do not deal with battery usage optimization, but provide APIs and allow designers to smartly manage it (i.e., simply using Google Maps App on an Android phone, implies continued requests for GPS location every one second, even if not moving; an opportune usage of the embedded sensors would allow the reduction of GPS requests, then the reduction of battery consumption). Therefore the smart usage of mobile capabilities for reducing the battery consumption is still an open problem. In this thesis I will present a strategy (applied to Android mobile devices, but easily generalizable to other platforms) to face this issue.
Privacy. The knowledge of context and personal information related to mobile users allows a range of relevant services, but this mechanism can violate their privacy. Often users are asked to choose whether to share their own location information, or not. In the case of acceptance a given level of location privacy

is ensured; otherwise, the user is not allowed to access the desired service. One idea to preserve privacy and, at the same time, to be able to access all the desired services, is for the user to decide the granularity of the shared locations, having the possibility to choose when, where and how the location should be shared. It is easy to understand that "*shared data cannot be secure*", but this implies that the device is independent from any back-end server, for all the required heavy computational tasks. In this dissertation, this issue will be addressed, and algorithms will be provided to make the mobile device autonomous.

This dissertation addresses and solves the problem of:

providing location information continuously and accurately while ensuring efficient resource usage and without violating the privacy of the user.

1.2 Motivation: the new generation of Location-Based Applications

Location information can be considered the basis for many mobile applications, which provide new services and facilities (B'far [2004]). Initially, location-based applications (which provide services based on the user's geographical location) were "show me nearby restaurant"-type applications, they were reactive and mostly client-server focused; users would have asked an application or a system for information and received a response. The next generation (Shek [2010]), that is our present and future generation, is more proactive and interactive with the user, thanks to new push notification mechanisms (Cremonese et al. [2010]). For example, users can receive relevant information based on their location, without the need to manually search for it. This new generation of location-based applications provides many benefits for users and service providers (i.e., GoogleNow).

- The vast amount of data available on the Internet is filtered into relevant information based on the user's current context.
- Only relevant information is shown to the users, speeding up decisions and activities and highlighting information that users may not normally be aware of (i.e., temporary road closure for traffic).
- The amount of data entry required by the user to access a service is reduced, given the integration of embedded sensors (i.e., accelerometer, digital compass and cameras) data within applications.
- The sharing of location tagged information (i.e., photos and reviews) allows up-to-date localized information to be available to many users.

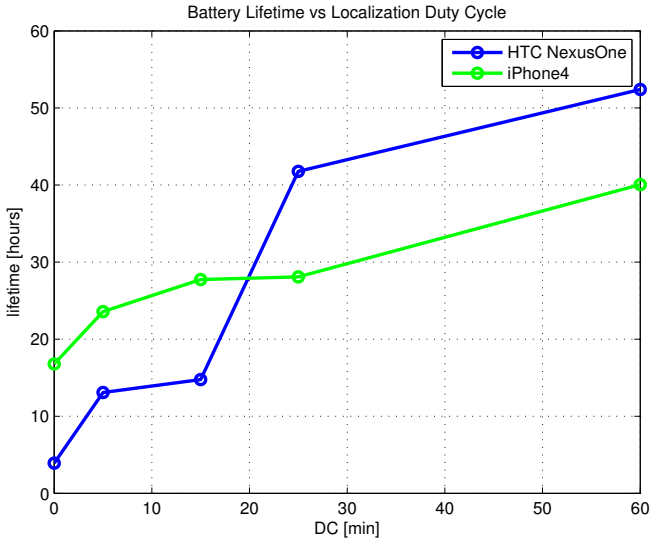


Figure 1.1. Battery Life vs Localization's Duty Cycle

- Service providers can build models for improving services provided to users in real-time and over longer term, by collecting and elaborating data from their movements (i.e., location traces) and combining them with associated tagged information.

However, an "efficient localization" is critical for many applications, and the existing solutions are far from being smart and computationally efficient: they drain phone's resources, mainly work outdoors (and difficultly in some city's scenario) and hardly give a really precise location of the user.

In this thesis I focus mainly on the battery resource of mobile devices. Since new smartphones are provided with powerful processors and quantity of RAM which makes them able to perform heavy computations, I exploit this innovative technology to smartly reduce the usage of more expensive hardware components (e.g., GPS, WiFi).

Figure 1.1 shows the average battery lifetime of a smartphone with respect to the time interval between consecutive location tracking (*duty cycle*: DC), while running a location based application (the VibN[2] application described in Miluzzo et al. [2011]). I measured the variation of the battery life for different values of the DC, by using two devices: the HTC NexusOne running Android 2.2, and

[2]I worked for the design and development of this application during my visiting period at the Dartmouth College (USA), 2010

the iPhone4. VibN is a localization service able to profile the social behavior of people, allowing users to record audio snippets, geo-tag them, and display them on a map as a way to provide social hotspots insights application. It is composed by a client and a server side. The VibN client application running on the mobile device, performs queries to the OS Localization Manager at fixed interval of time (with fixed DC) and retrieves information about the current location. Without duty cycling (continuous localization), the battery lifetime of the phone is quite short: even if for the iPhone it is longer, it does not reach the daily lifespan. Increasing the duration of the DC, obviously also the lifetime increases: the increment is logarithmic but the behavior depends on the device and on the operating system. Being interested in providing a continuous localization service we can concentrate our reasoning on the first portion of the graph, when the DC ranges between [0-5] minutes. For both platforms, the gain in lifetime is between [7-10] hours, which means that performing a location reading every fixed interval of 5 minutes increases the phone lifetime of almost 10 hours.

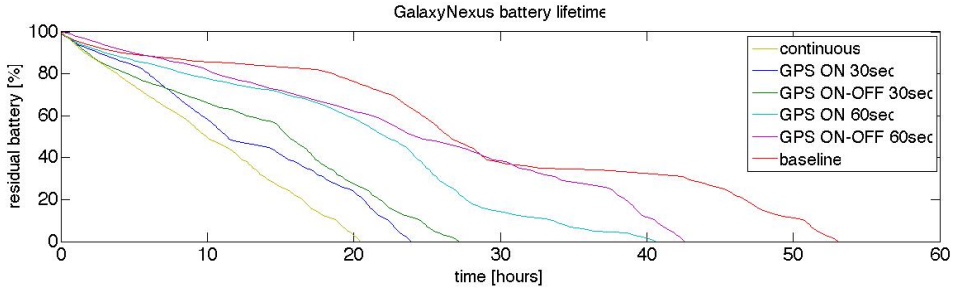


Figure 1.2. Battery lifetime for a smartphone Samsung Galaxy Nexus, Android 4.1.1

The dependency between the mobile device lifetime to the localization duty-cycle is not strictly related to the application mentioned above (VibN), but it can be generalized to all the location-based applications and services. This is visible in figure 1.2, which represents the battery discharge of a Galaxy Nexus device running Android 4.1.1, while only running a continuous localization service at different duty cycles (the figure is described more in detail in section 3.3). Also in this case we can notice the great difference in battery lifetime duration while performing localization at different frequencies.

This motivates the approach presented in this dissertation which instead of using a fixed duty cycling, adapts the localization procedure to the actual mobility behavior of the user. Additionally it complements the reduced frequency of the location readings - requiring then a reduced amount of resources - with more

lightweight localization-complementary techniques (e.g., movement prediction and activity inference).

1.3 State of the Art

Recently, many methods for the efficient *resources usage* in location-based applications have been suggested. Gaonkar et al. [2008] presented Micro-Blog, a participatory sensing application which addresses the challenge of balancing the competing goals of accurate location coordinates and long battery life. Since accurate localization cannot come at the cost of unacceptably short battery lifetime, Micro-Blog strategy consists in unfrequently using more accurate, but power-hungry localization services (i.e., WiFi). The frequent location readings are performed by using less accurate and therefore more power-efficient services (i.e., GSM localization). However this approach, while reducing the energy consumption due to the localization, decreases also the location information accuracy.

Abdesslem et al. [2009] proposed SenseLess, which also leverages the different energy consumption characteristics of sensors embedded into mobile phones, to maximise battery life in mobile-sensing applications. It uses the less expensive sensors more often, thereby enabling the usage of more expensive sensors less frequently, saving more than 58% of energy when determining a user's location, while maintaining the fidelity of the sensed data with respect to a power-hungry GPS-based system, for a typical indoor and outdoor walk. This approach has a great impact on the battery consumption, however it strictly depends on the mobility behavior of the user: for instance, if the user is always moving and visiting different locations, the amount of energy saved decreases significantly. As for the previously presented work, also in this case a user mobility learning procedure could be helpful.

Kjærgaard et al. [2009] designed a system called EnTracked which profiles how mobile devices consume power, including positioning and communication. Then, a model is proposed to estimate and predict the system conditions and mobility of a device. In the EnTracked, the accelerometer is utilized to detect pedestrian movement in order to activate the GPS module and the UMTS module (messages sent to a back-end server) properly. Constandache et al. [2009] proposed EnLoc, a solution dealing with localization technologies cost: it is an energy-efficient localization framework developed to face the unacceptable energy cost of GPS, used by many mobile phone applications. The framework characterizes the optimal localization accuracy for a given energy budget, and

develops prediction based heuristics for real-time use. Both *EnTracked* and *EnLoc* provide a smart user adaptive solution for the energy used by the embedded sensors while providing a localization service, however their performances are strictly related to the context. Also for these solutions, a user mobility learning procedure could improve the energy saving performances.

Kim et al. [2010] introduced SenseLoc, a solution working with an approach very similar to the one implemented by the SLS, providing everyday contextual information abstracting locations as place visits and path travels. Based on user's mobility, SensLoc proactively controls active cycles of a GPS receiver, a WiFi scanner, and an accelerometer. However, this solution is not autonomous and it depends on back-end server for the computational offloading.

RAPS, presented by Paek et al. [2010], is a rate-adaptive positioning system for smartphone applications which uses a collection of techniques to cleverly determine when to turn on GPS. It estimates user movements using a duty-cycled accelerometer, and utilizes Bluetooth technology to reduce position uncertainty among neighboring devices. It also employs celltower-RSS blacklisting to detect GPS unavailability (i.e., indoors) and avoids turning on GPS in these cases. The authors demonstrated that the system can increase phone battery lifetime by more than a factor of 3.8 compared with GPS always on. However, also in this case, the battery consumption strictly depend on the mobility of the user.

LifeMap, presented by Chon and Cha [2011], is a smartphone-based context provider for smartphones, for indoor and outdoor environment. It provides an advanced location service for mobile users. It uses inertial sensors (accelerometer and digital compass) to perform indoor localization (implementing a step counter). The information is combined with GPS and Wi-Fi positioning systems, to generate user context in daily life. The presented system reduces the energy consumption by using a minimum set of sensors to define context in a given situation. However, the authors state the need for a technique to minimize the energy consumption of the solution, by adding to the system a human-centric location-prediction module.

Bareth and Kupper [2011] introduced a system which enhances the energy usage on mobile devices while performing localization: it dynamically deactivates different positioning technologies and only activates the positioning method with the least energy consumption (GPS, WiFi, GSM). In this way, the algorithm can reliably and accurately determine, if the user leaves or enters predefined geographic areas, while preserving valuable energy resources. However this solution offers an effective localization in terms of energy consumption, at the price of a decreased accuracy in the information retrieved.

Also Oshin et al. [2012] presented an algorithm to improve the energy-

efficiency of GPS based location sensing applications used by smartphones. They used the embedded smartphone accelerometer to differentiate between two main users activity: stationary and in-motion. In this way, they activate the expensive GPS only when needed, that is when the user is moving. Only applying this simple activity differentiation, they have shown that the battery consumption is decreased significantly. However the mobility context detection algorithm presented in this work could be further improved by considering a wider set of user activities.

SensTrack described by Zhang et al. [2013], is a location tracking service that leverages the sensor (acceleration and orientation sensor) hints on the smartphone to reduce the usage of GPS. Furthermore it switches to the alternate location sensing method based on WiFi when users move indoors. A machine learning technique is then employed to reconstruct the trajectory from the recorded location samples, by analyzing the collected data offline. The authors demonstrated that their approach reduces the usage of GPS and still achieve a high tracking accuracy. However this system works only for pedestrian movements and it is quite expensive in terms of energy consumption because it assume a continuous sampling of the acceleration and orientation sensors.

SmartLoc is a smart localization system, presented in Bo et al. [2013], which aims at improving the localization accuracy when the GPS signal is weak (i.e., metropolitan areas) while performing outdoor localization, and at the same time it reduces the energy consumption for localization by carefully turning on GPS periodically. To achieve its goal, SmatLoc leverages the lower-power inertial sensors embedded in smartphones, and in particular it uses the accelerometer, the magnetometer and the gyroscope together with the GPS, to estimate the location and the traveling distance, detecting automatically landmarks (e.g., bridge, traffic lights) and special driving patterns (e.g., turning, uphill, and downhill). However this solution is only designed for vehicles and is dependent from fixed landmarks, in order to calibrate the localization when the cumulative error given by the inertial navigation becomes very high.

Chon et al. [2014] present a system called SmartDC, which similarly to the SLS approach, learns the user mobility behavior with respect to his regular movements among his relevant PoIs. The authors implemented a Markov decision algorithm in order to predict the time duration of a visit to a PoI. In this way the SmartDC is able to adapt the location reading duty-cycling with respect to the predictions, minimizing at the same time the localization energy consumption. Although this method achieves very good results (it consumes 81% less energy than a periodic sensing scheme), it requires a very long time to learn the residence-time patterns (i.e., it requires three months to reach around $72 \pm 9\%$

predictability) and it does not work without WiFi coverage. Furthermore it has been validated offline, emulating the SmartDC system over a dataset of real traces collected by 57 users over four weeks (LifeMap dataset Chon and Cha [2011]).

In this dissertation I present the SLS system, which similarly to the related work presented above, aims at providing a continuous localization service, reducing the required amount of energy with respect to a standard localization approach. The novelty of the proposed solution with respect to the state of the art consists in: *(i)* shortening the bootstrap time to learn the user mobility regularities and to predict the next visited locations and departures time; *(ii)* being adaptive to the current user's activities (being able to infer the current activity over a set of four main classes), hence very fast in detecting the change of user's visited contexts; *(iii)* being implemented and deployed on real smartphones and *(iv)* being validated not only in an emulated environment (by using datasets of real traces) but also with real experiments.

In table 1.1 I propose a comparison of the SLS with the previously presented energy-efficient location-based solutions, in terms of: technologies used for the localization (e.g., GPS, WiFi, GSM. Bluetooth), sensors usage (e.g., accelerometers and other sensors), dependency on back-end servers (as processing units and/or databases), and ability to learn and make predictions based on the moving history, or the capability to learn and predict moving patterns by means of inertial navigation reasoning (i.e., *SmartLoc*). Similarly to the state of the art, also the SLS system aims to reduce the energy consumption of a continuous localization procedure, however without impacting the accuracy of the localization itself and the availability of the service: the system will not only reduce the location sampling frequency to reduce the battery consumption, but it will smartly decide when to use the localization technologies provided by the mobile operating system, according to the user's history (learned habits and behavior), and to the current context (self adaptiveness). A further innovative aspect of the SLS consists on the fact that the system completely runs into the mobile phone, without relying on a back-end server and without sharing information with other nodes: this characteristic helps in preserving the privacy of the user which does not share any sensitive data with any other nodes or server.

	GPS	WiFi	GSM	Bluetooth	Sensors	Back-end server	Location learning/ prediction
MicroBlog [2008]	•	•	•			•	
SenseLess [2009]	•	•			•		
EnTracked [2009]	•				•	•	
EnLoc [2009]	•	•	•			•	•
SenseLoc [2010]	•	•			•	•	
RAPS [2010]	•		•	•	•		
LiveMap [2011]	•	•	•	•	•		
Bareth and Kupper [2011] [2011]	•	•	•				
Oshin et al. [2012] [2012]	•				•		
SenseTrack [2013]	•				•	•	•
SmartLoc [2013]	•				•		•
SmartDC [2014]	•	•	•				•
SLS	•	•	•		•		•

Table 1.1. 2: Energy-Efficient Location-Based Solutions comparison

1.4 Contributions

The main focus of this thesis consists in reducing the resource consumption, mainly the battery consumption, due to the localization. However the battery consumption is not a simple function of the localization duty-cycle (as I already mentioned in section 1.2), but it depends on multiple aspects: on the platform (e.g., on the hardware of the mobile device, on the running OS, on its version, etc.) as well as on its usage (e.g., which applications are running on the mobile, and with which frequency the user is interacting with them, etc.) and on the activities of the carrying user. Therefore, the main contribution of this thesis consists in the study and implementation of a **smartly dynamic location duty-cycling** which adapts to the actual movements of the user. This is performed by SLS, implementing an activity inference algorithm which understands whether the user is moving and which is the activity performed. In this way, it performs the localization only when necessary, and adapt the tracking duty cycle to the inferred activity. Additionally, the SLS incrementally learns the user's mobility behavior and adapts the localization duty-cycles to the location prediction availability: the system performs location prediction instead of a direct location tracking whenever it is possible.

Thanks to its innovative approach, the SLS is expected to outperforms existing solutions for mobile phones, in terms of:

- impact of the service on the overall device's lifetime and on the computational load of the device's processor;
- service's response time;
- preservation of the user's privacy.

In terms of privacy, the advantage of moving the localization computational unit into the mobile device is straightforward (while traditionally it was residing on the server side).

1.5 The Goal: Solving the Efficient Localization Problem

The localization problem I aim to address with my work, is just beneath the application layer: it regards the acquiring of location information by relying on different technologies, which are not only the standard location tracking techniques used nowadays. Smartphones need battery, memory and computational power to perform usual operations. Hence, when adding localization to the set of services provided to the mobile phone's user, it is crucial to minimize the impact on the phone's life and overall performance.

The goal of this thesis is to study a localization solution which includes, in addition to the standard location tracking techniques, the support of other technologies, nowadays available on mobile devices; as well as Human Mobility Modelling (HMM) and Machine Learning (ML) techniques (details in chapters 4, 5 and 6), in order to provide a Smart Location Service (SLS) which is expected to outperform existing solutions for mobile phones (Papandrea and Giordano [2012], Papandrea and Giordano [2014], Papandrea [2012]).

To solve the problem stated in section 1.1, my goal is to find out a new way to provide a "broad" localization service exploiting the smartness of these technologically advanced devices.

In fact, a smart-phone is generally a communication platform, which has a powerful processor (e.g., ARMv8-A 1.4 GHz dual-core Cyclone CPU for the Apple's iPhone6 and an Nvidia Tegra K1 2.3 GHz dual-core Denver CPU for the HTC Google Nexus9), a great amount of memory (e.g., 2GB RAM for the Google Nexus9, 1 GB RAM for the iPhone6, 2GB RAM for the HTC One), extensible internal storage capacity (e.g., 16 or 32 GB flash memory storage for the

Nexus9, 16/64/128 GB flash memory for the iPhone6), sensors embedded (e.g., accelerometer, 3-axis gyroscope, digital compass, barometer, proximity sensor, ambient light sensor, Assisted GPS), different communication interfaces (e.g., Wi-Fi 802.11 a/b/g/n/ac, Wi-Fi Direct, 4G, Bluetooth Low Energy, Near Field Communication) and open operating system (e.g., Android, iOS, Symbian, LiMo, Openmoko, bada).

All these new generation device characteristics give the opportunity to approach the localization problem in a smarter way with respect to the past. We can now merge all the capabilities given by these new technologies to provide a "smarter localization service". The measure of how much smart this new localization service is, with respect to existing solutions, is evaluated on the basis of a list of metrics.

1. Impact of the service on the overall device's life and on the computational load of the device's processor.
2. Enhancement on the service's response time.
3. Enhancement in privacy's issues derived by moving the localization computational unit, that traditionally is on the server side, into the mobile device.

Therefore, the evaluation will be performed by means of a comparative study of the continuous localization service provided by the SLS, with two alternative solutions.

1. A very simple localization solution, including only well known technologies, and providing a continuous tracking;
2. A continuous localization service which provide a tracking procedure fairly comparable to the SLS, but using Google APIs for the user's Activity Recognition.

The comparative study will not be a merely comparison of the resources (mainly the energy) used to perform localization, but the evaluation will be performed considering the trade-off of the energy consumed given a number of requirements (e.g., accuracy for outdoor, exchanged network traffic, location availability). Each single module composing the system will be firstly evaluated independently, and subsequently the whole SLS will be validated by means of a real experiment.

1.6 Organization of the thesis

The rest of the thesis is structured as follows.

Chapter 2 presents the SLS system from the functional point of view. It deals with the SLS system, whose goal is to manage the trade-off between service accuracy and resource usage. The idea behind the proposed system, is to find a criterion which allows the device to receive a continuous localization service, without although having to query continuously the localization engine, hence reducing the resources consumption due to the service. This criterion is driven by two main logic: the first one is the location prediction performed by modeling the user's movements; the second one is the context and activity analysis whereby the device understands whether a query to the localization engine is actually required. In particular, chapter 2 introduces the system architecture, it deals with the analytical study of the SLS and introduces the three main modules which compose the system: the Activity Inference Module, the Learning Module and the Prediction Module.

Since the main target of this thesis is the *energy efficiency* of the localization on the mobile devices, chapter 3 is focused on the importance and the different methodologies to measure the battery consumption. I present in this chapter the state of the art in this area, and the two different methodologies adopted in this thesis, presenting the pros and cons of each approach. At the end of the chapter I further motivate the work performed for this thesis, by measuring the energy required by a mobile device while performing different tasks (e.g., localization by means of GPS, accelerometer sampling).

The next chapters analyze more in detail each single module of the SLS. Specifically, chapter 4 focuses on the Activity Inference module. It presents the state of the art on this research area and explains in detail the algorithm implemented by the SLS, comparing it with the related works, and evaluating it.

Chapter 5 focuses on the Learning Module implemented by the SLS. It presents the state of the art on human mobility modeling and explains in detail the algorithm implemented by this module. It analyzes the regularity of human mobility with respect to the locations visited and, in particular, it shows the existence of some patterns according to the relevance of the identified locations. In this chapter I explain the learning algorithm implemented by the SLS, which in the first phase was a graph based algorithm, learning information about consecutively visited locations. In a second phase, the algorithm included the timing feature, and associated the user's visits to relevant locations with their duration and probability distribution during the day. This chapter concludes with some results obtained by analyzing this single module.

Chapter 6 introduces the algorithm implemented by the *Prediction module*. It presents the prediction based on the learned location-based and location and time-based models. Specific results are shown at the end of the chapter, evalu-

ating the single module.

Chapter 7 presents the complete implementation of the SLS and how the three previously presented modules work together. The chapter presents also a validation section where I show the performances of the SLS against a standard continuous localization service. The validation is performed in terms of battery lifetime against location availability: I show the difference in lifetime for the phone running the SLS and the one running the continuous localization service, and compare it with the location information availability given by the two services. Furthermore, I compare the SLS against a modified version of the system, whose Inference Module uses the Google Activity Recognition APIs. These two systems have been compared in terms of mobile phone lifetime, amount of network traffic exchanged, accuracy of the inference. Moreover, I quantify the impact of the prediction on the localization, and I calculate the accuracy on the identification of the PoIs, against the ground truth.

Finally, chapter 8 summarizes the work described in this thesis, it provides an overview of the main findings and results, and it describes the future directions derived from this work.

Chapter 2

System Overview

In this chapter I present the design of the Smart Localization System: and in particular, the architecture of the SLS and its approach to the localization problem presented in the previous chapter (section 2.2). Each module composing the SLS system will be presented separately, together with the logic which is behind them (sections 2.2.1 and 2.2.2). Each module will then be explained in more details and evaluated in the consecutive chapters. At the end of this chapter (section 2.3) I present two datasets which have been used to evaluate the SLS modules, and in particular a trajectories and a continuous location dataset. The first one contains data basically collected by people moving among different relevant locations. The second one instead, contains the location data continuously collected by some users during their daily routine, hence it includes both locations data collected while moving and while visiting relevant locations.

2.1 Introduction

To clarify the main points already mentioned in the first chapter, I briefly summarize here the target *working environment* of the proposed localization service, the *problems* that are addressed, the *methodologies* used and the *improvements* provided.

Working environment. Our system is meant to reside beneath the application layer (as depicted in figure 2.1, which represents the system from a functional point of view, and 2.2 which represents the system from a more technical point of view), and aims to offer it an optimal localization service, which is in charge of providing accurate information, as well as, of managing the mobile device resources. The system is thought for a new generation mobile phone, with a CPU clock rate on the order of GHz (i.e., 2 GHz or more, for the smartphones

nowadays available on the market) and memory capacity on the order of GB (i.e., 2GB or more). A set of embedded sensors are available on the platform, such that it can exploit the data sensed by an accelerometer, a GPS receiver, WiFi and GSM interfaces, etc. The target working environment for the system is a broad scenario, which can be thought as the aggregation of almost every *human's accessible places*. The ubiquity of the service and the target platform are the most crucial sources of challenges.

Addressed problems. The proposed solution address the issues of a continuous sensing system, which could be expensive in terms of battery consumption and CPU load, which needs to be responsive and accurate, and has critical security issues.

Methodologies used and improvements provided. The solution I provide to address these challenges is a combination of different technologies, which does not heavily impact on the overall computational load of the phone and whose portability is characterized by a large scope. The system uses well-known technologies nowadays adopted by already existing localization services, such as GPS, WiFi and GSM, (or more specifically, the Localization Manager provided by mobile Operating Systems: *OS-Localization Manager* in Figure 2.2) to retrieve location information. These technologies are combined with a human mobility model (*HMM Manager* in the figure), whose goal is to learn about the users movements, and to allow the system to make predictions (by means of the *Prediction Manager*). For each device, the model starts from scratch and is dynamically modified with time. By collecting data, the model incrementally learns about the user's habits in everyday movements, and adapts itself to the carrying user. This is done to avoid unnecessary frequent location measurements, hence worthless battery consumption. The initial phase of *incremental learning* may require a significant amount of time, depending on the user behavior. After this initial phase the system will be able to exploit the learned information in order to consume less energy. More improvements on the overall system performance can be added by the usage of sensed data, performing inferences and adjustments on the movements prediction: for example, the accelerometer readings can be used to infer user's activities; GPS can be used to identify relevant locations.

More details about the system architecture (Figure 2.2) will be explained in the following sections, and in the rest of the thesis.

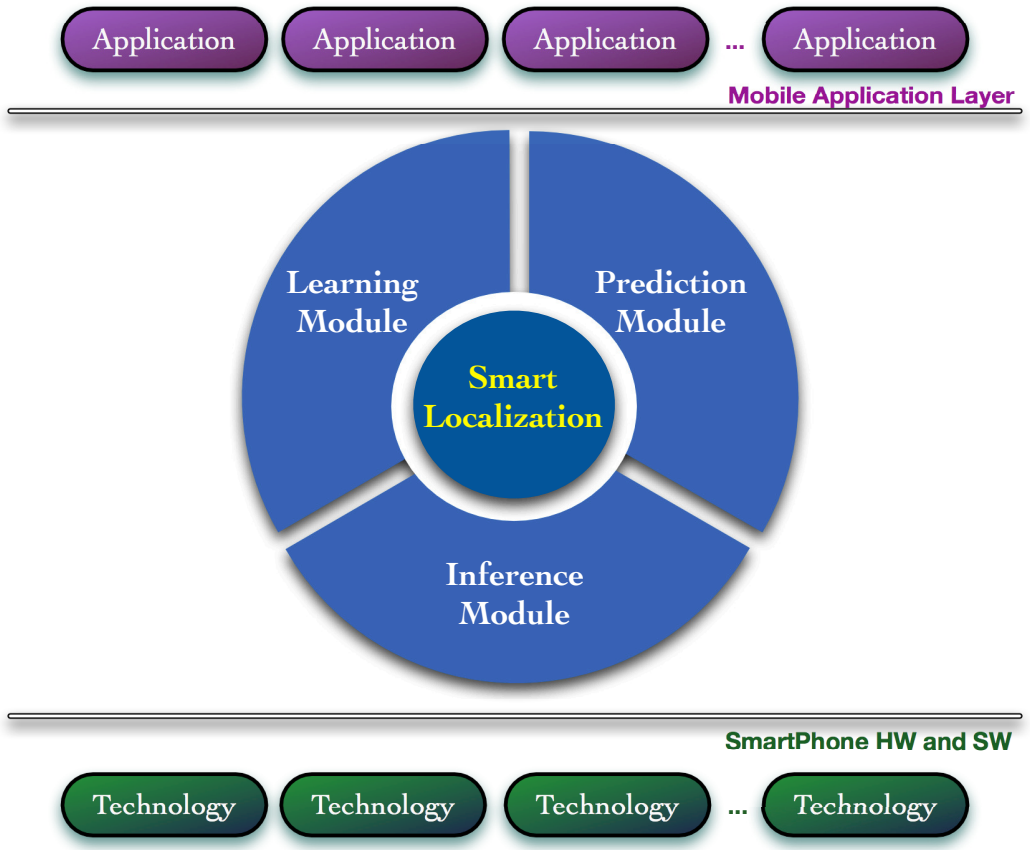


Figure 2.1. SLS Functional Structure

2.2 System Overview: the SLS architecture and approach

The SLS approach is driven by the two criteria presented in the previous chapter: *Personalized Modelling* and *Optimal Location Reading*. The system architecture (figure 2.2) is introduced by Papandrea and Giordano [2012], and includes a set of managers that collaboratively act to collect sensing and location information, learn about user's behavior and support localization with prediction[1]. The SLS exploits the human mobility to learn about personal habits and thus performs prediction based on past user's behavior. However, as the system performs all the activities without the need of external data analysis systems, i.e. there is no data exchange or external data query, it does not incur in privacy issues.

[1]The *tuned data* is transmitted from the Learning Manager to the HMM Manager at precise intervals, not in real time as it is for the communications between the other managers

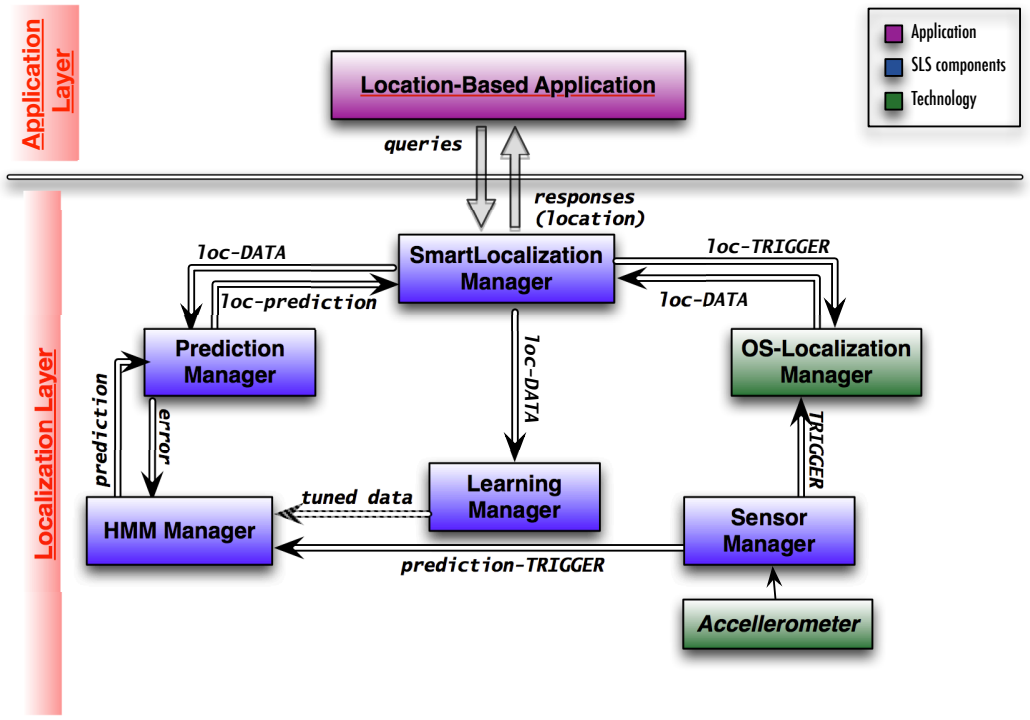


Figure 2.2. SLS Architecture

Figure 2.1 gives a functional view of the SLS system. It relies on the technologies (HW and SW built-in functionalities) provided by the mobile phone, represented in the figure by general *Technology* blocks:

- the OS Localization Manager which retrieves location information using well-known technologies, such as GPS, WiFi and GSM (*SmartPhone SW* in the figure).
- the HW technologies, such as the accelerometer (*SmartPhone HW* in the figure).

The SLS approach is centered on a reasoning element, the *Smart Localization* component, and supported by the following functional modules:

- the SLS Learning and Predicting Modules, which work together to implement the personalized modelling and movement predictions;
- the SLS Inference Module, which performs the optimized location reading;

The *Smart Localization component* (SLoc) plays a central role in the SLS approach, it is the "brain" of the system where all the decisions are taken in order

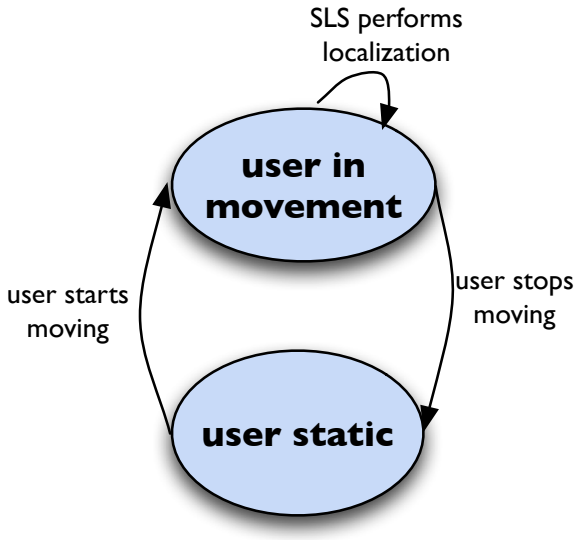


Figure 2.3. SLS State Machine

to optimize the localization: whether the prediction is satisfactory or there is still the need to learn about the user by collecting additional location information, or if it is necessary to query the OS Localization Manager, hence to keep active the sensing devices. Figure 2.3 shows the two main states which differentiate the behavior of the SLS: the inference module triggers the events which characterize the transition from one state to the other. The localization procedure is performed only when the system is in the "*user in movement*" state. The high level flowchart of the SLoc component reasoning, while updating the location information, is shown in Figure 2.4.

The smartness of SLS allows the resource (especially battery) consumption management. As we illustrated in Papandrea and Giordano [2012], when a real location based application [Miluzzo et al., 2011] uses the OS Localization Manager with continuous location tracking (no duty cycling), the battery lifetime of the phone is quite short. Even if in the future we will have the support of longer lifetime batteries, the tradeoff between battery consumption and optimal localization will remain an issue. To achieve optimal localization while reducing the battery consumption, the SLS approach fosters less frequent location readings complemented by less expensive localization techniques (i.e., movement prediction).

2.2.1 Learning and Prediction Modules

In figure 2.5 we show how the Learning and the Prediction Modules work together within the SLS: the components involved are the Prediction Manager, the Mobility Model Manager and the Learning Manager. Each manager supervises the realization of a specific task, and all together contribute to accomplish the *personalized location prediction* of the user. The Learning Manager (*LMng*) stores locally the localization data, performs some mining over it and calculates the parameters which will personalize the prediction model. LMng periodically applies a clustering algorithm[2] over the data and retrieves the set of points of interest (as explained in the next chapters) and related probabilities which are needed by the Mobility Modelling Manager (*MMMng*). While running, by collecting data, the LMng incrementally learns about the user's habits and contexts, and allows the MMMng to adapt the model to the learned information.

The MMMng applies the parameters computed by the LMng in the predic-

[2]In the final version of the SLS the PoIs are retrieved without the need of a clustering algorithms. More details can be found in chapter 7

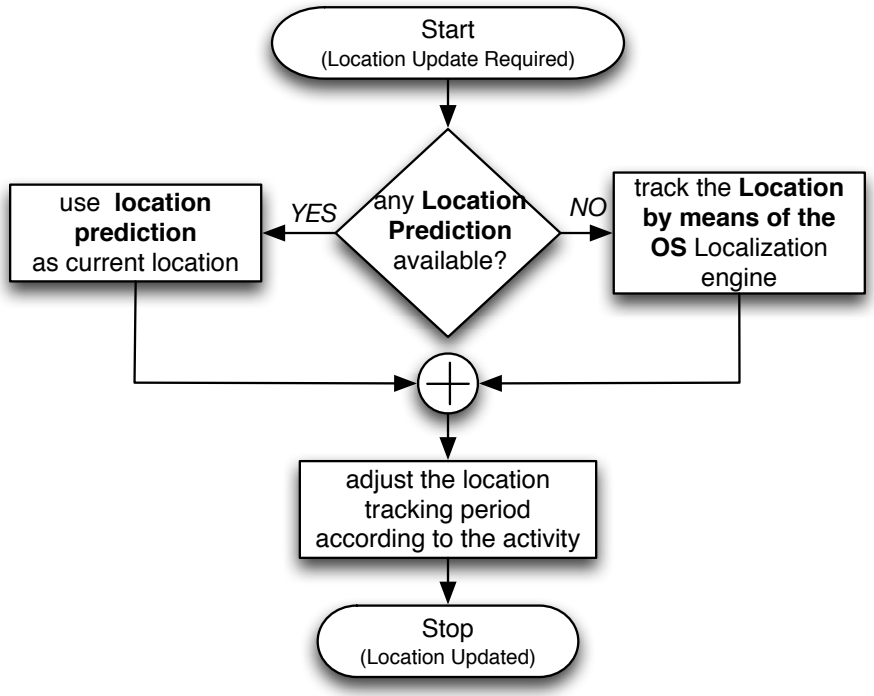


Figure 2.4. SLS Flowchart: when the user is moving, the SLS tries to predict the location. If this fails, the SLS uses the standard OS tracking techniques.

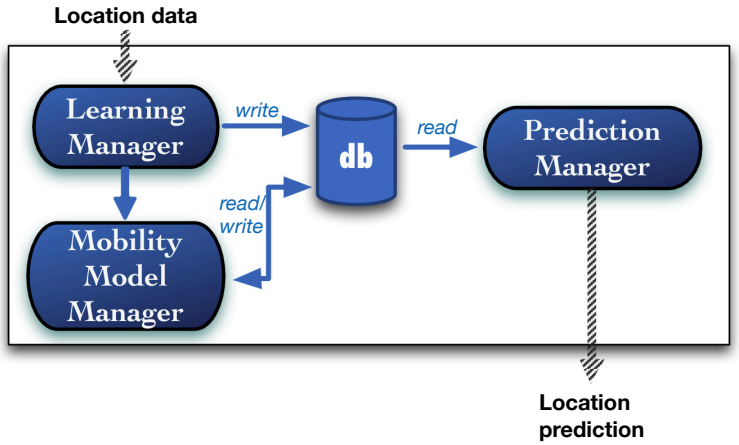


Figure 2.5. SLS Learning and Prediction Modules Architecture

tion mobility model; finally, the Prediction Manager (*PMng*) uses the model for predicting the next user movements (see chapter 6) and for calculating the *prediction error* with respect to the real locations, when the user is approaching it.

Before starting any localization procedure, the SLoc component interrogates the Prediction Module in order to check if there is any available prediction about the next movement. If available and reliable (more details about the prediction reliability are explained in chapter 7), it just takes the prediction as current position. If no prediction is available, the SLoc triggers the Operating System Localization manager (*OSLMng*).

For each device, the SLS starts the Mobility Model from scratch and dynamically modifies its parameters with time to personalize the predictions. Clearly, during the *Start Up* phase (from the first run of the system, until the model has been generated), the localization is performed only using the node's tracking technologies (OSLMng), as the SLS still need to personalize the mobility model to the carrying user.

When the location prediction error value goes below a given threshold, the bootstrap phase of the prediction model is considered completed. At this point, the localization procedure starts using the Prediction Module for the localization, and decreases the triggering frequency of the OSLMng. This personalization of the model further allows (in addition to "*the user is moving?*" check) to avoid unnecessary frequent location measurements, as several habits and contexts are known, hence it prevents worthless battery usage. In terms of battery consumption, initially the system does not gain anything with respect to other techniques in literature. The real advantage is visible at the end of the bootstrap phase,

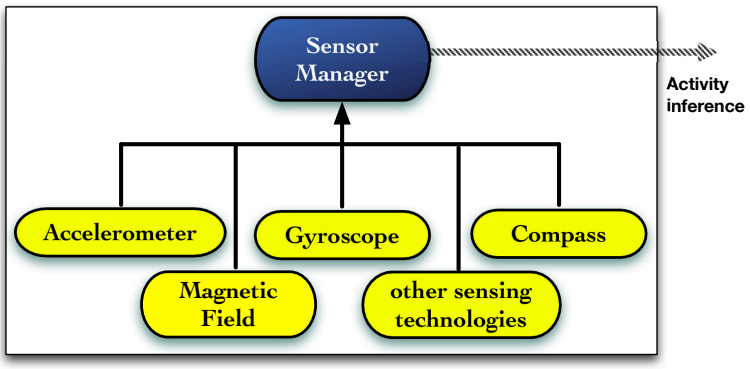


Figure 2.6. SLS Inference Module Architecture

after which the node starts decreasing the frequency of direct location tracking, alternating them with predictions.

2.2.2 Inference Module

The Inference Module, as illustrated in Figure 2.6, is composed by the Sensor Manager which interacts directly with some mobile phone HW components.

The Sensor Manager (Smng), based on some mining on the data collected by the sensors, infers the activity of the user and sends back this information to the SLoc component which decides its actions based on the activity of the user:

1. if the user is static, the SLoc component returns the last tracked location (the user did not move since the last location reading);

2. if the user is moving, the SLoc component changes the tracking frequency (interval of time between consecutive location readings) according to the actual activity of the user.

The purpose of this inference is to reduce the resources usage, by adding flexibility in the localization procedure. While the user is static, there is no need to perform location readings, because the location does not change until the next movement. While the user is moving, according to the kind of movement the localization procedure may be different, in order to retrieve location data with similar accuracy: in particular, if the user is moving slowly (i.e., walking), a less frequent location reading allows the system to track all the movements of the user, while if the user is moving faster (i.e., travelling on a vehicle) it is necessary to perform more frequent location reading.

Differently from traditional approaches in literature, the SLS performs inferences on the user's activity without the need of a back-end server for the data elaboration. The activity classification algorithm is lightweight to run efficiently on the client side.

To identify the user status, the Sensor Manager analyzes the three axis accelerometer data, calculates a set of features over it and applies a classification algorithm to the features. I selected a set of features which allow the system to distinguish between different categories of movement:

- not moving (e.g., still or standing),
- moving by foot (e.g., walking, running),
- slow vehicle (i.e., biking) and
- moving with a vehicle (e.g., motorbike, car, bus, train).

The classification algorithm has been chosen in order to optimize the trade-off between the simplicity of the algorithm itself (which has to run entirely on the mobile device) and the accuracy of the activity classification: our goal is to exploit a lightweight classification algorithm which performs a reliable inference, guaranteeing a limited impact on the mobile device resources usage (more details in chapter 4).

The presence of the Inference Module inside the SLS is beneficial because it allows a reduced energy consumption: inferring the real activity of the users, the Inference module makes the system self-adaptive and hence able to avoid unnecessary location readings.

2.3 Dataset

To validate my work I performed the preliminary studies (before the realization of the complete system) over two location datasets. Unfortunately, for most of the publicly available location datasets the data corresponds to trajectories [Bracciale et al., 2014], or to WiFi or GSM continuously sampled data [*CRAWDAD data set Microsoft/Vanlan (v. 2007-09-14)*, 2007], or to data collected in a small context (e.g., a mall, a university campus) [Rhee et al., 2009; *CRAWDAD data set cu/cu_wart (v. 2011-10-24)*, 2011] or for a small time (i.e., a conference scenario). To show the performances of our system and to evaluate our solutions we need a continuous sampled location dataset collected over a long sampling duration. Similarly, we need a level of precision that is higher than the one we

can obtain with traditional WiFi or GSM technologies. The datasets we used are: one mainly composed of trajectories (selecting a subset of the data which presents only few discontinuities), and the second one which consists of continuously sampled location data. In the following by using the term *trajectory* we mean a sub-sequence of a continuously sampled location data: trajectories are discontinuous records of mobility data and consequently they do not cover the whole sampling period. These datasets have different characteristics in terms of spatial and temporal distribution of the visited places. However for both of them we only consider the PoIs in terms of geographical places without semantic meaning. We make this decision for two main reasons: firstly to preserve the privacy of the users (users may have issues revealing why did they visit a certain location). A second reason is the fact that a PoI is not an exact location, its dimension (or geographical scope) depends on how the user visits the place: hence a PoI can be a shopping area where the user roams around (which may include shops, restaurants, bars,...), or an office where the user spends most of her/his time sitting in the same location. By showing the validity of our results in both cases, we demonstrate its independence of the dataset characteristics.

2.3.1 Trajectories Dataset

As trajectories dataset I used the one collected by the GeoLife project and released by Microsoft Research Asia (Zheng et al. [2009, 2008, 2010]). The dataset consists of a collection of GPS coordinates related to the movements of 178 people in a period of over 4 years (from April 2007 to October 2011), and it is widely distributed across over 30 cities of China and even in some cities located in the USA and Europe. People participating to the experiment are students, government staff and employees from Microsoft and several other companies equipped with GPS loggers or GPS-phones. Overall the dataset provides 17.621 trajectories [3] with a total distance of $1,251,654$ kilometers and a total duration of $48,203$ hours. With respect to other datasets with mobility data collected in a limited area or in a particular context, Geolife dataset offers a high heterogeneity. As a matter of fact, it contains a broad range of users' outdoor movements, including both everyday routines imposed by working activities and free time activities. For my study, which is centered on the locations visited by the users during their daily lives, the most interesting characteristic of this dataset is its temporal and spatial fine granularity, as 91% of the GPS tra-

[3] A GPS trajectory of the dataset corresponds to a set of consecutive location samples (timestamp, latitude, longitude, altitude).

jectory are recorded with a dense representation, every $1 \sim 5$ seconds or every $5 \sim 10$ meters per location sample. However, the dataset has been built for the transportation prediction task, and thus does not directly characterize the Points of Interest[4]. Furthermore, if on the one hand the dataset is very rich, on the other side it exhibits an high level of fragmentation, especially with regard to features as the effective duration of the trajectories, the data collection period, the number of trajectories per user, the duration of the gaps between discontinuous trajectories. Indicatively, more than half of the trajectories spans less than one hour, while about 60% of users collected data for less than a month. For those reasons, GeoLife dataset requires large pre-processing for reducing the trajectory points and extracting the visited points. This modification affects also the suitable number of users and the daily traces, per user, from which we can extract the PoIs.

2.3.2 Continuous Dataset

Although GeoLife represents the most reliable dataset publicly available for our purpose, even after preprocessing, its nature still remains trajectory centered, and it differs from a continuously sampled dataset. As opposite to the Microsoft one, which is a large dataset collected in a metropolitan area, I supervised a local collection campaign to retrieve a dataset of continuously sampled GPS coordinates, during users' daily routine, in a small area environment. A real experiment has been conducted to collect traces in the area of Ticino and North Italy, over a time period of 20 days, from a group of 12 users (Papandrea and Giordano [2014]). The data collection system have been installed on the primary mobile phone of the users, to ensure they continuously carry it with them. Different devices running several versions of the Android OS (table 2.1) have been used to collect data.

The mobile phone sampling service performs a location reading every 60 seconds, and works both outdoor and indoor. The location information is provided by the Android OS Localization Manager which queries both GPS and Network (WiFi or GSM) Providers. The service selects the best location information available at each sampling minute, and stores locally the retrieved data. The service runs continuously, collecting data 24 hours per day in the best case, for the whole duration of the experiment. For privacy reasons, we allowed the users to manually pause the service. Thus, the collected data is not always a 24 hours

[4]Point of Interest (PoI) is an interesting location for a user, where she/he spends a relevant amount of time. More details will we presented in section 5.3

Device Model	Adroid Version
Samsung Nexus S	4.0.4
HTC Nexus One	2.3.6
HTC Desire Z	2.3.3
Samsung Galaxy II S	2.3.4
Garmin-Asus nuvifone MP	2.1.1
Samsung GT I9100	2.3.3
LG-P920	2.3.5
HTC Hero	2.1.1
LG-P990	2.3.4
HTC Desire	2.3.7
HTC Sensation	4.0.3
Motorola Milestone BP6X	2.1.1

Table 2.1. Devices Specifications

continuous data, but may present some gaps. Also from this dataset I select a subset of significant users which have collected a significant amount of data. More specifically, I selected the users which collected at least 14 relevant days of data (two weeks), where a relevant day includes at least 6 hours of location sampling. The resulting number of relevant users considered for my study is 6.

2.4 Conclusion

In this chapter I presented the solution to the localization problem, proposed in this thesis. I presented each module composing the SLS from the functional point of view. The *Inference module* is in charge of exploiting the potentialities of the accelerometer to enhance the localization procedure, and in particular it helps the SLS to adapt its behavior according to the actual activity of the carrying user. The *Learning module* and the *Prediction module* exploit the computational capabilities of the mobile device, and learn information about the mobility habits of the user, in order to be able to predict the next movements.

All these modules will be better explained in the next chapters: they will be firstly evaluated separately and independently from the system. At the end of the thesis the modules will be presented and evaluated as part of the whole SLS system.

Chapter 3

Analysis of Battery Consumption

3.1 Introduction

In this chapter I present two different methodologies to evaluate the battery consumption on Android mobile devices. The main purpose of this thesis development is to study and to realize a smart methodology for a continuous localization service for mobile devices. Therefore, to be able to evaluate the system and to show its validity in terms of smart usage of the battery, it is essential to firstly analyze the measurement possibilities.

Many existing works related to the smart usage of the mobile device's resources/hardware proposed different solutions for the battery measurement. In Wang et al. [2009], the authors measured the sensor power consumption of their employed mobile devices (Nokia N95) through the Nokia Energy Profiler, a stand-alone application that allows the monitoring of the running applications' energy usage in real time. Lin et al. [2010] experimentally measured the energy usage for multiple location modalities on an AT&T Tilt (HTC TyTN II) mobile phone, removing the battery and supplying the power by means of a Monsoon Solutions Power Monitor. This approach is very similar to the one implemented in this thesis, and presented below in section 3.3. Also Paek et al. [2010] used the Power Monitor device from Monsoon Solutions Inc. for their power measurements and cross-verified them with the Nokia Energy Profiler v1.2 software tool (they performed their studies with the Nokia N95 smartphone). In Flipsen et al. [2012] the authors studied the power breakdown of five different smartphones, measuring the power consumption by means of an Agilent N6705A DC Power Analyser, and emulating the battery with an Agilent Power Analyser and attaching probes to the battery input pins present on the smartphone. This last approach is very similar to our implemented one, however for many mobile de-

vices this methodology is not applicable (for example, for the HTC Google Nexus One phone), where the device implements a general check of the hardware components at the bootstrap: the ah-hoc battery necessary to connect the mobile phone to the power supply and to the multimeter is not recognized as a compatible component, and the device does not switch on. An interesting approach has been described by Ferrari et al. [2014] which presented POEM (Portable Open Source Energy Monitor). This system allows the measurement of the energy consumption of every single Android application component down to the control flow level. Their approach is based on the Arduino Leonardo board, which is basically used to read the current flowing into the mobile device.

All these approaches described in the state of the art are very interesting, and allows a fine grained measurement of the mobile device energy consumption. However, they have a common drawback: they are static systems which do not allow the energy measurement while the mobile device is carried by the user.

I present in this chapter two measurement procedures. These are applied in the thesis for the battery consumption estimation for both static measurements and evaluation of mobile scenarios. In section 3.2 I show the difference between two localization procedures by means of the Android API and their different impact on the battery resource. In section 3.3 I explain the two battery measurements methodologies I applied while developing and evaluating this thesis, explaining the pros and cons of both approaches. Finally in section 3.4 I show the difference in battery consumption for different tasks on the mobile device, and especially for the tasks performed by the SLS: this evaluation motivates the SLS reasoning explained in the following chapters of the thesis.

3.2 Differentiation of localization procedures

The localization is an expensive task for a mobile device, in terms of battery consumption. The amount of battery necessary to localize a user strictly depends on two factors: on the methodology used to perform the localization and on the user's mobility. The study reported on this thesis has been performed by using Android mobile platforms.

Android offers mainly two data sources to retrieve location information:

- the GPS provider, which requires the line-of-sight to GPS satellites and retrieves in most of the cases very accurate location data;
- the NETWORK provider, which retrieves a more coarse location information exploiting the WiFi and GSM interfaces.

A third provider has been introduced into the 8th Android API level in 2010: the PASSIVE provider. This is different for the others: it is a special location provider which can be used to passively receive location updates from other applications querying directly the previous listed providers. According to its nature, the passive provider is not reliable. In fact if there are no other applications or services performing localization on the device, it will never get any location information. For this reason, in my work I always used an active localization approach, directly asking for location updates to the GPS and NETWORK providers.

The Android Localization manager gives the possibility to specify directly the provider we are interested in, while performing localization. Or alternatively, if many providers are available, it allows us to specify a set of criteria, and it is then the Localization Manager which select the one which matches better the specified criteria, according to the available ones. Specifying a set of criteria makes the localization procedure strictly dependent on the user's mobility. For example, consider we have to perform a regular tracking of a user, reading his location every 60 seconds for an interval of time of one hour, then the total number of readings expected is 60 (one per minute). If we specify a set of criteria to select a suitable localization provider, which includes that the accuracy of the localization has to be set to *fine*[1], we can expect that the provider selected for this task is the GPS. However the selection of the provider strictly depends on the current context of the mobile device: if the device is indoor, the finest possible localization is reached by the NETWORK provider, while if it is outdoor obviously the finest localization may be retrieved by the GPS provider. This means that, within 60 location readings, the amount of them performed by the GPS, and the ones performed by the NETWORK provider depend on the movement of the user. However, it is guaranteed the number of location updates. If, instead, we specify the name of the provider, for instance GPS, we will not retrieve updates in case the user visits indoor locations, or when the GPS is not reachable.

During the experiments performed to evaluate the battery consumption given by the localization procedure, I always specifies the location required criteria. In this way I am always sure about the number of location readings performed per interval of time.

The number of expected location readings is important, as shown in the next chapters, for the application of a Density Based algorithms on the location data. In order to apply this algorithm to the location data and to be able to identify clusters of locations according to their geographical density, it is very important

[1]Android terminology

the data distribution to be quite uniform in time.

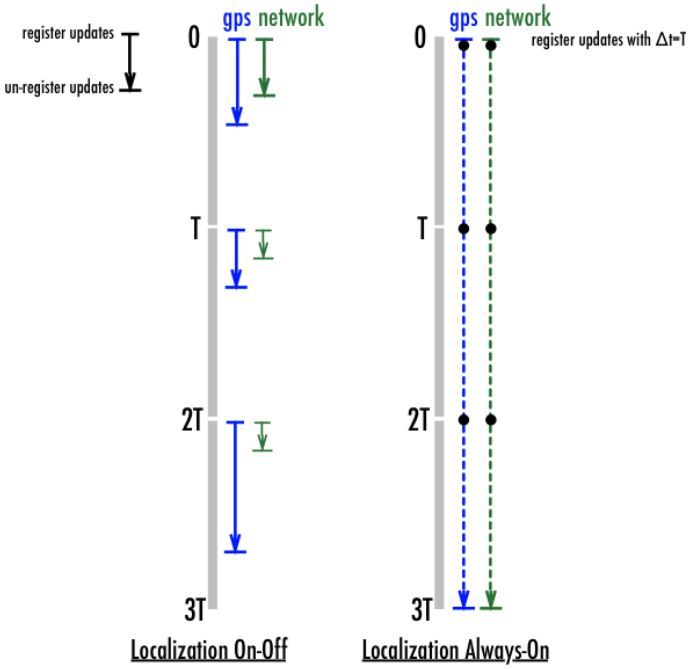


Figure 3.1. Localization mechanism

In this section I evaluate two different mechanisms in performing the localization, by means of the Android providers. The Android Operating System, or more specifically, the Android Localization Manager (middleware between the real android localization interfaces and the Android third-party applications), gives the possibility to register for location updates, specifying the querying period. In this way the operating system provides the requesting application the updated location information, at approximately the specified frequency. However, this procedure is quite expensive in terms of battery consumption, because the localization procedure is "*always-on*": the specified frequency does not impact the localization itself, but affects only the current location notification frequency, which is sent to the requesting application.

To slightly reduce the battery consumption, it is possible to implement an alternative mechanism, switching ON and OFF the localization procedure, as required. Figure 3.1 shows the two different localization mechanism: the *always-on* on the right, and the *on-off localization* on the left. This second procedure, differently from the previous one, sends requests for location updates regularly at a specified period (T in the figure), and stops the localization as soon as it received the first location update. As visible from the figure, this second mecha-

nism has a not predictable response time, in fact, it strictly depends on the user's context. If the user is visiting an environment in which the providers are not easily reachable, the localization updates may need a long delay before being available, and the battery consumption may be similar or even higher than the one of the localization always-on procedure.

3.3 Battery consumption measurement

The Android operating system allows the monitoring of the device's battery level. In particular, we can evaluate different tasks by measuring the amount of battery discharged while running them during a fixed interval of time. However this battery measurement procedure does not give us an accurate evaluation of the energy used by each specific task, but it gives an idea of the difference in the *amount of required battery*. To give an example, I measured the lifetime of an Android phone, and the corresponding decreasing battery levels, while it was performing different tasks. Figure 1.2 shows the battery lifetime of a Samsung Galaxy Nexus device running the Android operating system, version 4.1.1. The device is running a continuously localization service, which requests updates in parallel to both GPS and NETWORK providers. The device is set to its factory state, and the only third-party service running is the localization-battery measurement application. There is no SIM card (then no connection to any GSM tower), there is no data traffic exchanged (WiFi enabled but not connected), the Bluetooth interface is disabled (since it is not necessary for our experiments) and the device is always static and in the same position for all the measurement sessions. The device is positioned indoor, but with GPS signal reachable (near by a window).

The instances shown in the figure represent the device running both Localization On-Off and Localization Always-On procedures (respectively *GPS ON-OFF* and *GPS ON* in the legend). For the *ON-OFF* instances, I applied different duty-cycles, which correspond to the periods between consecutive location update requests. To be able to properly evaluate the device lifetime, I show in the figure also the *baseline* instance (device not running any localization procedure) and *continuous instance* (device running the localization requesting continuous updates).

As visible from the figure, we can firstly notice that the residual battery decreases monotonically, but the slope is not constant. This is caused by the availability of the GPS signal and by the additional OS activities and running native services which also have a not negligible impact on the battery life. From this

measurements we cannot accurately evaluate the energy necessary to perform each localization methodology. We can just infer that the localization procedure which implies the switching ON the Localization providers updates periodically, and switching them OFF immediately after receiving the data, is slightly less expensive than asking for periodic location updates keeping the localization interfaces always ON. We can also notice the great difference in battery lifetime duration, when we continuously ask for location updates (with both GPS and NETWORK providers) and contrarily when we measure the baseline battery usage. The baseline, as expected, has the longest battery lifetime since the device is not performing any particular task, and the only contribution to the battery usage which is measured is due to the operating system. The same measurements have been performed with other devices in the same conditions and the obtained results are similar. I decided to report in this section only one example because it was not possible to average among them: in fact, I used different platforms running different versions of the Android OS.

As shown above, it is clear that by means of the Android API battery level monitoring we cannot perform an accurate study of the energy used by the mobile device, while performing a certain tasks. Hence I decided to perform the energy measurements on the mobile devices by means of an external *Multimeter*.

For the purposes of the energy measurement, I created a circuit as illustrated in figure 3.2 and figure 3.3. The circuits provides power to the Android device by means of a Power Supply DC (HP E3630A Triple Output DC Power Supply). In series with the Power Supply and the Smartphone I inserted a Multimeter (Agilent 34411A Digital Multimeter) to be able to measure the current flowing in the circuit, hence the current used by the phone.

To be able to insert an Android Smartphone in this circuit I used an ad-hoc battery-box which plugs the +/- connections of the Power Supply with the corresponding pin-out of the smartphone (the picture of the ad-hoc battery is shown in figure 3.4).

The device used to perform the battery measurements by means of the circuit shown above is the Samsung Galaxy Nexus, running the Android OS version 4.2.1. The voltage provided by the Power Supply is set to $3.7Volt$, according to the device battery specification.

The multimeter inserted in the circuit is able to perform measurements at different sampling intervals. According to its technical specifications, the maximum sampling frequency we can set for the measurements is 50KHz. Unfortunately, at this frequency we cannot sample more than 20 seconds of data, because of the device inner buffer limited dimension.

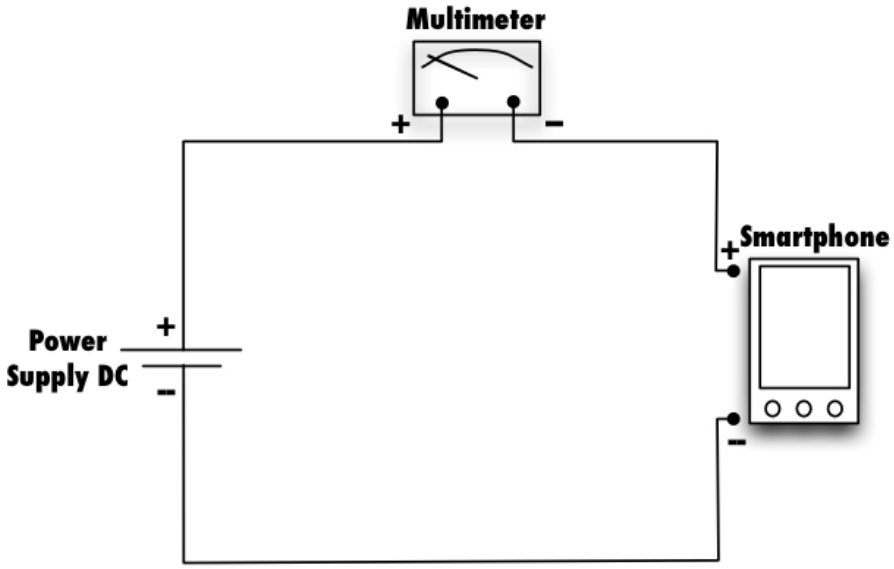


Figure 3.2. Circuit for energy measurements

A first study performed with this circuit, aimed to the measurement of the difference in the amount of information retrieved by setting the multimeter at different sampling frequencies. I measured the energy used by the mobile device while performing a continuous localization task (without duty-cycle) querying only the GPS provider, in an environment where the GPS signal is reachable, keeping the phone static to a fixed position. The phone is set to its factory status, no Google account is registered, all the network interfaces are disabled (WiFi, Bluetooth), and the device is not provided with any SIM card: this way I measure mainly the energy used by the phone to perform the localization task, in addition to the baseline energy required by the operating system.

To retrieve the energy used by the localization service implemented on the mobile device, I firstly measured the current flowing in the circuit for a duration of 20 seconds [2] for different sampling frequencies. I calculated the mean value of the measured current over the sampling interval:

$$A_{mean}[A]$$

To calculate the Power I multiply the voltage provided by the Power Supply to the mean current value:

$$P[W] = Voltage[V] * A_{mean}[A]$$

[2]This duration correspond to the longest possible measurement duration with the largest sampling frequency

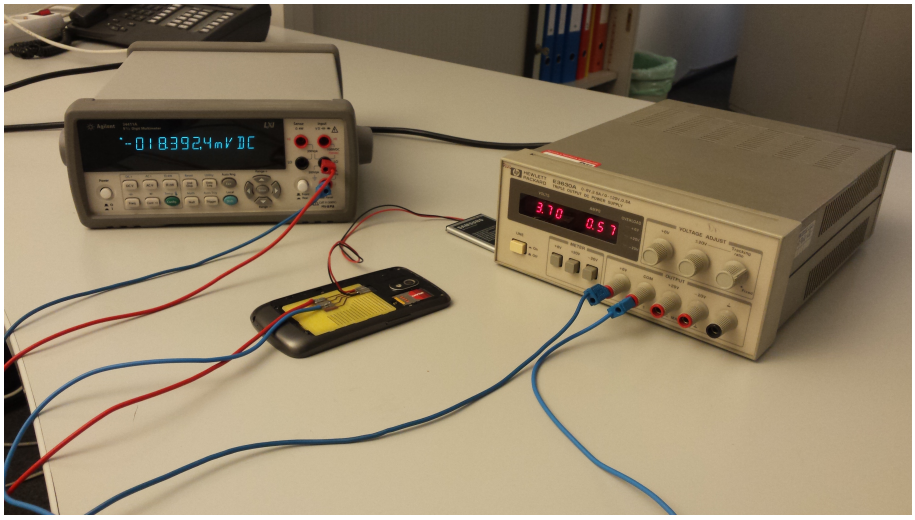


Figure 3.3. Circuit for energy measurements

This value correspond to the Power drain of the battery performing a specific task. To finally calculate the Energy consumed over a certain amount of time, I multiply the Power to the sampling interval duration in hours (in this case $20sec = 0.0056hour$).

$$E[Wh] = P[W] * 0.0056[h]$$

Table 3.1 contains the Power drain (the mean and standard deviation) related to the current measured with different sampling frequency.

Sampling frequency	**P[mWatt] mean**	**P[mWatt] standard deviation**
50 KHz	238.2	9.6
5 KHz	233.5	3.1
500 Hz	234.7	2.1

Table 3.1. Power measurement at different sampling frequencies

As visible form the table, the calculated mean power drain is very slightly affected by the different sampling frequencies of the multimeter. Therefore I decided to continue with the measurement of the current flowing to the mobile device, to calculate the Energy consumption, by setting a sampling frequency of the multimeter equal to 500Hz. This setting will allow longer measurements.

Figure 3.4. Ad-hoc battery for energy measurements

3.4 Comparison of the battery consumption for different tasks

In this section I show the difference in the amount of energy necessary to perform different tasks by means of a mobile device. In particular, I measured the energy necessary to run a localization service, to get data from the embedded acceleration sample, and finally I compared this measurements with the baseline energy required by the operating system.

Table 3.2 shows the mean and standard deviation of the power drain in milliwatt (mWatt) measured by means of the Android device described before.

These measurements strictly depend on the used device, on the hardware of the embedded sensors and on the Android OS version, hence it may be quite different for other platforms. However, the ratio between the different measures is quite interesting, and it could be considered valid for many devices. As expected, the difference in the amount of energy necessary by the phone while being idle

	P[mWatt] mean	**P[mWatt] standard deviation**
GPS	234.7	2.1
Accelerometer(Fastest)	210.0	5.6
Baseline (screen OFF)	17.6	1.2

Table 3.2. Power drain associated to different tasks

(baseline energy, when the device has the screen off), and the amount necessary for the other two tasks is quite high. However, the amount of energy necessary for the GPS localization (GPS receiver signals processing) and the Accelerometer sampling (at the sampling period of 100Hz) is quite similar. However, the purpose of this study is to provide a motivation for the SLS strategy. As explained in the next chapters of the thesis, one of the main strategies of the SLS in order to reduce the energy consumption given by the continuous localization system, is to use the accelerometer for an activity inference study. The inference will be performed, in the worst case, sampling the acceleration data for an overall duration of 5 seconds, every minute. It means, with a Power consumption of $210 mWatt$ for $5 seconds$ of sampling, the amount of energy used corresponds to $0.294 mWh$. For the GPS instead, the SLS performs, in the worst case, a location update every 60 seconds. In case of only GPS provider available, if the location updates arrives in 5 seconds, with a Power consumption of $234.7 mWatt$, the amount of energy used corresponds to $0.326 mWh$, which is slightly higher than the amount used for the accelerometer sampling. However, in most of the cases, the GPS provider need a longer time to get an update, which strictly depends on the environmental context. To give an idea, in the worst case, if the GPS needs 60 seconds to retrieve the updated data (it waits for the complete update period), it requires an amount of energy corresponding to $3.911 mWh$, which corresponds to 13 times the energy spent for the accelerometer sampling.

The presented difference in energy consumption reinforce the main open problem, which motivates this thesis. Given the high cost in terms of energy consumption associated to the localization, it is clearly necessary the application of a smart methodology to be able to provide a continuous localization service. With this study I quantified the amount of energy necessary for the Accelerometer sensor sampling and the GPS reading, quantifying the energy saved in choosing a study on the accelerometer instead of a direct GPS reading. The precise energy consumption on the GPS reading in a mobile scenario, however, depends on the visited contexts. For this reason I mentioned the best/worst cases in the energy evaluation.

3.5 Conclusions

In this chapter I presented two different methodologies to measure the battery consumption of a mobile device, to be able to evaluate specific tasks. In particular, I presented the possibility offered by the Android APIs to monitor the battery discharge. This methodology is quite portable, because simply logging the time-stamped battery level changes, it is possible to have a qualitative evaluation of the dependency between the battery lifetime and the tasks running on the device. The second methodology presented consists in a quantitative measurement of the energy used by the device. Applying this methodology, it is important to set the device such that the main impact on the battery usage is given by the target task: in fact, this methodology allows the measurement of the overall energy used by the mobile device.

The main difference between the two methodologies consists in the obtained results: the first one provides a more qualitative evaluation, while the second one gives a quantitative measurement of the energy consumption. However, the second methodology implies the usage of an ad-hoc measurement circuit, which therefore does not allow the evaluation in scenarios where the device is moving (carried by a moving user). Therefore, during the development of my thesis, I applied both the presented methodologies. The second one (ad-hoc measurement circuit) allowed me to set important parameters for the SLS systems, and to strengthen this work's motivation. The first methodology has been used in the evaluation of the final experiment, when the complete SLS system was validated against a continuous localization system, in order compare the battery lifetime, while moving.

Chapter 4

Activity Inference

To further enhance the mobility model implemented by the SLS and described in details in chapter 6, I exploit the sensing capabilities of nowadays smart-phones. My main goal here is to optimize the resources used by the localization procedure, by interpreting the accelerometer data to dynamically trigger location readings, according to the users activities. In this chapter I present the *Inference Module*, how it works together with the other modules of the SLS and which is its task inside the system. In section 4.2 I give an overview of the state of the art on the field of *activity inference*, and in particular, I present in section 4.2.1 the major related works on context recognition, where the sensor data is used to infer the mobile user's context with the goal of enhancing a particular service. In section 4.2.2 I present the related works on activity inference, and in particular I concentrate on the used methodology and algorithms. In section 4.3 I present the two main experiments I performed to collect data for the inference study. Section 4.4 focus on the raw data collection on the mobile platform, and on the study about the sensor employed by the Inference module and the sampling rate. In section 4.5 I explain the decision about the data inference window and subsequently, in section 4.6 I give more details about the classification algorithm: the activities recognized, the features extracted from the sensors data and the classification algorithm. Finally, section 4.7 reports the experiments performed to evaluate the algorithm implemented by the Inference module, and the measured performances.

4.1 Introduction

The main task of the *Inference Module* is to perform activity inference and mobility tracking. Differently from many other existing solutions, the presented

algorithm runs locally on the mobile device, thus saving bandwidth and energy cost for the data transmission, and it is able to recognize various types of activities with high precision. In this chapter I will present the algorithm used and its technical precision. Furthermore, at the end of this thesis, in chapter 7 I will evaluate the *Inference Module* in terms of energy saving when inserting it within the SLS.

4.2 State of the Art

Sensors have been used in quite recent works (Reddy et al. [2010]; Miluzzo et al. [2008]; Ofstad et al. [2008]; Kim et al. [2010]; Bolliger et al. [2009]; Bamis and Savvides [2010]) to infer the context a user carrying the phone belongs to. I will present in this section the State of the Art in the research area of activity inference. The related works are separated in two different sections: in the first one (section 4.2.1) I will present some main existing works on the usage of sensors as mean to infer the user's context, with the purpose of enhancing the localization or other mobile services. In the second section (4.2.2) I focus more on the activity inference methodologies, the sensors used, the algorithms and features, and the energy constraints.

4.2.1 Context Recognition

Lot of work has been recently done to retrieve information about the phone's context and user's activity, by analysing data retrieved with sensors embedded into the mobile device, with the purpose of enhancing other services (such as, localization, daily life monitoring, health care, etc.). In particular, in my work I am interested in the usage of the activity inference performed by means of mobile phones, providing context-awareness in localization services and/or applications.

Ofstad et al. [2008] presented a system called AAMPL, whose goal is to enhance the mobile phone's context recognition, by adding the analysis of accelerometer signatures to location information. They argue that the physical location by itself may not be sufficient to define the user's context, hence they use the accelerometer data to classify the business category (e.g., restaurant, fast food, retail store) the user belongs to. The data is collected by a three axis accelerometer embedded into the phone and placed on the right pant's pocket of the user. The classification is done in two separate stages: the first one, performed on the client side, classifies the sitting or standing activity of the user by

exploiting a Bayesian classifier; the second stage, performed on the server side, uses a set of three features (percentage of points that are in the standing state, average variance over all three axes for points in the standing state and amount of time in seconds) collected in the first stage, to perform the business category classification, by using a K-Nearest Neighbors classifier.

Miluzzo et al. [2008] presented an application called CenceMe, which uses sensor-enabled mobile phones to combine the inference of the presence of individuals with sharing of this information through social networking applications. The personal sensing presence is derived from a two-levels classifier, which runs both on the client and on the server side. In this case the client side is responsible of the classification of some primitive features (e.g., sound classification, activity inference, Bluetooth MAC addresses in the phone's vicinity, GPS reading and random pictures), while the back-end classification returns facts (e.g., conversation classifier, social context, mobility model detector, location classifier). What is relevant to my work, is the classification of the activity, resulting in a *Primitive feature*. They analyzed the three axis accelerometer data, and in particular a set of three features evaluated from the raw data (mean, standard deviation and the number of peaks per unit of time) to classify the user's activity (sitting, standing, walking, running). They used a supervised classification algorithm: after the training phase, they feed the training set to a J48 Decision Tree algorithms. This algorithm has been demonstrated to be lightweight and efficient, being able to complete the classification process in less than 1 second.

A transportation mode classifier is also presented by Reddy et al. [2010]. The classification system works outdoor and uses accelerometer data in conjunction with GPS coordinates to identify whether the user is stationary, walking, running, biking, or in motorized transportation. GPS data provides speed information, which is then placed in the corresponding range of values, associated to a transportation mode. Accelerometer data instead, is used when the activity causes a change in motion: the value of the variance is used to classify the activity. Also in this case, a time window of 1 second is used for the classification process. The features used are, in terms of the three axis accelerometer, the magnitude of the force vector, and the mean, variance, energy, and Discrete Fourier Transform energy coefficients, based on this magnitude. The classification system consists in two stages: a decision tree followed by a first-order discrete Hidden Markov Model, and achieves an accuracy level of 93.6%.

Kim et al. [2010] presented in SensLoc a simple movement detection mechanism, where the accelerometer is used in a smart localization service, to find the opportunities to save energy when the device is stationary. They consider the acceleration magnitude over the three axis to tolerate random orientations of the

device. When the variance of the magnitude is above a certain threshold, they consider the opportunity to start the localization mechanism. This service also uses GPS for path tracking and WiFi for place detection. With this smart system they reached to consume 13% of the energy than algorithms that periodically collect coordinates, correctly detecting 94% of the place visits and tracking 95% of the total travel distance.

Song et al. [2013] proposed a solution for improving indoor localization by employing sensors embedded into mobile devices. More specifically, they develop a floor localization system for minimizing the delays in emergency call responses. They used accelerometer, gyroscope and magnetometer to calculate the vertical displacement of people moving among different floors inside buildings, by means of elevator, stairs and escalator.

Table 4.1 shows explicitly the differences between the main approaches presented above, in terms of utilized sensor data, activities recognized, algorithms employed and extracted features; additionally I present a quantitative comparation of the related works in terms of measured recall.

In my work I will also exploit the sensors embedded into mobile devices to retrieve context information, with the purpose of reducing resources consumption due to localization procedures. Differently form many existing solutions, the inference task will be performed entirely on the mobile by the SLS, preventing any privacy issues with the transmission of the user activity information to other nodes. The evaluation of the proposed solution is then performed measuring not only the accuracy of the algorithm, but more emphasis will be given to the actual energy saved by the SLS when including the *Inference Module* within the system.

4.2.2 Activity Inference

In this section I give an overview of the state of the art in Activity Inference, summarizing the sensors employed, the algorithms used, the learning techniques, the activities recognised, and the constraints with the energy consumption issues. At the end of this section I will present the open challenges which I will face with my work.

Activity inference, also known as transportation-mode recognition or activity recognition, has recently been a very active research area. This research direction encountered the growing capabilities of recent years mobile devices,

	Sensor Data	Activities	Algorithms	Features	Precision
Ofstad et al. [2008]	3-axis accelerometer (right pants' pocket)	sitting and standing (client), business category (server)	Bayesian classifier (client) and K-nearest neighbors classifier (server)	num point (time), standing variance, % standing	98.9%
Miluzzo et al. [2008]	3-axis accelerometer	sitting, standing, walking, running	J48 Decision Tree Algorithm (client)	mean, standard deviation, number of picks per unit of time	̃79% with 1 second inference window
Reddy et al. [2010]	3-axis accelerometer and GPS	stationary, walking, running, biking, motorized transportation	Decision Tree + First-Order Discrete Hidden Markiv Model	GPS speed value and mean, standard deviation, energy, Discrete Fourier Transform, energy coefficients of the acceleration magnitude	̃93.7% with 1 second inference window
Kim et al. [2010]	3-axis accelerometer	static and in movement	threshold comparation	variance of the acceleration magnitude	̃90% with a delay of 120 sec
Hemminki et al. [2013]	3-axis accelerometer	stationary, walk, bus, train, metro, tram, car	Discrete Hidden Markov Model + Instance based classifier (Adaptive Boosting)	Different domains: statistical (10 features), time (4 features), frequency (11 features), peak (5 features) and segment (14 features)	̃80%

Table 4.1. 2: Energy-Efficient Location-Based Solutions comparison

characterized by impressive computing, networking and sensing powers. As a consequence, various available mobile markets and many researchers started providing and developing context-aware applications and services for mobility tracking and optimization, patient and general health monitoring (Puiatti et al. [2011]), smart home applications (Robles and Kim [2010]), etc. The main core of all these applications consists of an activity inference algorithm, where raw sensor data is collected and analysed to infer human activities.

Sensors

Researchers have attempted to use various on-board and external sensors to identify the current activity of the users (Su et al. [2014]). Some of them use body sensors (i.e., wrist sensors) to identify fine-grained activity, mainly physical exercises (Keally et al. [2011]; Cheng et al. [2013]). Many algorithms employ GPS only data (Xu et al. [2010]; Gonzalez et al. [2010]; Patterson et al. [2003]), accelerometer only data (Yan et al. [2012]; Hemminki et al. [2013]; Cheng et al. [2013]; Keally et al. [2011]) or both (Reddy et al. [2010]; Ofstad et al. [2008]). Few solutions use barometer or gyroscope (Jonathan Lester [2006]; Ustev et al. [2013]; Muralidharan et al. [2014]; Vanini and Giordano [2013]), mainly because of their high energy costs. Some researchers turned their attention to context information such as GIS data (Stenneth et al. [2011]). This approach is very well suited for large metropolitan areas, where GPS coverage suffers from the skyscraper tunneling effect and the GSM cells dimension is reduced.

Algorithms

Many related works use different machine learning classifiers for activity recognition. The most used one is the Decision Tree (Miluzzo et al. [2008]; Siirtola and Röning [2012]), in fact it is a lightweight algorithm suitable for a resource constrained mobile platform. Furthermore, this algorithm allows a hierarchical classification of human activities. Other machine learning algorithms used in recent works on activity inference are: Decision Tables (Bao and Intille [2004]), easy to implement but unable to identify hierarchies of activities; K-Nearest Neighbors (Ustev et al. [2013]),which is similar to Decision Tree in terms of computational complexity; Hidden Markov model (Jonathan Lester [2006]), which is a good candidate in capturing the transitions between different activities; and Support Vector Machines (Anguita et al. [2012]) which unfortunately have an higher computational cost.

Learning techniques

Another important issue is where the training of the classifier takes place, if on a server (offline) or on the smart phone (online). The work of Reddy et al. [2010] trains the classifier offline, but does the classification of the activities online. Thus, it does not need to access a server for data processing, hence it saves energy. PBN (Keally et al. [2011]) is a sensor-node Android smartphone system, where the AdBoost training is performed online. However, the underlying classifiers used for boosting are trained offline. NuActiv (Cheng et al.

[2013]) is one of the leaders, where all of the software is implemented online, even the support vector machine classifier. However, this classifier achieves only maximum 80% accuracy.

Activities recognised

Another dimension of all existing works is which activities they are targeting. Some focus on daily routine activities, such as running, sitting, watching TV, etc.(Miluzzo et al. [2008]). Others target special sport activities (Keally et al. [2011]; Cheng et al. [2013]), while others focus on transportation modes (Hemminki et al. [2013]; Stenneth et al. [2011]; Reddy et al. [2010]; Patterson et al. [2003]). Among the works focusing on transportation-mode recognition, there are further differences. Some target only a very limited set of activities like still, walk and vehicle (Patterson et al. [2003]), also called *Simple Activities* (Su et al. [2014]), others focus on motorized vehicles and walk only (Reddy et al. [2010]). Only few of them include more precise transportation modes, such as metro or tram (Hemminki et al. [2013]).

Energy constraints

Much research has also been focused on making the algorithms energy efficient, e.g. through reducing the sensors sampling rate and the data processing. In fact, one of the main issue in activity inference is the sampling rate. Sampling from the sensors at an high rate provides more information but also it may introduce more noise and use more energy. However, a low sampling rate requires less energy but may lead to a loss of information. Special middle-ware for managing sampling rates and used sensors have been developed (Lu et al. [2010]; Wang et al. [2009]) or adaptive algorithms have been employed (Reddy et al. [2010]; Lim et al. [2013]). These efforts have shown to reduce energy consumption, but it is unclear whether the solutions are usable or not - e.g. whether the smartphone can survive a typical working day without recharging. For example, Reddy et al. [2010] reports a lifetime of approximately 7 hours. On the other hand, the data processing has been typically offloaded to a server for energy preservation, which is not necessarily the optimal solution, as it requires network communication.

Main open challenges

Summarizing, according to the analysis of the state of the art presented above, activity recognition is a very well studied topic. There exist many algorithms

which employ different sensors and machine learning algorithms to classify varying classes of activities. However, a closer look especially on studies where existing algorithms have been reproduced show us a different picture. There are still several issues and open challenges.

1. Typically the only reported precision of the algorithms is the recall precision - i.e. using the same dataset for training and testing. This is a major issue for many machine learning algorithm. The recall precision is generally very high, while a user-testing precision, especially in this data domain typically drops significantly. The user-testing precision is calculated over an experimental collected dataset, independent from the training dataset, and the correctness of the inferred activity is defined by the user itself (i.e., similar activities might be considered equivalent by a user, as for the case of strolling and walking, and traveling on a bus or on a car). The only work among the ones presented above, which actually evaluates the user-testing precision is Hemminki et al. [2013], where the reported mean precision of 82.1% is much lower than for others.

2. Which classifier to use? It seems like any would do the job, but different classifiers have different properties. Some are able to better generalize the results and thus achieve higher testing precision, others are very lightweight and can be easily implemented pervasively. Selecting a lightweight classifier is crucial.

3. Which activities can be reliably recognized? Many researchers report about problems differentiating between various walking modes (e.g. running and walking) or between different motorized vehicles. For example, Hemminki et al. [2013] included “train” as recognized transportation mode, however achieving only 68% precision.

4. Energy efficiency. User-oriented energy efficiency is a major requirement. This is, the smartphone must survive at least a full working day before recharge, even with continuous application run.

Considering the main issues introduced above, I present here the “Inference Module” of the SLS which tackles exactly these open challenges, and implements a pervasive solution with high accuracy. It uses a lightweight algorithm and ensures an energy efficient service. In this chapter I will evaluate the accuracy of the Inference Module in terms of the traditional recall calculation. In chapter 7, this module will be considered in conjunction with the whole system, and evaluated in terms of user-oriented precision.

4.3 Experimental data collection

In this section I present the details about the two datasets used for the development and analysis of the Inference Module. In the first part of my work I performed a preliminary study with a small set of data, to evaluate the feasibility of the Inference Module, to select a set of sensors involved in the inferential task, and to identify the set of significant features. This dataset is presented below in section 4.3.1. During a second phase of the SLS development, I performed a campaign to collect a bigger dataset, involving more people, more devices and more activities. Finally I used this second dataset to implement the algorithm adopted by the Inference Module and to test it. Details about this dataset are reported below in section 4.3.2.

4.3.1 First data collection campaign: feasibility study

A small dataset has been collected, as stated before, for a preliminary study. This dataset includes acceleration data for 31 minutes, collected by 6 different people, carrying in total 9 different devices. The users involved in this data collection have been asked to log their activities while being still, walking, biking or driving (table 4.2). These users placed their devices in different positions with respect to their body (i.e., inside the jacket's pocket, into the hand bag, in the backpack, etc.). In this way, the dataset allows the implementation of an inference algorithm which is body-position tolerant, and which is trained by people with different body movement style.

Sampled Data [minutes]	31
Number of users	6
Number of devices	9
Activities performed	still walking biking driving

Table 4.2. Preliminary data-set specifications

4.3.2 Final data collection campaign

After the preliminary study, a bigger data collection campaign has been carried out, where the purpose was to collect more data, by more people, performing

a larger variety of activities. The goal was the implementation of the Inference Module of the SLS, and the analysis of the performances of the activity inference algorithm, while predicting more specific activities.

The collected dataset includes more than 53 hours of linear acceleration data, involves 8 users carrying in total 10 different devices, and performing 9 different activities (table 4.3). Also in this case, the dataset allows the implementation of an inference algorithm which is body-position and body-movement style tolerant. As visible form the table, the activities are divided in 4 groups according to their similarities. In fact during my study, this dataset has been used to implement both the algorithm to infer each specific activities, and also to predict the group an activity belongs to (more details in section 4.6.1).

Sampled Data [minutes]	3197
Number of users	8
Number of devices	10
Activities performed	still
	stand
	walk
	run
	bike
	motorbike
	bus
	car
	train

Table 4.3. Final data-set specification

4.4 Raw data collection

In this section I will present the data collection phase of the Inference Module. The related studies about the sensors used and the sampling rate.

4.4.1 Sensors

The *Inference Module* samples raw data from the mobile device embedded sensors, to perform activity inference. Initially I evaluated the possibility to involve different sensors, and in particular:

(a) Device-coordinate system (b) Globe-coordinate system

Figure 4.1. Device and Globe coordinate systems

- the **acceleration** sensor, which measures the acceleration applied to the device in m/s^2, including the force of gravity. This sensor output data is provided in three components, which correspond to the projection of the acceleration force on the x, y, and z axes of the device coordinate system (figure 4.1a);

- the **magnetic field** sensor, which measures the geomagnetic field strength along the x, y, and z axes of the device coordinate system;

- the **orientation** sensor, which measures the rotation of the device along the x (*Pitch*), y (*Roll*) and z (*Azimuth*) axes of the globe coordinate system (fig. 4.1b).

After an initial study of the first results obtained by considering the above mentioned three sensors, I selected the accelerometer as the most suitable sensor for the activity recognition task. The acceleration data used by the SLS is however, not the raw acceleration sampled by the device, but a derived measure. In particular, the Inference Module uses the *Linear acceleration* of the device, which corresponds to the acceleration force applied to the device, without including the gravitational component: this means that when the device is still, the linear acceleration vector amplitude is zero. The linear acceleration typically is implemented by the Android framework as a virtual sensor, applying a high-pass filter to the raw acceleration data. While sampling data, the *Inference module* calculates a set of features over it in order to infer the current user's activity.

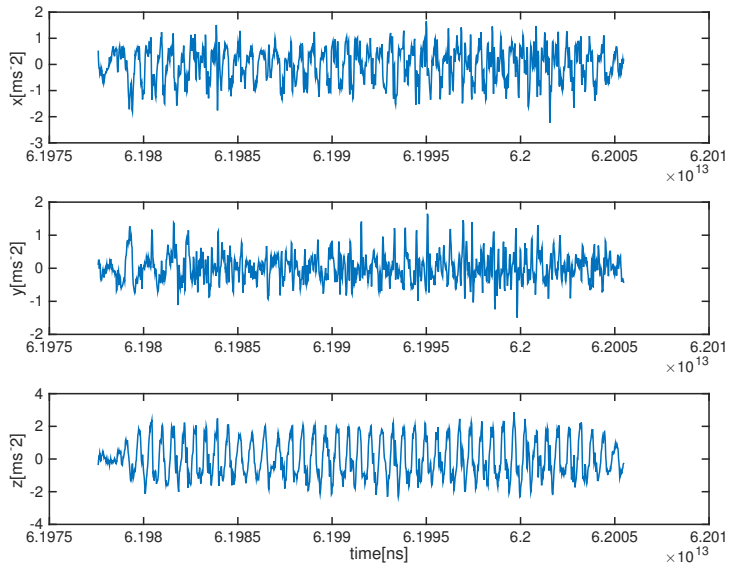


Figure 4.2. Linear acceleration raw data along the x, y, z axis of the device coordinate system, while performing walking activity

4.4.2 Sampling rate

The data collection campaign to build and evaluate the Inference module of the SLS has been performed by means of Android mobile phones. At the beginning of this thesis development, the available versions of the Android operating system allowed to specify the sensor sampling rate, not directly declaring the interval of time between consecutive samples, but selecting one of four different sampling rate classes:

- *DELAY NORMAL*: default sampling rate for android applications;
- *DELAY UI*: rate suitable for updating user interface features;
- *DELAY GAME*: sampling rate suitable for use in controlling games;
- *DELAY FASTEST*: fastest possible sampling rate.

Consecutively, newer versions of Android[1] introduced the possibility to specify the interval between consecutive samples, without relying on a fixed sampling class.

[1]Starting from Android 3.0 Honeycomb (API level 11) available from February 2011

Since both the Android OS and each device documentation do not provide precise information about the exact sensor sampling rates, I collected sensor data from different android phones running different versions of the Android OS and analysed it. A summary of the devices used is reported in table 4.4. In table 4.5 I report the average interval of time between consecutive samples from the accelerometer sensor, for each sampling rate class, and for different devices (listed in table 4.4).

Device num.	**Phone Model**	**Android OS version**
1	Samsung Galaxy S Advance	2.3.6
2	Samsung Google Galaxy Nexus	4.1.1
3	Huawei Ascend P1	4.0.3
4	Samsung Google Nexus S	4.1.2

Table 4.4. Android used devices

Declaring a sampling rate or a sampling class on the Android OS, does not guarantee the specified sampling rate to be actually applied. The Android OS tries to sample data in the best case with the rate specified, but this is not always guaranteed: this means that the data collected has a period between consecutive samples with a standard deviation different from zero. Furthermore, from the data reported in table 4.5 we can notice that the sensor sampling class is not a discriminant for the real sensor sampling rate, in fact it may depend both on the hardware and on the OS version.

	Avg period [sec]			
Sampling rate class	device 1	device 2	device 3	device 4
DELAY NORMAL	0.2	0.2	0.01	0.02
DELAY UI	0.06	0.07	0.01	0.02
DELAY GAME	0.02	0.02	0.01	0.02
DELAY FASTEST	0.01	0.01	0.01	0.02

Table 4.5. Android sampling rate classes

Figure 4.3 shows the energy used by the accelerometer of a Samsung Google Galaxy Nexus running the Android OS 4.1.1 (phone number 2 in table 4.4), while sampling the sensor at different rates. During the energy measurement phase the devices is set to its factory status and have no other services running on it, except the sensor sampling one; the WiFi and Bluetooth interfaces and

GPS localization are off; and the screen is off as well (since it has an high impact on the energy used by the device itself); the device is not provided with any GSM SIM card and there is no Google account registered. The specified setting are important to be able to measure the energy used for the accelerometer sampling, without any significant impact from different processes.

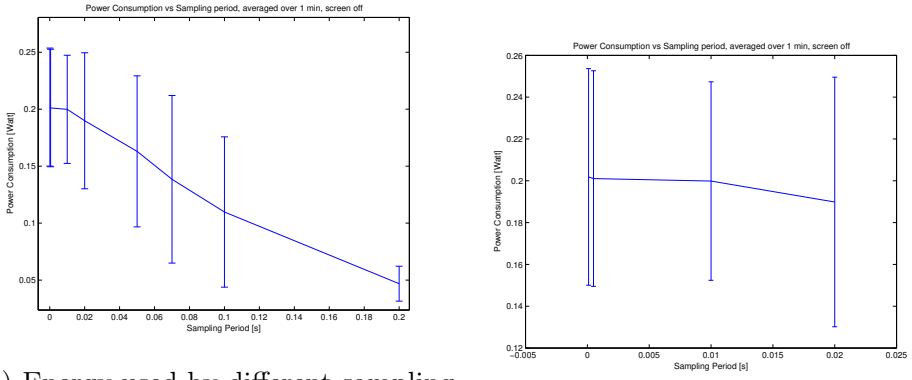
We measured the energy consumption given by different sensor sampling rates in order to optimize the trade off between the energy consumption of the sampling phase and the precision of the activity inference. These energy measurements have been performed by using the ad-hoc circuit presented in section 3.3. Data collected at higher rates provides more information of the device's movement, however it may introduce more noise. Therefore, since a higher sampling rate does not always lead to a higher accuracy and, as shown in figure 4.3 it is more expensive in terms of energy consumption, in my study all the experiments are performed by setting a sampling rate equal to

$$dt_s = 0.01sec \tag{4.1}$$

or, respectively, setting the *DELAY FASTEST* sampling class for the above mentioned device. As we can see in figure 4.3, the power consumption of the accelerometer sampling decreases monotonically with values of dt_s greater than 0,01 sec. For values lower than this threshold (figure 4.3b), the power consumption remains quite stable. This behavior may be caused by a physical limitation of the embedded sensor: while the operating system provides us with updates from the sensor at higher frequencies, the actual sampling from the accelerometer cannot be performed with frequencies higher than the a fixed threshold of the sensor. This motivates the chosen sampling rate dt_s.

4.5 Inference Window Selection

To perform the activity inference, the system collects acceleration raw data and calculates the features over consecutive averaging sliding windows (figure 4.4). Moreover, the systems considers overlapping sliding windows to be able to capture all the activities, ensuring robustness in the prediction. The inference performed is in fact not continuous because it would be expensive in term of computational costs, and at the same time it would provide redundant information. Instead the inference is discrete and performed at every sliding window. The overlapping ensures the capability of the Inference Module to account all the activities, even if they happen across two consecutive inference windows. The



(a) Energy used by different sampling rates

(b) ZOOM for $dt_s = [0.001, 0.02] sec$

Figure 4.3

duration of the inference window affects considerably the inferential study. Obviously the energy necessary for sampling a small time window of data is lower, however the activity inference performed over a small set of data have a lower accuracy. On the other hand, considering a very long time window does not always allow a more accurate inference, in fact it may include more activities and it has an higher cost for unnecessary more expensive sampling.

In order to select the best value for the sliding window duration, I studied empirically the trade-off between its complexity in terms of computational time and the precision of the inferential algorithm. In figure 4.5 I show the empirical evaluation of the *complexity* of the features calculation in terms of computational time, against the *precision* of the inferential study, for different dimensions of the time window. The *complexity*, the blu line with the standard deviation bars in the figure, gives an idea of the impact of different sampling windows duration on the feature calculation computational time. The data has been collected by means of the device number 2 (in table 4.4), running only the feature calculation service. The data over which the features are calculated, for different dimensions of the window, is always the same and is stored locally on the device. As expected, the computational run-time, as visible in the graph, increases monotonically with the increment of the time window duration.

The *precision* has been measured considering a Decision Tree classifier, and calculating a set of features (mean, standard deviation, peak-peak amplitude, energy) described in section 4.6.4 . For this empirical study we used a dataset whose specifications are described in section 4.3.2. The training of the classifier is performed over a training-subset of the available data, corresponding to

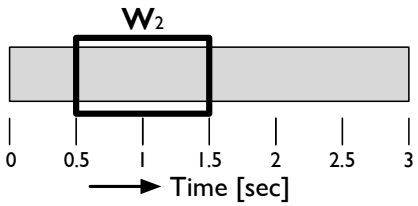


Figure 4.4. Sliding Window

70% of the corresponding sampling windows, selected randomly. The precision depicted in figure 4.5 is calculated over the remaining 30% of the data.

As stated before, the inference is performed by using a sliding window technique, where consecutive windows slide by half the window duration. Considering the example represented in figure 4.4, if the first sampling window is $w_1 = [0, 1)$ and the sliding time is $t_s = 0.5sec$, the second window will be $w_2 = [0.5, 1.5)$, the third $w_3 = [1, 2)$, etc.

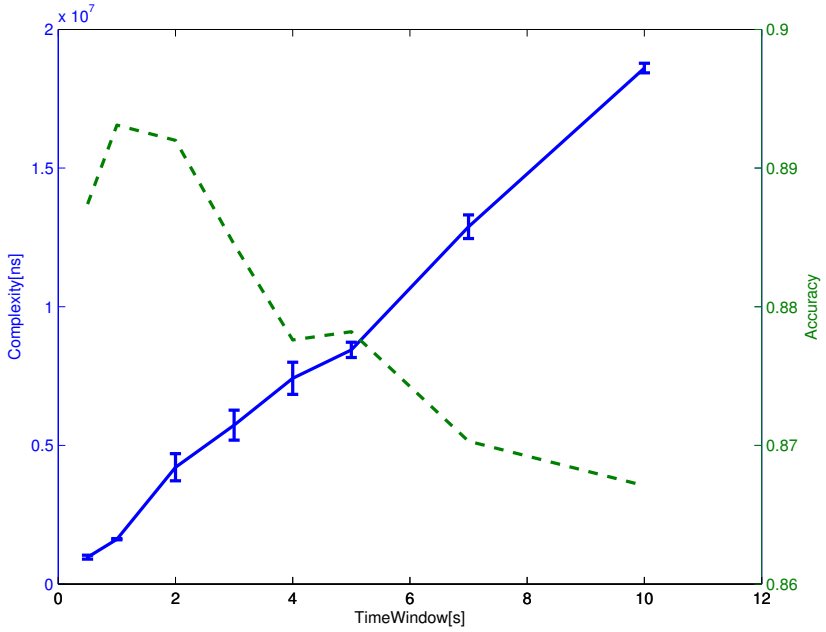


Figure 4.5. Accuracy and Complexity of the features calculation, over different inference window dimensions

From this empirical study, whose results are depicted in figure 4.5, time windows with a duration in the interval [1-5] seconds have a good trade-off be-

tween computational run-time and inferential precision. However the accuracy decreases for time windows larger than 1 second, where the algorithm has to average the sampled values for a quite long interval of time. Considering time windows smaller than this value, the algorithm receives a too small quantity of information, decreasing the accuracy as well. According to these empirical results, the value used by the SLS for the duration of the inference window is:

$$t_w = 1sec \tag{4.2}$$

4.6 Activity Classification

In this section I will give more details about the activities recognized by the SLS, the calculation of the features and the classification algorithm.

4.6.1 Activities recognized

The Activity inference module allows the SLS to change its behavior according to the current user's activity. The main idea consists in stopping the localization tracking when the user is not moving, and changing the location tracking frequency according to the user's activity, whenever in movement. For this reason, the Inference Module recognizes not the specific activity of the user, but its class. The activity categories recognized by the SLS are separated in four different classes which determine the related location tracking frequency class (more details in chapter 7).

The dataset collected during the *final data collection campaign* (described in section 4.3.2) includes a set of 9 different user's activities. In table 4.6 I list the activities collected in the database, and associate to each of them the corresponding class.

4.6.2 Features Extraction

According to the state of the art, there is an extensive list of features that may be considered to identify human's activities, both in time and frequency domains (Su et al. [2014]). The selection of the features is challenging, since there may be situations in which they provide redundant information (Misra and Lim [2011]) at an high price, given by the resources consumption: the challenge consists in the selection of the minimal set of features which enables a reliable inference of the user's activity. An initial set of interesting features mainly used

Activity Recognized	Activity Class
still stand	not moving
walk run	moving by foot
bike	slow vehicle
motorbike car bus train	fast vehicle

Table 4.6. Classes of Activity

in most related works (Kim et al. [2010]; Miluzzo et al. [2008]; Ofstad et al. [2008]; Reddy et al. [2010]) is listed below.

- *Mean*. The mean value of the sampling window, for all the three dimensions of the sensor data vector (x, y, z).
- *Max, Min*. The maximum and minimum values of the sampling window, for the three axes.
- *Standard deviation* and *Variance* of the sampling window.
- *Correlation* calculated between each pair of axes of the raw data.
- *Data Magnitude* $\sqrt{x^2 + y^2 + z^2}$, calculated over the three axis, to tolerate the random orientation of the device (Kim et al. [2010]).
- *Peak-peak* amplitude of the sinusoidal wave and *Variance* of the amplitude.
- *Discrete Fourier Transform* of the raw data.
- *Signal Energy*;
- *Number of Peaks per time-unit*. Frequency of the sinusoidal wave.
- *Peak regularity*. Ratio of the peak amplitude in each coordinate axis divided by the sinusoidal extension in time (time between peak) and its *variance*.
- *Entropy* which helps in discriminating activities with similar energy features.

With an initial study described below in the next section, I demonstrated that only calculating a small set of features over the sensors raw data allows a straightforward inference of the user activity among predefined categories (walking, running, driving). This is a first result I obtained in an initial feasibility study, which was very promising considering the inferential study of the SLS, which is however interested in the classification of four more general categories of user's activities.

Initial feasibility study

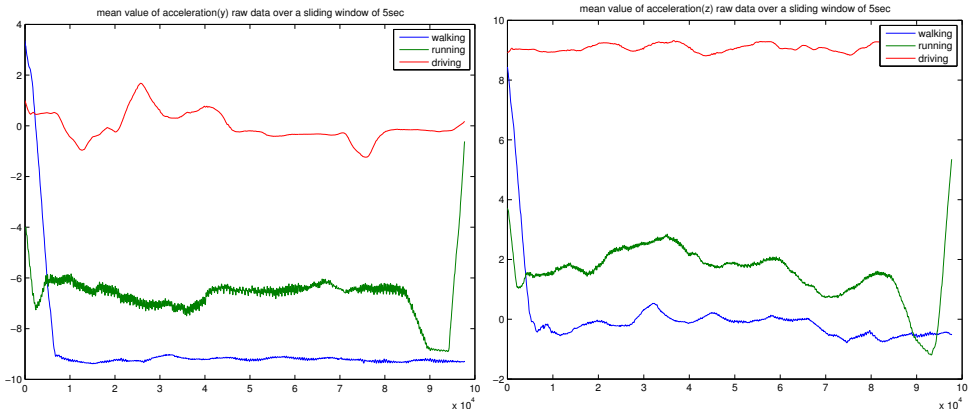
In line with the state on the art (Kim et al. [2010]; Miluzzo et al. [2008]; Ofstad et al. [2008]; Reddy et al. [2010]), I performed some experiments for studying how to differentiate between different human activities. A real experiment has been run with users collecting sensors data with two different devices (HTC NexusOne running Android 2.3.6 and a Samsung GT I9100 running Android 2.3.3), while performing three different activities (walking, running, driving). The data was collected sampling three different embedded sensors: accelerometer, magnetic field and orientation sensor. The average sampling rate (the fastest possible according to the devices themselves and the versions of the operating system installed onto the devices) is equal to 25Hz (sampling period of 40 milliseconds). While collecting data the phones have been positioned respectively into the left and right pockets of the user's jacket, when the user was walking and running; instead, while driving, the user positioned the phones into the car (not attached to the user's body, then not influenced by the relative movements of the user inside the car).

Some mining over the sampled data has been performed off-line to understand if it is possible to recognize the user's activity. A set of features have been calculated over the data for sliding windows of 5 seconds. I show here some of the most significant features, which allow a clear differentiation of the user's activity among the three predefined categories: walking, running, driving. In each figure presented below, the blue line represents the data collected while walking, the green data has been collected while running, and the red data while driving.

Figures 4.6a and 4.6b represent respectively the mean values calculated over the y and z axis of the accelerometer raw data. Both of them are represented in ms^{-2}.

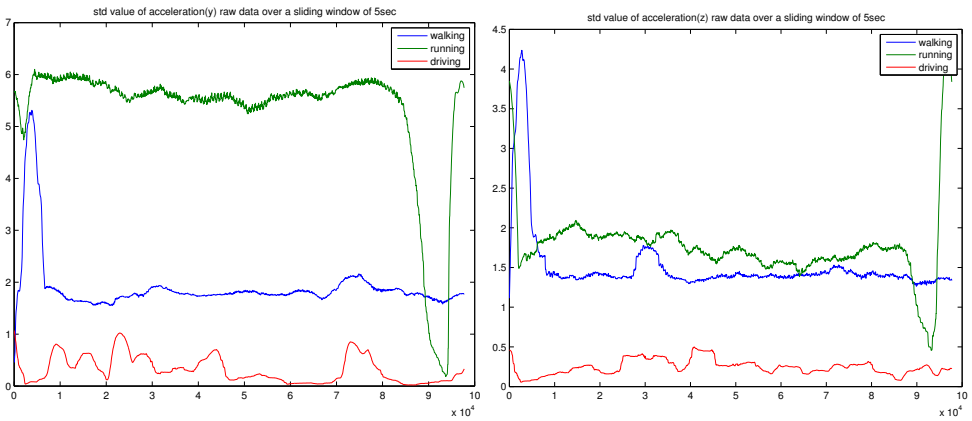
Figures 4.7a and 4.7b represent respectively the standard deviation value calculated over the y and z axis of the accelerometer raw data.

Finally figure 4.8 represents the peak-peak amplitude of the accelerometer



(a) y-axis (b) z-axis

Figure 4.6. Acceleration: mean feature



(a) y-axis (b) z-axis

Figure 4.7. Acceleration: standard deviation feature

data on the y axis. This value consists basically on the distance between consecutive local maximum and minimum values.

Figures 4.9a and 4.9b represent respectively the standard deviation and the average peak-peak amplitude calculated over the magnitude of the acceleration data. This feature has been considered to tolerate the random position of the mobile device with respect to the user body. The magnitude of the acceleration data is defined with the formula $\sqrt{x^2 + y^2 + z^2}$, where x, y and z are the values of the raw accelerometer data, over the three axis.

Also some features calculated over the orientation sensor data allow a straightforward differentiation of the user activity among the three predefined classes.

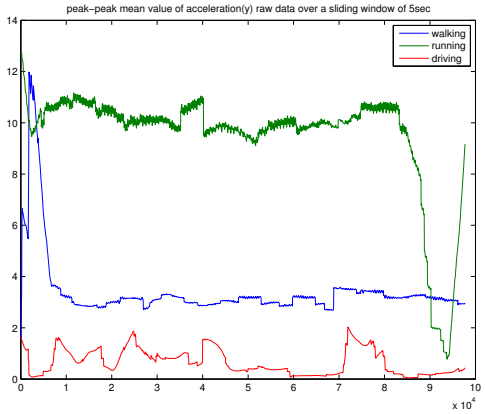
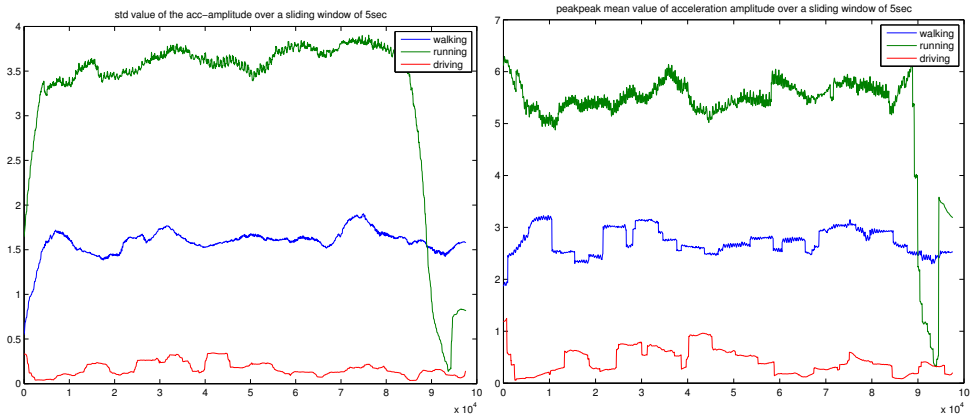


Figure 4.8. Acceleration: peak-peak amplitude feature (y-axis)

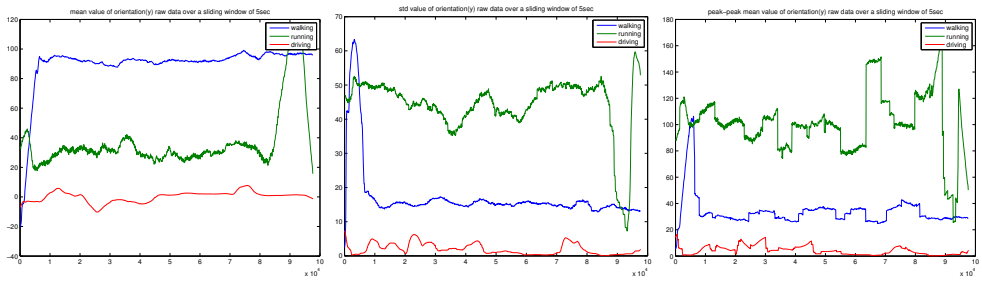


(a) Standard deviation feature (b) Peak-peak amplitude feature

Figure 4.9. Acceleration magnitude

Figures 4.10a, 4.10b and 4.10c represent respectively the mean, the standard deviation and the average peak-peak amplitude value for the orientation raw data, over the y-axis (pitch in degrees).

Analyzing the figures presented in this section, I can conclude that a straightforward inference of the user activity among predefined categories is possible by simply calculating a small set of features over the sensors' data. However, this is only a subset of the significant features calculated in this feasibility study. From this initial analysis I selected the accelerometer as the most suitable sensor to perform activity inference. And also a final set of features has been selected for the *Inference Module*. A more detailed list of features is presented in the next section.



(a) Mean (b) Standard deviation (c) Peak-peak amplitude

Figure 4.10. Orientation sensor (y-axis)

4.6.3 Features Selection

The activity inference analysis is performed entirely on the mobile device, without any support from a beck-end server. This solution brings to an important advantage: the device is independent from any network connection, hence the activity inference procedure works also when the telephone is not connected to the internet. The SLS solution has therefore a challenging requirement: the inference methodology has to be as lightweight as possible, to be run on a device with a limited battery capacity. As stated above, the activity inference module is included in the SLS in order to reduce the excessive energy consumption of the expensive GPS localization. This motivation force us to keep the inferential study as lightweight as possible.

In order to be able to extract a set of features which allows the system to distinguish between different users' activities, the system analyses only the linear accelerometer data in the three-dimensional space. It considers each accelerometer axis separately ($\bar{a} =< a_x, a_y, a_z >$). Furthermore, in order to be able to tolerate the different positions of the sampling phone related to the user's body, it considers two additional calculated measures: the *magnitude* of the three-dimensional accelerometer vector,

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{4.3}$$

and the *sum* of the scalar components of the acceleration vector, over the three axis.

$$s = x + y + z \tag{4.4}$$

In fact, by using these measures it considers only the intensity of the acceleration without its vectorial part.

Starting from the set of input data presented above:

$$\left\{a_x, a_y, a_z, |a|, s\right\} \tag{4.5}$$

the system calculates a set of 30 significant features to perform the activity inference. Here below I give a detailed explanation of each feature. These features are calculated for each input data listed above, in equation 4.5

Mean With this feature, the Inference Module calculates the average value of the linear acceleration raw data for each inference window. Considering a signal f measured for a limited time window consisting of n samples, the mean value of the signal is given by the following formula.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} f_i \tag{4.6}$$

In figure 4.11 we can see an example of the mean feature calculated over the

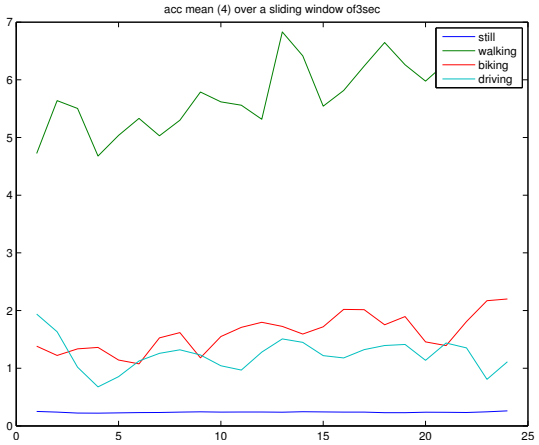


Figure 4.11. Mean of the amplitude on the linear acceleration

forth dimension of the input data (magnitude in equation 4.5), considering an inference window of 3 seconds. As visible from the figure, the *walking* and *still* activities are clearly distinguishable. *Biking* and *driving* are distinguishable from the other two activities. But it is not possible to differentiate between them by using only this feature.

Standard Deviation This is the measure of the quantity of signal variation according to its mean value. Considering n consecutive samples of a signal f, measured for a limited amount of time, the standard deviation is given by the following formula,

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (f_i - \mu)^2} \tag{4.7}$$

where μ is its mean value. In figure 4.12 we can see an example of the standard

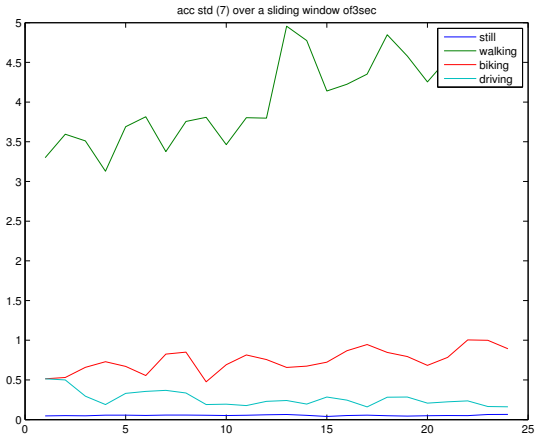


Figure 4.12. Standard deviation of the z-component on the linear acceleration

deviation feature calculated over the third dimension of the input data (equation 4.5), considering an inference window of 3 seconds. Differently from the mean feature, with the standard deviation it is possible to clearly distinguish all the four activities depicted in the graph.

Signal-to-Noise Ratio This feature compares the intensity of the actual signal to its noise. Since the Inference Module is calculating this feature over a limited time interval of the linear acceleration raw data signal, I assume the signal mean data to be constant over this window, hence it considers the standard deviation of the signal instead of its noise. This measure is given by the ratio of the signal mean value, to its standard deviation.

$$SNR = \frac{\mu}{\sigma} \tag{4.8}$$

In figure 4.13 we can see an example of the signal to noise feature calculated over the fifth dimension of the input data (equation 4.5), which correspond to the sum of the scalar components of the linear acceleration, considering an inference window of 3 seconds. With the signal to noise value it is possible to distinguish in a very straightforward way the driving and the still activity, from the other two depicted in the graph.

Peak-Peak Amplitude This feature gives an idea of the signal maximum-minimum peaks amplitude. To retrieve this feature, I calculate the mean value of the signal

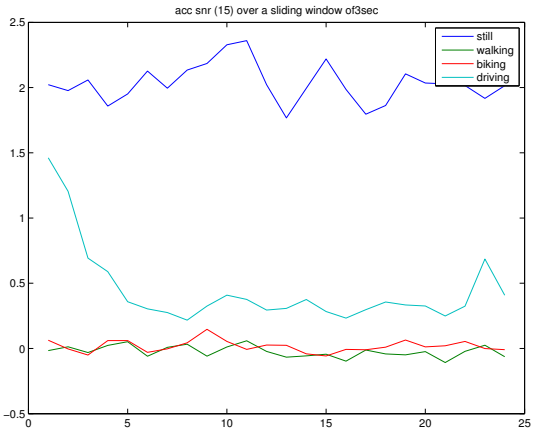


Figure 4.13. Signal to noise value of the sum of the scalar components of the linear acceleration

translated, such that the minimum picks correspond to zero.

$$P - PA = \frac{1}{n}\sum_{i=1}^{n}\left(f\left(t_i\right) - f\left(t_k\right)\right) \tag{4.9}$$

Where

$$\exists t_k : f\left(t_k\right) \leqslant f\left(t_j\right), \forall j \neq k \tag{4.10}$$

In figure 4.14 we can see an example of the peak-peak feature calculated over

Figure 4.14. Peak-peak feature calculated on the amplitude of the linear acceleration

the amplitude of the linear acceleration, considering an inference window of 3

seconds. Also with this feature it is possible to distinguish the walking and still activities, from the other activities depicted in the graph.

Energy This feature measures the quantity of movement, proportional to the energy necessary for the movement of the device. Considering a signal f measured for a limited time window consisting of n samples, the energy value of the signal is given by the following formula.

$$E = \frac{1}{n}\sum_{i=1}^{n} f_i^2 \tag{4.11}$$

In figure 4.15 we can see an example of the energy feature calculated on the

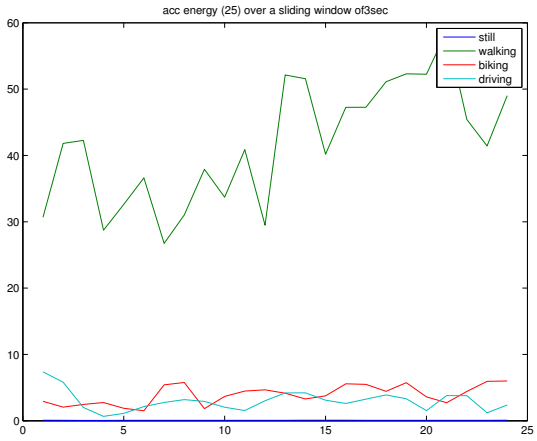


Figure 4.15. Energy calculated on the sum of the scalar components of the linear acceleration

sum of the scalar components of the linear acceleration, considering an inference window of 3 seconds. Also this feature allows the distinction of the walking and still activities from the other ones.

Derivative This feature measures how the raw data signal changes with time: the Inference Module calculates the punctual derivative of each single raw sensor sample, and average its absolute value over the inference window. Considering a signal f limited in time, and consisting of n samples, the derivative value of the signal is given by the following formula.

$$d = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{df_i}{dt_i} \right| \tag{4.12}$$

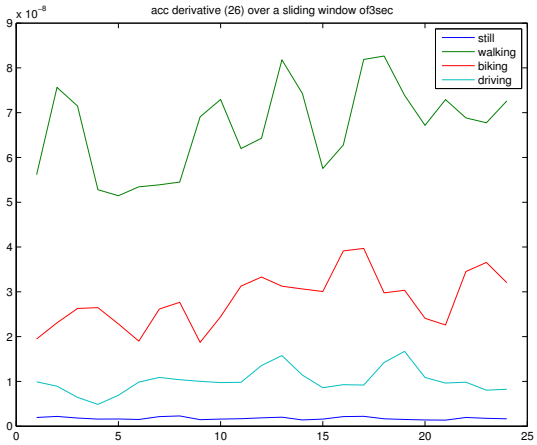


Figure 4.16. Derivative of the x-component of the linear acceleration

In figure 4.16 we can see an example of the derivative feature calculated on the first dimension of the input data (equation 4.5), which corresponds to the x-component of the linear acceleration, considering an inference window of 3 seconds. With this feature it is possible to easily distinguish all the four activities depicted in the picture.

4.6.4 On-Mobile features calculation

Many of the works from the state of the art perform the feature calculation Server-side. Some other works propose the calculation in two stages, one on the mobile and the second one on a back-end server, in order to reduce the amount of data transmitted over the internet. However, this solution may introduce privacy issues, since information about the current activity of the user is sent outside of the mobile device. On the other hand, the calculation of the features is an expensive task in terms of resources usage, and for some complex features it may be necessary to offload the computation on a server. In the SLS the calculation of the features is performed entirely on-line to avoid privacy issues, and the features calculation is lightweight enough to have a limited impact on the resources usage.

While collecting raw linear acceleration data, the SLS calculates the selected features (described above in section 4.6.3) without storing locally the raw data itself, in fact it does not need to read the data more than once. The Inference module temporarily stores locally only the features calculated, which are deleted as soon as the SLS performs the inference of the user's activity. The features calculation implementation is lightweight enough to run on the mobile and does

not have a strong impact on the resources usage (for both memory and CPU); there are no features in the frequency domain and the calculation is performed in an incremental way on the stream of sampled data.

More details about the incremental feature implementation is reported in Appendix A.

4.6.5 Classification Algorithm

The Inference Module implements its inferential study adopting a Decision Tree classifier, in fact according to the state of the art (section 4.2.2), it is the most suitable candidate to perform activity inference on the resources constrained target device.

The training of the Decision Tree is performed off-line. The dataset of labeled data (accelerometer data while performing known activities) used for training the algorithm is described in section 4.3.2. The two resulting decision trees have respectively: 255 levels, distinguishing between 4 different activities and 430 levels, distinguishing between 9 different activities (as described in section 4.6.1).

The Inference Module reads the stream of raw data from the linear acceleration sensor periodically. The period depends on the current inferred user's activity. At each period, it reads a total amount of 3 seconds of data, which corresponds to 5 consecutive overlapping inference windows. For each inference window, the Inference Module calculates the 30 features and submits them to the decision tree, to infer the most probable activity of the time window. Consecutively a majority study is performed for the 5 inferred activities, and its result is returned to the SLS as the actual activity of the user. This majority study is performed to tolerate very fast movements of the user, which do not reflect her/his current activity.

4.7 Experiments and Evaluation

The evaluation of the Inference Module has been performed offline, by using a dataset of labeled data (described in section 4.3.2). The precision of the inference has been calculated using the same dataset for training and testing the algorithm. The training phase has been performed selecting randomly 70% of the data (the random selection has been done on the inference windows), while the remaining 30% has been used for testing.

The evaluation has been done in two steps. First of all I evaluated the algorithm, while it was inferring all the 9 activities included in the dataset. The final results are shown below in figure 4.17 and table 4.7. More specifically, figure 4.17 represents the confusion matrix of the inference, where each element *(i,j)* is equal to the percentage of guessing the activity *j*, given the real activity is *i* (it shows, in the diagonal, the probably to predict the correct user's activity, per each different activity). Since the data-set which I am using is unbalanced (the amount of data collected per each class is different), I report in table 4.7 the performances of the inference algorithm in terms of *Precision, Recall, Accuracy*, calculated both separately per each activity, and globally for all of them together. As visible from the table, almost all the activities have a quite high inference recall value, except for *stand* and *bus*. For the *stand* activity, the main false negative (FN) errors are performed while predicting "*walk, bus, train*" instead. This is basically because of the noise in the training data: the collected data has been labeled as "train", while standing on the train; in the same way, the collected data has been labeled as "bus", while standing on the bus; and it has been labeled as "walk", for short standing intervals during the walk activity. For the *bus* activity, the main false negative errors are performed while predicting "*car*" instead. In fact, it is not easy to differentiate between a car and bus by using uniquely the accelerometer.

Activity	Precision	Recall	Accuracy
still	0.9664	0.9712	0.9827
stand	0.6952	0.6250	0.9864
walk	0.9352	0.9399	0.9723
run	1.0000	0.9877	0.9999
bike	0.9036	0.8827	0.9939
motorbike	0.8789	0.8635	0.9970
bus	0.6802	0.6726	0.9498
car	0.8393	0.8465	0.9327
train	0.9018	0.8996	0.9713
Global	0.8930	0.8930	0.9762

Table 4.7. Evaluation of the inference algorithm: 9 activities

In a second step, I evaluated the algorithm, while it was learning and inferring the 4 classes of activities (listed in section 4.6.1). In this case, all the classes of activities have a high inference recall value. This is mainly explained by the fact that each class includes a group of similar activities, which corresponds to

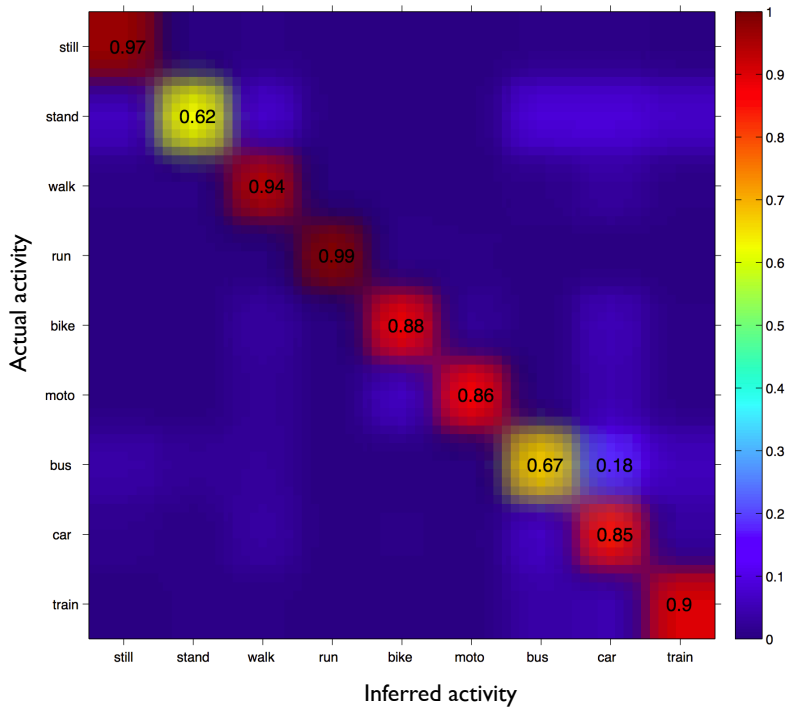


Figure 4.17. Confusion Matrix global for all 9 activities

the main cause of false negative predictions in the previous evaluation. In fact, in the evaluation with 9 activities, most of the errors were performed while inferring activities in the same class.

Activity	Precision	Recall	Accuracy
Not moving	0.9616	0.9529	0.9744
Moving by foot	0.9356	0.9375	0.9716
Slow vehicle	0.8881	0.8798	0.9931
Fast vehicle	0.9463	0.9518	0.9544
Global	0.9468	0.9468	0.9734

Table 4.8. Evaluation of the inference algorithm: 4 classes of activity

The global precision calculated for the activity inference with 9 activities is 89.3%, while the global precision considering the 4 classes of activities is 94.7%. Since, as stated before, the dataset is not balanced in the amount of data per each activity or class of activity, I also calculated the global *balanced accuracy* (equation 4.13) for both cases, and reported it in table 4.9. In the formula

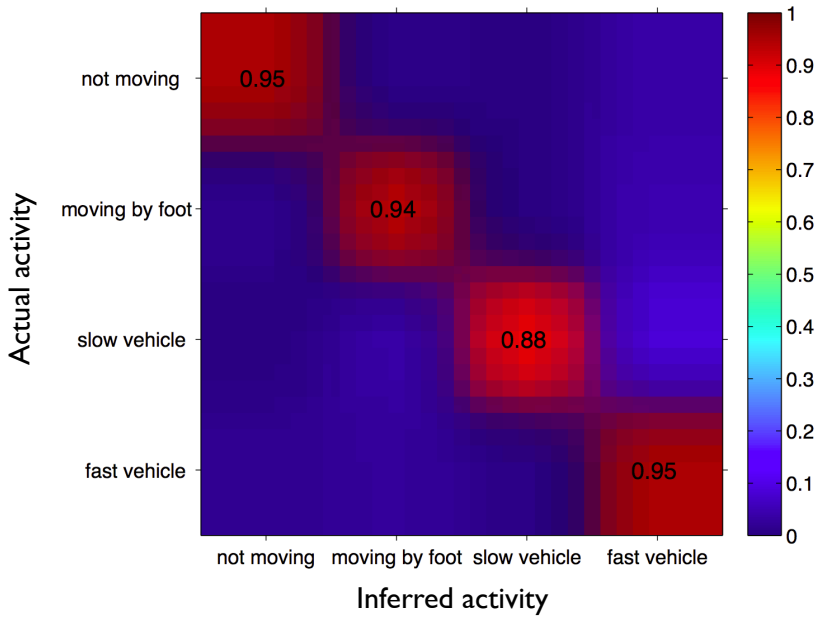


Figure 4.18. Confusion Matrix global, four classes of activities

4.13, TP stands for *True Positives* and TN for *True Negatives*, and FP and FN correspond respectively to *False Positives* and *False Negatives*.

$$BalancedAccuracy = \frac{0.5 * TP}{TP + FN} + \frac{0.5 * TN}{TN + FP} \tag{4.13}$$

9 Activities	**4 Classes of Activities**
0.9406	0.9645

Table 4.9. Balanced Accuracy

In chapter 7 I will analyse the activity inference including the majority study, and evaluate it in terms of user prediction and impact on the SLS overall battery usage.

4.8 Conclusion

In this chapter I described the algorithm implemented by the Inference Module, which reads the data sampled by a mobile device embedded sensor, and infers the activity of the user.

The algorithm uses the *linear acceleration* data, sampled with a frequency of 100Hz, and builds an input vector over it. A data input vector is associated to each sampling window of 1 second, and contains 5 values: the x, y and z scalar components of the acceleration, the acceleration magnitude, and the sum of the acceleration scalar components. The algorithm calculates a set of 30 features over this input vector (mean, standard deviation, signal to noise ratio, peak-peak amplitude, energy and derivative). These features are then provided to a Decision Tree algorithm which is trained to identify 9 different activities: still, stand, walk, run, bike, motorbike, car, bus and train. A second version of this tree is trained to identify 4 different groups among the activities specified before: not moving, moving by foot, slow vehicle, fast vehicle.

The algorithm has been evaluated and the resulting balanced accuracy resulted to be 94% while inferring 9 different activities, and 96.4% while inferring the 4 different classes of activities. The accuracy of the algorithm are comparable to other solutions presented in the related works. Moreover the inference algorithm does not need the support of a backend server for computing the feature and for running the algorithm itself. Its performances have been also calculated according to the user precision, by means of a final experiment presented at the end of this thesis, and its accuracy outperforms already existing solutions (more details are reported in chapter 7).

Chapter 5

Learning User Mobility

The mobility model implemented by the SLS is centered onto the concept of Point of Interest (PoI). A PoI represents a relevant location where the user spends a considerable amount of time, like home, place of work, a mall. The SLS while collecting data, analyzes it to understand and to model the movements of the user among her/his PoIs.

In this chapter I explain the location learning process performed by the SLS. In section 5.3 I give a definition of PoI and an explanation of the methodology applied to identified it. Subsequently we present two consecutive phases of my work: in the first phase (section 5.3.1) the SLS performs its learning procedure analysing only the collected location data; in the second phase (section 5.4.2), the learning procedure includes also the time information.The results related to these two learning phases are presented in the next chapter, where we describe the prediction procedures based on the learned models. In section 5.5 and 5.6 I analyse the identified PoIs and divide them in different relevance classes, studying the movements of the user and identifying the features which characterize the human mobility among the selected classes. The State of the Art about human mobility modeling and prediction will be presented in the next chapter.

5.1 Introduction

In this chapter I go into the details of the *Learning Module* (as presented above in figure 2.1) of the Smart Localization System. To be able to make predictions about the user's movements, the SLS system needs to collect data about visited locations, and to perform some mining over it. To this aim, the *Learning Manager* (introduced in section 2.2.1) stores locally the localization data, perform some mining procedure over it and calculates the parameters which will personalize

the user prediction model. Thus, while running, by collecting data, the Learning Manager incrementally learns about the user's habits and contexts, and adapt the model to them.

This personalization of the model allows the system to perform location predictions in order to reduce the localization triggering frequency and to avoid unnecessary frequent location measurements. In fact, several habits and contexts of the user are known, hence worthless battery consumption. Clearly, during the *Start Up phase* (from the first run of the system, until the model has been tuned) the localization is performed only using the nodes tracking technologies, as the SLS still needs to personalize the mobility model to the carrying user. Hence, in terms of battery consumption, initially the system does not have a significant gain with respect to already known techniques. The real advantage is visible at the end of the bootstrap phase, after which the system starts decreasing the frequency of direct location tracking, alternating them with predictions.

5.2 State of the Art

Recent faster transportation methods have made people mobility very common for both businesses and daily life. In addition, advances in communications technology, data analysis and smart infrastructure are enabling to streamline the transportation strategies, simplifying connections and shortening the commuting times. These two aspects together resulted in a high mobility degree for many people, both for their business or as a lifestyle.

However, despite the higher mobility degree, I argue in this chapter that visited locations may be classified in three main categories. One of the three classes includes the places that a person visit more frequently and thus were a person can be caught with higher probability. However these locations are limited in number.

Watts and Strogatz [1998] were modeling the famous *six-degree* property of Milgram, giving birth to the small world phenomena era: the average path length for social networks of people was established to be six. Dunbar [1992] measured the correlation between neocortical volume and typical social group size in a wide range of primates and human communities. The author showed that, because of the limit imposed by neocortical processing capacity, people can have stable interpersonal relationships with only a limited number of individuals. Thus, the Dunbar's number is the measure of the humans' social network size, and is between 100 and 200 individuals (Dunbar [1998]). In addition to the neuro-scientific limits, we can also individuate some physical constraints,

as our time and interests are finite and therefore we cannot have (strict) social interaction with the whole world.

Both results concur to give a surprising view of how our social world is *small* (connected with small number of hops) and cannot go over certain limits (we have limited numbers of strong connections). I argue that, similarly, *our physical world is small* and cannot go over certain limits (Gonzalez et al. [2008]): we can commute, with small number of hops, between very far places, but the number of points that we frequently reach is limited. Intuitively, the fact that we can commute everywhere, with small number of hops is clear, but the fact that our high interest locations are few is not that evident, especially if we consider the evolution of our society toward a very dynamic lifestyle.

Thus, regardless the increased attractiveness of a place or the possibility to reach places more quickly, people will keep on moving around their most relevant points for most of their time. This indicates that clearly those points are the ones that better represent and characterize our life. Hence human mobility can be modeled in terms of movements among these locations.

Several related works focused on modeling the human mobility, where people are considered as a community and their mobility is driven by social factors (Zignani [2012]; Musolesi and Mascolo [2006]). Calabrese et al. [2010] modeled human mobility by using a probabilistic approach, and in particular by analyzing the person and collective past behavior. Also Gambs et al. [2012] build a user mobility model based on the observations of the user's past mobility behavior. Their goal is the location prediction. For this purpose they implement a Mobility Markov Chain (MMC) algorithm in order to incorporate the n previous visited locations for next location prediction (where $n = 2$).

In this chapter I present two probabilistic approaches to build an individual mobility model, characterized by only the user own data, hence without considering social factors. I analyze the past movements of the user among the most relevant points of interest, trying model his mobility behavior given the regularity of these movements.

5.3 Point of Interest

The user's mobility model implemented by the SLS is mainly centered into the concept of PoI. The localization data, while being collected by the Smart Localization component (figure 2.2)of the SLS, is also stored locally to retrieve, with time, information about user's specific relevant PoIs (e.g., home, work, gym, club).

Definition 1. *A PoI is a location area relevant for a user (i.e.,where the user spends a relevant amount of time). Its range in space may be in the order of few meters, if the location refers to a very specific place (i.e., work office) or larger (e.g., a mall, a stadium), depending on how the user visits the location.*

A clustering algorithm runs periodically over the location points visited by the user and selects centroids among them, according to the frequency of the visits and their duration. The results of the clustering procedure is stored into a User Prediction Graph (5.4.1). The SLS continuously retrieves data and periodically refines the identified clusters by re-executing the clustering algorithm, introducing the new collected data. This is done to tolerate and take into account possible changes in the users habits and frequently visited locations.

In the next sections I perform the analysis of the location data and the identification of the clusters offline, by using the two datasets presented in section 2.3.

5.3.1 PoIs Identification

GPS datasets present many difficulties as concerns the identification of the PoIs. In fact they do not give any information about the interests expressed by the user, hence the only meaningful property for the PoI identification is the user's still activity. Assuming a constant sampling rate for the GPS dataset, the pause periods and the places visited by users translate in an higher concentration of recorded points. This way the PoIs extraction corresponds to the unsupervised task of density-based clustering.

DBSCAN (Density Based) (Ester et al. [1996]) clustering algorithm is a very good candidate for managing location data, but it is sensitive to ε and $minPoint$ parameters (which represent respectively, the distance radius for defining the neighborhood of a location point, and the minimum number of points required inside the neighborhood to define it as a cluster), and it can only handle datasets which fit in memory. To overcome the parameters problem in my work I decided to dynamically set the values of both ε and $minPoint$ (more details in section 6.2.1). To handle memory problems instead, a good candidate may be the DJ-Cluster [Zhou et al., 2004], Density-and-Join based algorithm which is a variation of DBSCAN and focuses on performances issues: in particular, it requires at most a single scan of the data. Both the algorithms work on the same basis, hence we use the DBSCAN for offline experiments, and DJ-Cluser for eventual realtime analysis of the data performed only with the mobile device. Furthermore, for both the algorithms we apply the Euclidean Distance Formula for cal-

culating distances between locations. When working on a spherical surface, the great-circle distance between two GPS points is given by the *haversine formula*. During the analysis described in this chapter the calculated distances involved only nearby locations, hence I apply the *Euclidean distance* formula, which gives a good approximation of the real great-circle distance and especially is computationally less expensive for our platform.

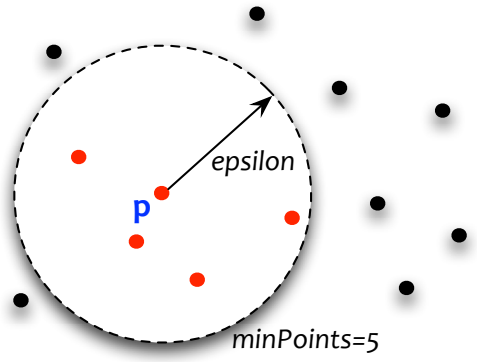


Figure 5.1. DBSCAN neighborhood

The Density based clustering algorithm. For each sampled location point (i.e., a raw trace data including time-stamp, latitude and longitude of the user) the clustering algorithm calculates its *neighborhood* (Equation 5.1); if the point has no neighbors it is labeled as NOISE, otherwise the neighboring points are created as a new cluster, or joined with an existing one if any neighbor belongs to an existing cluster. The neighbourhood N of a point p includes all points within distance ε from p, and it is considered valid if there are at least m of such points (minimum number of points).

$$N(p) = \{q \in S | dist(p,q) \leq \varepsilon\} \tag{5.1}$$

Where S is the set of all points, q is any point in the samples, ε is the radius of a circle around p that defines the density, and it holds:

$$\left|N(p)\right| \geq m \tag{5.2}$$

The cardinality of $N(p)$ is greater than m in order to be able to consider $N(p)$ a cluster.

5.4 User Mobility Learning Algorithm

In this section I present the evolution of the implemented user mobility model. In a first version, the model is a location based algorithm, learning the mobility habits of the user referring exclusively to her/his geographical displacements (section 5.4.1). A second version of the algorithm (which is the final one, implemented by the SLS) is presented in section 5.4.2 and takes into account not only the location but also the timing feature.

5.4.1 Location based Learning Algorithm

The goal of the SLS is to learn relevant information about the user, to optimize the battery usage when she/he visits very well known places, and to reserve battery resources for more critical situations, when the user visits unknown locations. To achieve this goal, by collecting data, the SLS learns about the user habits and constructs a *directional graph*: the User Prediction Graph (UPG).

The User Prediction Graph

An example of UPG is depicted in figure 6.4, where each vertex represents a cluster (a Points of Interest), and the arcs are the connections between consecutively visited relevant locations. Both vertexes and arcs are characterized by weights.

Each node (cluster) i is associated to a value which represents its **relevance-weight**: that is the measure of how relevant is a location to the user (more details in section 5.5.1). The formula to calculate this value is reported in equation 5.3 and corresponds to the ratio between the number of samples collected which reside into a cluster, divided by the total number of samples. The PoIs which are more frequently visited and for longer periods of time, have high relevance values.

$$\mathbb{P}_i = \frac{num\ of\ samples\ into\ cluster\ i\ during\ the\ clustering\ period}{total\ num\ of\ samples\ during\ the\ clustering\ period} \tag{5.3}$$

The value associated to each directed arc (**the arc weight**) of the graph, is the probability to move from the source vertex (or PoI) i to the connected one j. Its value $\mathbb{P}_{ij}$ corresponds to the ratio of the number of moves from node i to node j over the total number of moves from node i.

$$\mathbb{P}_{ij} = \frac{num\ of\ moves\ from\ i\ to\ j}{total\ num\ of\ moves\ from\ i} \tag{5.4}$$

In this first learning algorithm, both the arcs and nodes weights depend only on the current location feature (i.e., the arc probability depends on the current location of the user in the graph, that means on the current node); in a second phase, the learning algorithm has been improved introducing the timing feature: the arc probability changed accordingly, depending instead on the current location of the user in the graph and also on the current time-stamp (section 5.4.2).

Each vertex in the UPG includes a set of information: latitude and longitude of the PoI's centroid and its extension in space, average number of visits per day, average duration of the visit. Also each arc in the graph includes information, such as average duration of the movements through the arc and number of arc crossings. All those information are continuously updated, while collecting user's data.

The SLS continuously retrieves data, periodically performs a clustering procedure over it and refines the set of identified clusters. After identifying the set of PoIs for the window of time over which the clustering procedure has been run, a *Temporary directed Prediction Graph (TPG)* is built by the SLS. While the UPG includes the complete user's movements history since the beginning of the learning procedure, this temporary graph only stores data about a limited clustering time window. After each clustering procedure, the last retrieved TPG is merged into the general UPG in order to keep it updated.

Updating the UPG

Updating the UPG consists in including the last period learned data, as defined in the TPG, by:

- adding the new PoIs, which correspond to unknown locations, visited by the user for the first time during the last processed window of time;
- updating PoIs already known and which have been also visited in the last clustering period, by modifying their weight, centroid location and space extension;
- updating the weights on already existing arcs or integrating new arcs, when new connections between PoIs are identified.

The correspondence between PoIs is measured in terms of distance between their centroids, using also in this case the Euclidean Distance formula.

Definition 2. *Two PoIs correspond to the same location area, if their distance does not exceed the sum of their respective standard deviation values.*

More formally:

$$PoI_A \approx PoI_B \Leftrightarrow$$

$$dist\left\langle PoI_A, PoI_B \right\rangle \leq \left(std_{PoI_A} + std_{PoI_B}\right) \tag{5.5}$$

The *standard deviation* of a PoI is the average value of its *radius*: the deviation of the distance from each location point included into the cluster, to its centroid. The value of the standard deviation is calculated incrementally, while the location points are included into the cluster, by applying the formula:

$$\sigma = \sqrt{\frac{n\left(\sum_i x_i^2\right) - \left(\sum_i x_i\right)^2}{n(n-1)}} \tag{5.6}$$

where n is the total number of location points belonging to the cluster, and x_i is the distance of PoI_i to the cluster's centroid.

As reported in equation (5.4), the weight associated to each arc is the probability to move from the current cluster i to the connected one j. The absence of an arc between two clusters means that the user never visited those two clusters consecutively. When the results of the last temporary graph are merged onto the UPG, all the arcs' values are updated according to a factor α, which smooths the weights of the old data with respect to the newly integrated one.

$$\mathbb{P}\left(moving\ from\ S\ to\ D_k\right) = (1-\alpha)\left(\frac{w<S,D_k>}{\sum_i w<S,D_i>}\right) +$$

$$\alpha\mathbb{P}_{old}\left(moving\ from\ S\ to\ D_k\right) \tag{5.7}$$

Assigning a larger weight to the history (past data) with respect to the new data, leads the learning algorithm to be more impacted by the history of the user. However, the computation of the predictions does not have to be heavily affected by very old activities of the user. While performing predictions by means of this algorithm, error rate is expected to increase in two cases: considering the graph update, when a very large weight is assigned to the history, and oppositely also when the same weight is given to the history and to the current data. The best value of alpha in general is in between these two opposite situations. However, according to the goal of the prediction algorithm, the model is needed to predict the next user location when the user visits some PoIs with a certain regularity: that means that not all the known places are interesting for the model, but among them, only the ones which are visited frequently by the user.

The value associated to each node i represents its relevance-weight, different from equation 5.3, which was referred to the temporal prediction graph. For the

UPG this weight is the relevance of the node for the total duration of the learning period (e.g., a location visited only once but for a long duration, may have an high weight in the corresponding temporary graph, but a low relevance in the UPG). The SLS maintains information about the relevance weight of each node in order to keep the UPG dimension limited, pruning the nodes of the UPG when they are old and with a very small associated relevance value (this concept will we recalled in chapter 7, where I will present and evaluate the complete SLS solution).

5.4.2 Location and Time based Learning Algorithm

The location data alone gives the SLS great but limited information about the user's habits. Adding to this data also timing information, helps the system learning more about the user, and gives the possibility to make better mobility predictions.

Also the *location and time based learning algorithm* identifies the PoIs using a Density Based clustering algorithm (as described in section 5.3.1) over the location data, applying the Euclidean Distance formula for calculating distances between point. This clustering algorithm is applied only to the spacial domain and results in a set of clusters, which are *potentially* locations relevant to the user. However, each cluster embeds information related to the time domain[1] that is used in a consecutive analysis. Among the identified PoIs it may be possible to have small clusters (i.e., low density of samples) which correspond to locations frequently visited by the user, but never for a considerable interval of time. For example, if the user always drive on the same road from home to work and backward, and she/he frequently incurs in a point of the road characterized by traffic jam, that point may become a cluster[2]. To filter out those PoIs which are not relevant for the system prediction purposes, we apply a *Temporal Post-Filter* to the identified set of clusters.

Definition 3. *A cluster is considered a relevant PoI if and only if it has been visited by the user for an interval of time equal or greater than a Temporal Threshold.*

The value of this threshold has been evaluated empirically (details in section 6.3.1).

After the filtering procedure, the data (pruned by noise samples identified by the clustering algorithm, and by samples associated to not relevant PoIs) is read

[1] Each point in a cluster is associated with a timestamp.

[2] The system does not assign any semantic meaning to locations, hence a PoI may be situated wherever a user spends a relevant amount of time.

once again in the same order in which it has been sampled (temporal order) and a set of histograms are built over it. The data histograms represent, per user and per day, which clusters have been visited and the duration of each visit. For this analysis the SLS considers the timing values (i.e., duration of a visit) as a multiple of a small interval of time dt. Hence, on each histogram we memorize the probability for a generic $User_A$ to be in a certain PoI_P during a day_D of the week (e.g., sunday,monday, ...), for every interval of time *dt*. In fact, the system associates a probability value to each interval dt of the day. While updating the history of the user, each daily histogram is updated by means of the following learning formula:

$$P_{new}\left(PoI_P \mid day_D, dt_t\right) = \\ P_{old}\left(PoI_P \mid day_D, dt_t\right) * \alpha + \\ isCurrentlyVisited * (1-\alpha) \tag{5.8}$$

Where the possible values for the variable *isCurrentlyVisited* are $\{0,1\}$, which correspond respectively to the possibility that the related PoI is currently visited in that day/dt (value 1) or not (value 0). The old probability to be in PoI_P during day_D in the time interval dt_t is weighted by the variable α, and is summed up to the $(1-\alpha)$ value, in turn multiplied by the $isCurrentlyVisited$ variable. The value of α used during the experiments, has been determined empirically (more details in section 6.3.1). Also in this case applies the same reasoning presented before: assigning a larger weight to the history (past data) with respect to the new samples, allows the learning algorithm to be more impacted by the history of the user. However, the computation of the predictions does not have to be heavily affected by very old activities of the user. Additionally we cannot give the same weight to the history and to the current data. The best value of α in general is in between these two opposite situations.

Every time a user collects a sample data, all the PoI's histograms related to the corresponding tuple $\{day_D, dt_t\}$ are updated accordingly. At the end of each day, once the sampled data has been processed, the results of the clustering procedure and the consecutive time filtering results are represented by a sequence of actions. Each action refers to a visit to a PoI and is characterized by three main values:

- *cluster id*: each PoI is identified by a unique id number, which is used to retrieve its related information (e.g., centroid's coordinates, density, standard deviation of its points' distance to the centroid);

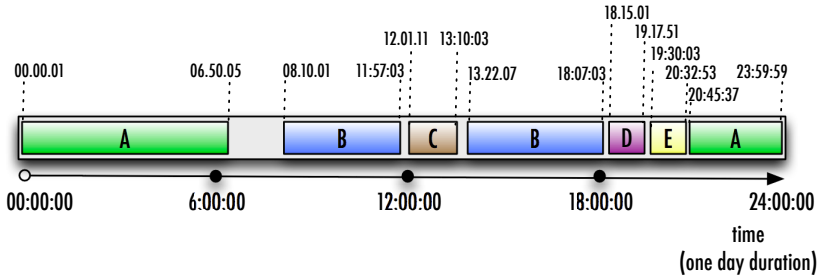


Figure 5.2. User's daily actions: example of sequence of visits to relevant PoIs

- *start time*: starting timestamp of the PoI's visit;
- *end time*: ending timestamp of the PoI's visit.

An example of a daily sequence of actions learned by the system, is represented in figure 5.2. In this figure, the user is visiting five different PoIs, and some of them are visited twice in the same day.

A timestamp information is then converted in a tuple $< day, time_{dt} >$ where *day* is the corresponding day of the week (e.g., sunday, monday, ...) and $time_{dt}$ is the index of the interval of time during the day, inside which the timestamp has its value. For example, considering an interval of time dt equal to 10 minutes, and a timestamp related to a clock time of $10:01:00$ *a.m.* (601 minutes from the beginning of the current day), the corresponding $time_{dt} = 60$ (number of dt intervals after the last midnight, $60 * dt$).

While the learning phase is executed, the user's learned history continues being updated with newly learned location data. Thus, the number of known PoIs increases significantly, even though only a small subset of them have a significant probability (relevance for the user). Therefore, the algorithm removes from the user's history all those PoIs with a global probability lower than a minimum threshold (more details in chapter 7).

5.5 PoI Classification

In this section, I introduce a classification algorithm of the Points of Interest visited by a user. This classification allows the definition of a general profile of user, characterized by the number of visited locations and time spent there. I present here my experimental approach, starting from real traces and deriving a statistical evaluation. I use the results to this evaluation to separate the visited PoIs into 3 main classes of relevance: *High, Medium and Low Interest Locations*

- *HILs, MILs, LILs*[3] (Giordano and Papandrea [2012]). I show that, on average, people visits just few locations (HILs), but they spend there more than 50% of their time. Also the MILs are low in number, and they are visited for about 10% of the time, while the remaining points, the ones in the LILs class are visited for a very short amount of time.

5.5.1 Relevance

After identifying the set of PoIs relevant for a user (as described in section 5.3.1) I classify the visited PoIs according to their relevance to the user itself, this allows the characterization of the user's mobility within each locations' class of relevance. The *relevance* of a certain location L_i is calculated on the mobility history of each user, and it is defined as:

$$relevance(L_i) = \frac{d_{\text{visit}}(L_i)}{d_{\text{total}}} \tag{5.9}$$

where $d_{\text{visit}}(L_i)$ is the number of days a location L_i has been visited (one or more times per day) by the user; the d_{total} is the total number of sampling days, collected by the user. The relevance of a certain location is, according to the formula, the percentage of days the user visit this location, over the total number of days of sampling.

According to relevance values, I show that the PoIs associated to each user can be grouped in 3 classes:

- **High Interest Locations (HILs)**: locations most frequently visited by the user. It is easy to infer their semantic meaning, and associate them to home location, work place, gym, etc.
- **Medium Interest Locations (MILs)**: locations of interest for the user, but visited just occasionally.
- **Low Interest Locations (LILs)**: PoIs unlikely visited more than very few times.

The evaluation of the PoIs' relevance allows a straightforward identification of these three classes. For example, in figure 5.3 I show a cumulative characterization of the PoIs identified for all the users of the GeoLife dataset, introduced in section 2.3.1 (I used here the complete dataset, without applying any filter to the

[3]Called respectively *Mostly, Occasionally and Exceptionally Visited Points of Interest - HILs, MILs, LILs* in Giordano and Papandrea [2012]

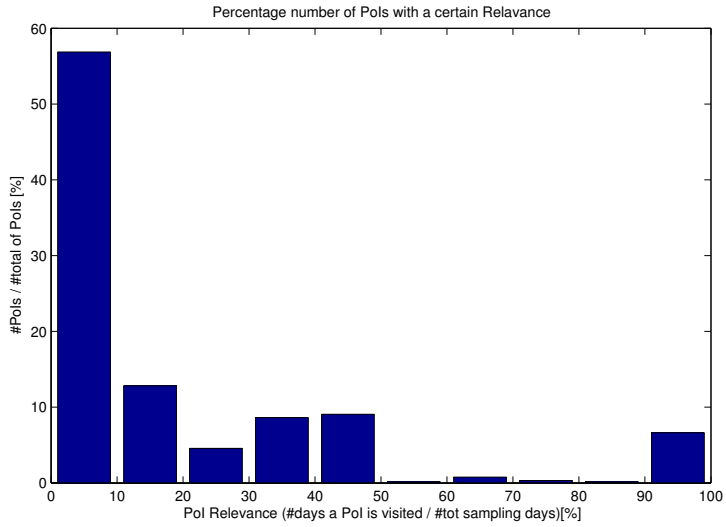


Figure 5.3. PoIs Relevance

data). In the x-axis we identify 10 bins of relevance spanning form 0% to 100%, where each of them has a width of 10%. For each interval of relevance I show in the figure the average percentage of the number of PoIs (calculated over all the users) belonging to the corresponding class. From the figure, it is easily visible that, on average, 57% of the PoIs visited by a user are within the LIL relevance class: this means that more than half of the PoIs seen by each user, are Low Interest Locations, that the user hardly visits for multiple times, hence locations with a low relevance value. 6.7% of PoIs can be classified within the HIL class (locations with high relevance value): this gives an idea of the limited number of locations which are visited by each user almost daily. The identification of the upper and lower bounds for each of the three classes is strictly related to every single user; in fact it depends on the user's mobility style. The next section will explain how to identify the classes bounds for each user.

5.5.2 Finding classes of relevance

As it has been highlighted by the above discussion, relevance class bounds could change among the subjects. As a consequence, class bounds cannot be fixed *a priori* but claim at an automatically detection algorithm able to adapt to the single user mobility pattern. In particular, I adopt an unsupervised approach which groups the PoIs of a single user according to their relevance and maximizes their separability. The chosen clustering algorithm is the *k-means* with $k = 3$ which

corresponds to the number of classes of PoIs. To avoid the problem related to the initial choice of the centroids, I run 10 replicas of k-means with different initial seeds and choose the partition that minimizes the within-cluster sums of point-to-centroid distances, thus maximizing the separability. Figure 5.4 shows the result of the k-means clustering on a sampled user. The LIL class (purple box) covers the relevance range from 0.01 to 0.12, the MIL (red box) spans the range from 0.16 to 0.46 and the HIL class (green line) contains only one PoI with relevance 0.82.

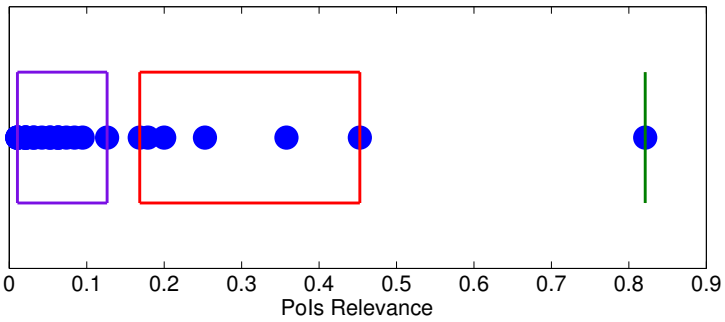


Figure 5.4. Three classes of relevance in a sampled user

5.5.3 Data pre-processing

Although GeoLife (section 2.3.1) represents the most reliable dataset publicly available, it was not collected to extract PoIs and thus it need some pre-processing in order to find the most meaningful trajectories to our goal. The need of a pre-processing phase is dictated by the dataset bias which flavors movements, while in this study I am basically interested in people which are still while visiting their PoIs. In particular the applied data pre-processing aims at densifying trajectory points corresponding to the pause phase by a filling heuristic, while removing

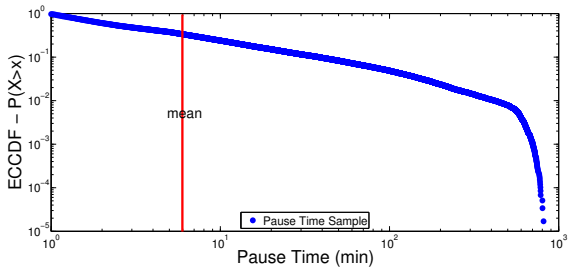


Figure 5.5. ECCDF of the aggregated pause times in the stay-locations.

points belonging to users' movements.

Indoor filling Mobility data collected by GPS devices present gaps because GPS signals are often disrupted inside buildings. This might represent a big problem, especially if one is interested in detecting the PoIs of a user. In fact, in many cases, buildings or other indoor locations represent the majority of the PoIs visited by a person during the day. To overcome the problem given by missing records (Lin et al. [2012]), so to avoid an underestimation of the number of PoIs, I apply the following simple rule.

Indoor filling rule: when the ending and beginning GPS points of a gap are within a distance of 35 meter and the gap duration is greater than 5 min, the user is taken as residing at the same location during that time.

This rule also supplies for the situation where the individual enters a building, or where the individual turns off the GPS device in an indoor place. Practically, I add as many GPS points equal to the entry point as the duration in sec of the gap. After the trajectory reconstruction phase, I noticed a big increment of points, anyway limited by the threshold imposed on the gap duration.

Movement phase reduction A filter has been applied to leave out data which describes the movements among the PoIs a user visits, thus reducing the number of points to analyze. This way we consider the periods in which a user stays still in a place, assuming that users manifest their interests by spending an amount of their time in such places. In order to extract the pause periods and their related GPS points from the whole individual trace, I apply the heuristic proposed in [Zignani et al., 2012; Zignani and Gaito, 2010], where a similar but smaller dataset has been analyzed.

If two points p_i and p_{i+1}, with timestamps indicated by $t(p_\cdot)$, do not satisfy

$$\frac{\left\|p_{i+1} - p_i\right\|}{t(p_{i+1}) - t(p_i)} \leq \Delta \tag{5.10}$$

then we delete p_{i+1} from the original trace, since it belongs to the movement phase.

Analyzing walking mobility data, I set the threshold to the very low value of $\Delta = 1.3 m/s$, according to the fact that human walking speed is about 4-5 km/h

(1.1-1.4 m/s). It seems a reasonable value as generally, while visiting a relevant location, people do not reach this maximum speed. This way, it is possible to capture points where a person is still or is moving very slowly inside a small area. The result of the speed filtering process is a sequence of points that forms the trajectory $S = ((p_1, t_1), ..., (p_n, t_n))$, where t_i is a timestamp and $p_i \in \mathbb{R}^2$, on which it is applied the PoIs extraction methodology proposed in section 5.3.1.

Users' selection The point reduction has also effects on the number of users and the number of days, per user, from which I can extract places of interest. When analysing the GeoLife dataset, the reduction is mainly due to the fact that it has been built for the transportation prediction task, and, as a consequence, it flavors movements. To overcome these limitation, I select the users considering two properties.

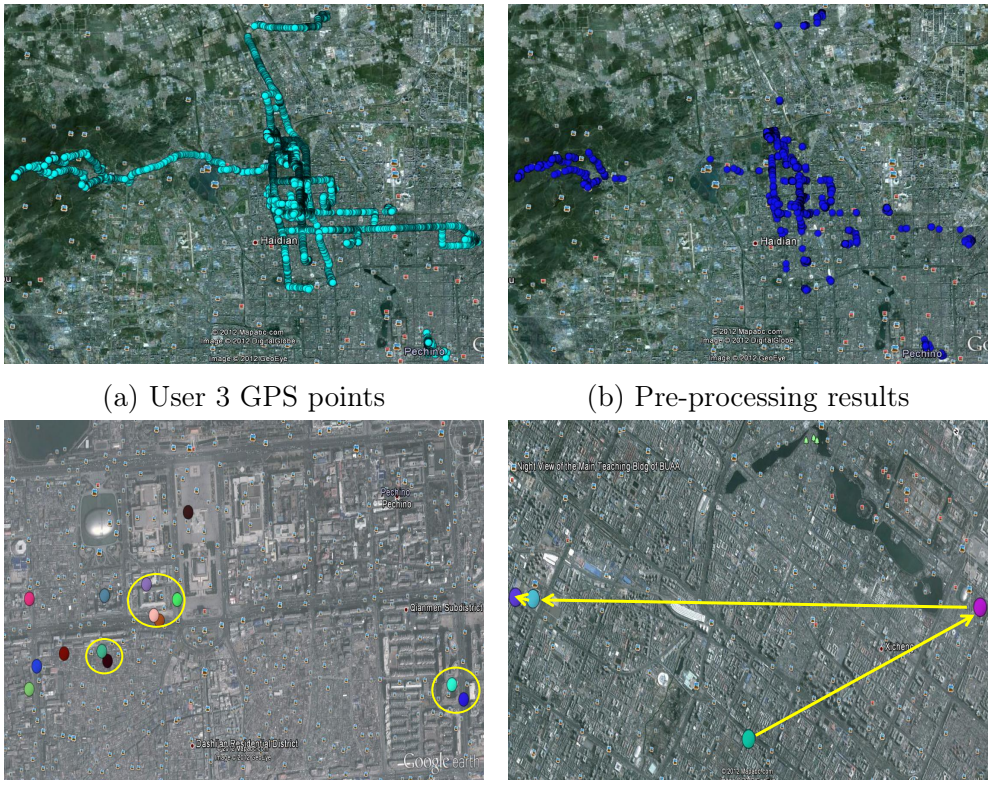
1. Period (in hours) a single day trace spans.
2. Number of days the single user traces cover.

In particular, for each user, I only consider the daily traces that record more than h hours. On these tracks I count the number of users that have more than d days of data. In particular, for all the users of the dataset, I filter out all the days of sampling (data collected within the 24 hours, going form 00:00 AM until 11:59 PM) which have $h \leq 3$ hour of sampling. All the remaining days are considered *relevant days*. After this first processing, I filter out all the users which collected less than 20 *relevant days* of data ($d < 20$): applying these filters to the GeoLife dataset, the resulting number of users is 21, over the total number of 178 users.

I apply these values for the users filter parameters, in order to optimize the trade-off between the importance of having a large number of users, to be able to generalize our analysis; and the need to deal with sampled data which does not only correspond to trajectories. For example, only by increasing of one hour the threshold h I obtained a number of users that is not enough to our goal (10). Though the analysed dataset is a collection of trajectories, hence only a reduced subset of collected data fulfill the specified requirements, I am able, also with this small filtered dataset, to obtain powerful results. Besides, note that the resulting dataset almost completely spans the original GeoLife period.

Finding PoIs

The identification of the PoIs in a trajectory dataset is slightly different compared to the approach we presented in section 5.3.1, which is mainly intended to analyse continuous datasets. In this case we applied a slightly different procedure.



(a) User 3 GPS points

(b) Pre-processing results

(c) Sub-PoIs issue

(d) Compact representation

Figure 5.6. PoIs extraction applied to the user 3's trajectories. In 5.6a we plot all the recorded points (raw data). In 5.6b we show the points resulting from the application of the pre-processing phase. In 5.6c the sub-PoIs that have to be grouped in the real PoI (yellow circle). In 5.6d a compact representation of user 3's mobility during a single day.

All the points of the trajectory dataset which belong to the pause phase and are the starting points to extract the PoIs. To reach this goal, I first find the possible *regions of interest* via a clustering algorithm and then I detect the real PoIs considering a pause time feature.

The possible regions of interest are identified by introducing the concept of *stay-location* L. A *stay-location* is an area where a person stops, independently of how long she/he stays there. Let us consider individual traces in order to extract stay-locations and analyze their properties. Also in this approach, to find stay-locations I apply the density-based clustering algorithm DBSCAN. The used parameters are $\epsilon = 10$ mt and $minPoint = 2$ neighbors (as explained in section 5.3.1, ϵ represents the maximum distance such that two points are considered

neighbors, while $minPoint$ is the minimum number of neighbors that a node must have to be considered in a cluster).

The first results show that in daily movements, there are many stay-locations where an individual stays for a short amount of time. Many of these stay-locations are meaningless as they represent small pauses in the movement towards the real destinations, the PoIs.

In the following analysis of the dataset I introduce a pause time feature, setting the threshold $\phi = 5$ min, which corresponds to the mean of the pause distribution in stay-locations, shown in Figure 5.5. In this analysis I do not consider the sum of the pause times in a stay-location; but only the single values. The thresholding results in the meaningful PoIs, although we observe situations, such that presented in Figure 5.6c, where there are many sub-PoIs of the same general PoI. To overcome this issue I run a second passage of DBSCAN with a larger ϵ on the centroids of the sub-PoIs detecting the real points of interest. Thus, the processing has two important effects: a drastic reduction of the number of stay-locations and the inference of the real PoIs.

Aside from finding PoIs, the above methodology has the capability to express human mobility as a compact trace that summarizes the transitions between PoIs and the users' pause time in them as shown in Figure 5.6d. Adopting this compact representation in the following section I can analyze some properties of the human mobility and of the PoIs human beings visit during their daily movements.

5.5.4 Experiments and Results

In this section I present the experimental results of the analysis performed over the GeoLife data after performing the pre-processing and the analysis of the PoIs and related classes of relevance, described in the previous sections. For each filtered user, I apply the k-mean algorithm (as explained in section 5.5.2) to classify the related PoIs in three main classes of relevance (section 5.5.1) and over these classes I study three main features:

1. the number of PoIs which reside within each class of relevance,
2. the percentage of time spent in each class,
3. the average time of the visits to the PoIs of the classes.

Figure 5.7 represents the number of PoIs associated to each class of relevance, per user. In the upper plot we can notice the large difference in the

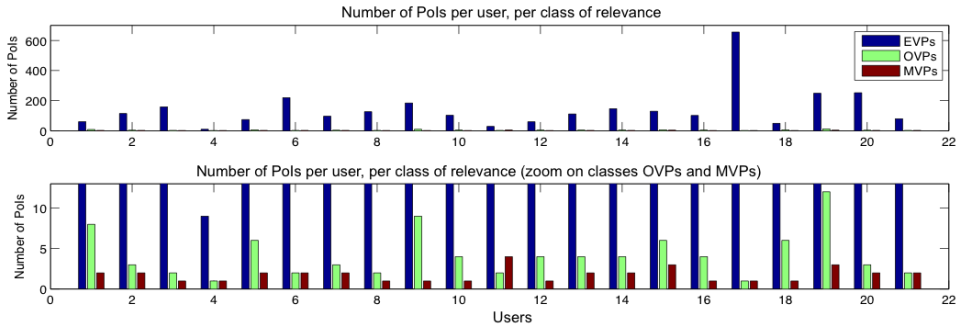


Figure 5.7. Number of PoIs per class of relevance

number of LILs, with respect to the PoIs belonging to the other two classes of relevance (MILs and HILs - which can barely be seen): this is an evidence of the fact that the user always visits new locations, but only few of them are visited regularly. In the lower plot, we zoom on the classes MILs and HILs: the number of MILs is limited and its average value is 4.19; also for the HILs the number per user is limited, and its average value is 1.76.

As expected, each user has a very small number of preferred locations (HILs) which are visited daily (e.g., home, work place), and a higher but still limited number of location of interest (MILs) which are visited with a lower frequency but regularly (e.g., gym, favorite restaurant, parent's house).

Figure 5.8 shows the average visit time to the PoIs, according to their class of relevance. From the figure we notice that for all users, the average visiting time to LILs is very limited and on average lower than one hour. The average visiting time for MILs and LILs depends to the mobility style of the user: some users tend to spend long time in their HILs, other users instead, use to have very long visits to the MILs. We will talk about the classification of the user's behavior below in this section. However, considering the PoIs classification, the HILs and MILs can be considered equally relevant for the user, even if the HILs are visited more frequently and more periodically than the MILs. The LILs are instead locations not really important to the user, and where (according to the figure) she/he spends on average a shorter interval of time.

In Figure 5.9 we represent a cumulative measure of the percentage of the total time each user spends visiting PoIs belonging to the three different classes of relevance. According to this figure, a user tends to spend more than half of the total time in the HILs and the rest of the time is almost equally distributed between the LILs and the MILs.

The interesting aspect of this analysis is further exploited in Figure 5.10, where we show the *percentage of the visit time per class*, for two different users.

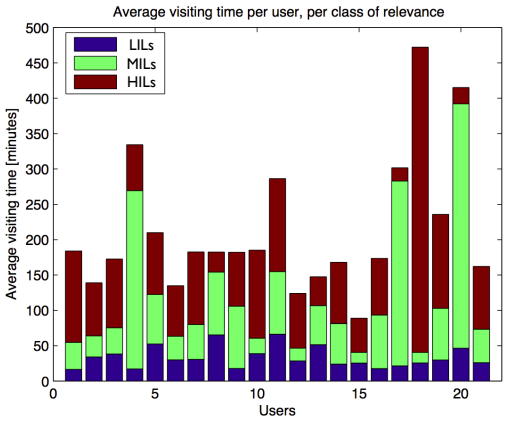


Figure 5.8. Average visiting time per class of relevance

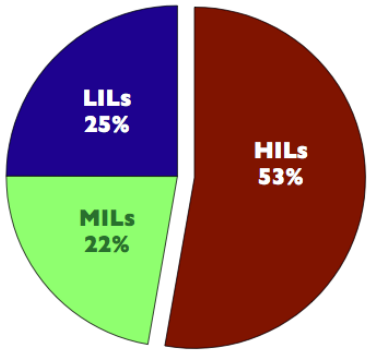


Figure 5.9. Percentage of the visiting time, per class of relevance (Cumulative)

While always showing the three classes pattern, the behavior of the two users radically differs. The user 69 has a very *creature-of-habit* behavior: it spends close to the 81% of the time in the HILs, and less than 9% in the LILs.

As opposite, user 25 is a *globetrotter*: the percentage of time spent in the HILs is below 10% (rounded to 10% in the figure), and the user spends most of the time in the LILs (close to 73%), even if the average time spent in each LIL is still significantly smaller than the average time spent in each HIL. This opens for new research approaches to human mobility based on visited places distribution.

5.6 Key Features in Human Mobility

Starting from the classification of the user's PoIs in classes of relevance, I derive some features which could drive how a user moves among interesting places.

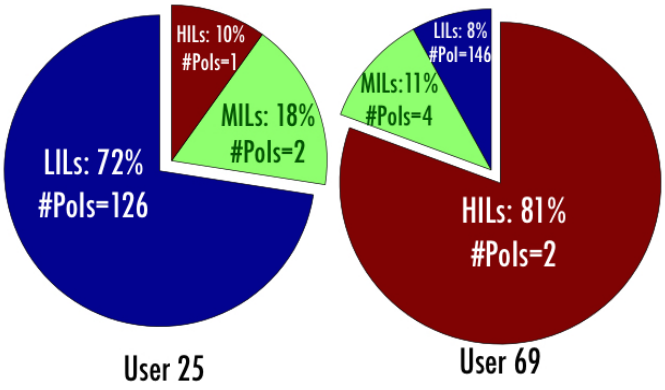


Figure 5.10. Different types of user mobility: the globetrotter (user 25) and the creature-of-habit (user 69) behavior derived by the time and number distribution of PoIs.

Specifically, in this part of the work I am not actually interested on the transportation mode a user adopts between different PoIs (i.e., which vehicle is used) or whether she/he tends to repeat some patterns, but only on locations visited consecutively and their associated relevance classes.

I characterize the users' mobility according to three different features:

1. the *relevance R*, accounting for the interest of a user for a PoI,

2. *geographical distance* $\triangle r$ between the departure D and the arrival A PoIs during the user's movement,

3. the *transfer time tt* between D and A.

The geographical distance has been traditionally chosen as fundamental feature of mobility studies because it is objective and constant. On the contrary, the relevance can be a subjective feature and can vary over time, as well as the time transfer is evidently not constant, as the duration of the same transfer, carried out under different conditions, can be affected by many factor.

To validate my study I performed the analysis over the two datasets presented in section 2.3, both the continuous dataset and the trajectory one. To be able to use the *GeoLife trajectory dataset* I performed a pre-processing of the data as described in the previous section (5.5.3). I used the filling heuristic specified before (Papandrea et al. [2013]) to densify the trajectory points corresponding to the pause phase, while removing points belonging to the users' movements. To summarize, I selected from the dataset a subset of significant users who have

collected at least 20 relevant days of data, where a relevant day includes at least 3 hours of location sampling.

Also for the *continuous dataset* I perform a data filtering, selecting a subset of significant users which have collected at least 14 relevant days of data (two weeks), where a relevant day includes at least 6 hours of location sampling. The resulting number of relevant users I considered for this study is 6. To identify the users' relevant PoIs, in this case, I only act on the algorithm tuning (Papandrea and Giordano [2014, 2012]).

5.6.1 Feature 1: Relevance

As already discussed in section 5.5.1, the *relevance* of a PoI allows to determine how likely an individual will move towards that place or return to it according to her/his history and how the place is important in the user's daily routine. The relevance distributions shown in Figure 5.11 exhibit a power-law behavior, where a huge number of PoIs are visited only few times, while few PoIs are commonly visited and have a very high value of relevance. Thus during a long observation period the number of preferred and recurrent locations are very limited.

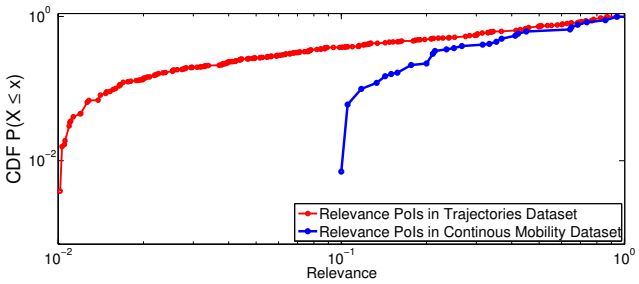


Figure 5.11. Cumulative distribution function of the PoIs relevance for both datasets.

According to the relevance values, we have shown in section 5.5 (Papandrea et al. [2013]) that the PoIs associated to each user can be automatically grouped in 3 classes: *Low Interest Locations (LILs)*, *Medium Interest Locations (MILs)* and *High Interest Locations (HILs)*. Figure 5.12 shows the histogram of the percentage of PoIs in the three classes of relevance: as visible from the figure, the PoIs partitioning in classes is common for both the datasets. Of course the LIL is the widest class due to the distribution of the relevance CDF.

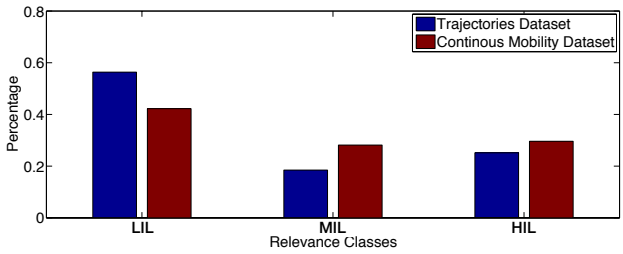


Figure 5.12. Comparison of the relevance class distribution.

5.6.2 Feature 2: Geographical Distance

I measure the geographical distance between the departure PoI *D* and the arrival PoI *A*, by considering their centroids and adopting the haversine formula (Sinnott [1984]) to incorporate the Earth curvature. A few works in literature (Rhee et al. [2011], Gonzalez et al. [2008]) have studied the mobility characteristic so far. They showed that it follows a Pareto distribution with an exponential cut-off due to the spatial limits of human mobility and suggest that human movements can be modeled by a Levy-walk process. As it can be seen in Figure 5.13, the same kind of distribution can be observed in both datasets up to different geographical limits (longer tail in the GeoLife Project dataset). As a consequence, these results validate the previous works that consider only the spatial distance when describing mobility of human beings.

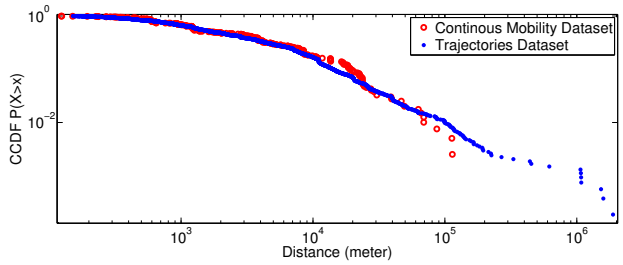


Figure 5.13. Complementary cumulative distribution function of the distance between consecutive PoIs for both datasets.

5.6.3 Feature 3: Transfer Time

How far are two PoIs apart? Remoteness has always been measured by means of PoIs geographical distance; we observe that distance can also be expressed

in terms of transfer time, i.e the time needed to move from the departure *D* to the arrival PoI *A*. The transfer time distribution of the dataset, as shown in Figure 5.14, is also a power-law with a cut-off but it smooths the long tail of the geographical distance distributions. Specifically, in the spatial case both distributions coincides unless few points in the tail, however if we consider the transfer time, people behave differently. In fact the cut-off values are totally different; about 1.5 hour in the continuous dataset and 4-5 hours in the GeoLife one.

The impact of this observation is fundamental as it suggests that time and space do not always match and, in particular, they do not match whenever long geographical distances are considered. I argue that the shorter tail in the time distribution is due to the fact that, as opposite to geographical distance distribution, in the time transfer analysis there are less occurrences of events far from the mean. It is unusual that people spend more than few hours in commuting between PoIs, while it is not unusual that the PoIs are far from each other, but connected by fast transportation means.

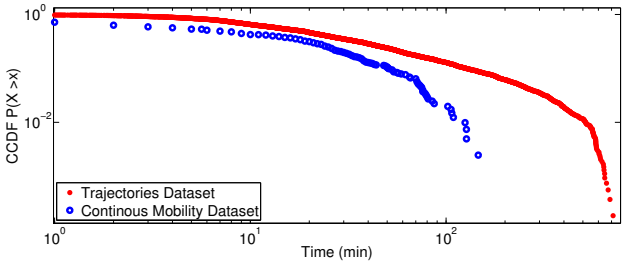


Figure 5.14. Complementary cumulative distribution function of the transfer time between consecutive PoIs for both datasets.

5.6.4 Time Transfer and Geographical Distance Correlation

In our daily life, we decide to move towards a particular place if we have enough time; by contrast, the current mobility analysis is driven only by the geographical distance. This dichotomy derives from the implicit assumption that time and distance are strictly related. Although this is roughly true on small scales, it comes out that this is not perfectly working when the scope of mobility enlarges to, for instance, metropolitan or regional areas. To shed light on this aspect of human mobility we have computed the *Pearson Correlation Coefficient* between geographical distances and transfer times on both dataset, defined as:

$$PCC(tt, \Delta r) = \frac{\sigma_{(tt,\Delta r)}}{\sigma_{tt} * \sigma_{\Delta r}} \tag{5.11}$$

where $\sigma_{(tt,\Delta r)}$ is the covariance between the temporal and the geographical distances respectively, σ_{tt} and $\sigma_{\Delta r}$ indicate their variances.

Dataset	PCC
Continuous Mobility Dataset	0.4
Trajectories Dataset	0.1

Table 5.1. Pearson Correlation Coefficient (PCC) between geographical distances and transfer times on the Trajectories and Continuous Mobility Datasets

As shown by Table 5.1, when applied to the continuous dataset, the Pearson Correlation Coefficient is equal to 0.4, which indicates a small/medium degree of correlation; however, if we consider the GeoLife dataset it is equal to 0.1, meaning that the two quantities are not correlated. In wider areas the adoption of different commuting strategies makes lose the proportionality between the transfer time and the distance, typical of movements in small regions. To deepen this issue I show in Figure 5.15 the relation between geographical distance and transfer time. Considering a displacement typical of the urban/metropolitan area, we observe that the average transfer time has a sub-linear trend accounting for the increasing speed of the different forms of transportation adopted to contract the geographical distances. This observation strengthen the intuition that temporal and spatial metrics capture different distances as the second contracts the first one.

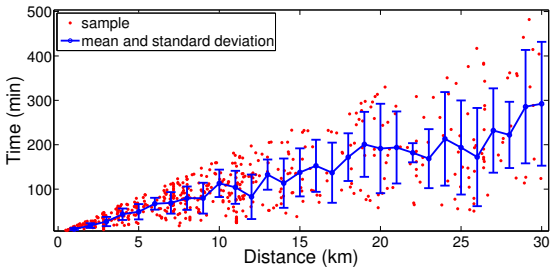
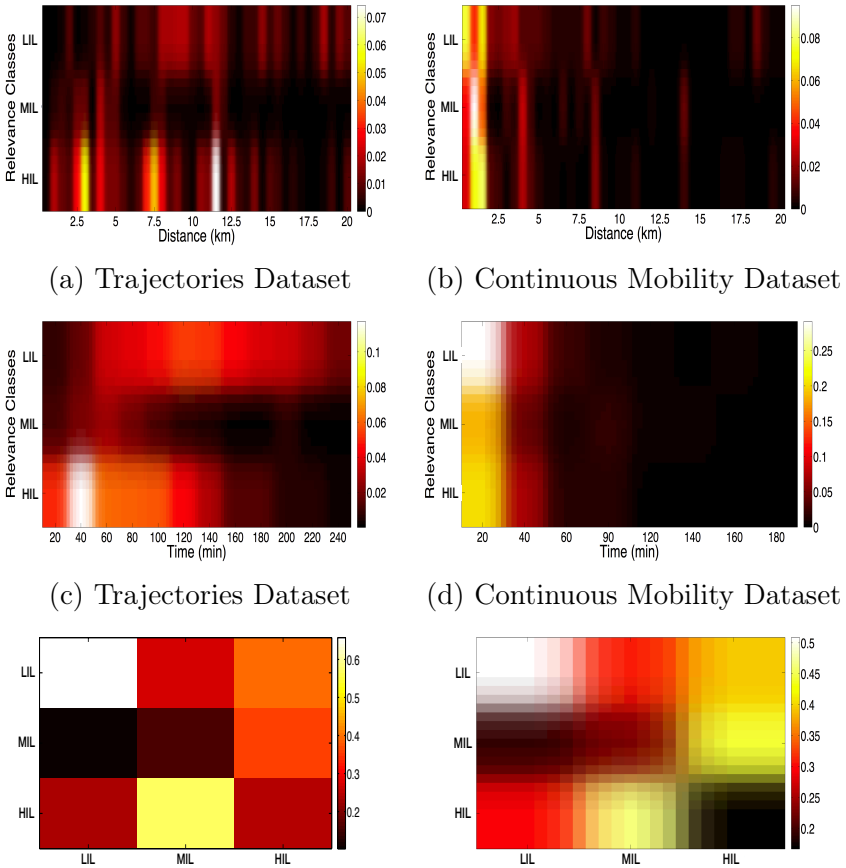


Figure 5.15. Relation between the traveled distance and the transfer time. Red dots presented the sample extracted from the GeoLife dataset and the blue line represents the mean trend (error bars correspond to the standard deviation).



(a) Trajectories Dataset

(b) Continuous Mobility Dataset

(c) Trajectories Dataset

(d) Continuous Mobility Dataset

(e) Class Transition in Trajectories Dataset

(f) Class Transition in Continuous Mobility Dataset

Figure 5.16. 5.16a) and 5.16b): joint probability distribution of the distance between consecutive PoIs and the relevance classes, $P(x \leq \Delta r < x + \delta, class = C)$. According to the heat bar, yellow and white squares represent higher probability. As regards distance we adopt 500 meter bins from 0 to 20 km ($\delta = 500$m). 5.16c and 5.16d: joint probability distribution of the transfer time between consecutive PoIs and the relevance classes $P(x \leq tt < x + \delta, class = C)$. According to the heat bar, yellow and white squares represent higher probability. In this case, we adopt 20 min bins from 0 to 4 hours ($\delta = 20$min). 5.16e and 5.16f: transition probability among relevance classes. Each square represents the conditional probability to move from a PoI in a class c_1 to a PoI in a class c_2 $P(C_{new} = c_2, C_{old} = c_1)$. On the x-axis the conditioning variable C_{old} and the on the y-axis the conditioned variable C_{new}.

5.6.5 Transition rules

The human decision to move from a point to another emerges from a complex decision making process that is influenced by a variety of human and contextual behavior. To improve the understanding of this process, I want to measure the impact of *relevance*, *distance* and *time* on the choice to get to a given arrival PoI *A*. Let us consider the movements between a departure *D* and an arrival *A*. Each transfer is characterized by the geographical distance between the two PoIs, the transfer time, the class of relevance of the departure PoI *D* and the class of relevance of the arrival PoI *A*.

I start by investigating the impact of the geographical distance on the destination's selection process. To this purpose, I specifically analyze human behavior for the three relevance classes, LIL, MIL and HIL and I group the distance values in 500m bins. As shown in Figure 5.16a and 5.16b where the joint probability of distances and classes is depicted, the behavior is very similar in both datasets. In all the three relevance classes of destination we note a not monotone decrease of the visiting probability with a non negligible probability that people move also toward more distant PoIs, as predicted by a Levy-walk process and indicated by some peaks of brighter color in the right part of the Figures.

A different behavior can be observed when we consider the transfer time instead of the geographical distance. The visiting probability in the MIL and HIL is monotonically decreasing (color blur from white to dark brown) with the temporal distance and reaches values close to zero according to different cut-off values, as shown in Figure 5.16c and 5.16d. This demonstrates that the transfer decision process of individuals is driven by the time they need to get to a place, as people are prone to save their time. This observation advocates a paradigm shift in the analysis of human mobility: *the time, not the distance, is the main parameter governing human decisions on movements*. Furthermore, although not monotone, the transfer time trend in the LIL is much more smoother than in the geographical case. In particular we can say that people who want to visit LILs are willing to spend more time to reach that places, as the highest probabilities are shifted towards 2-3 hours. This is due to the fact that also a technological component affects human mobility, as people use different transportation means for different scale of distances. When people move in small areas, as in the continuous dataset, the commutation times do not differ much for different type of transportation. As opposite, when we consider a large area, as in the trajectory dataset, the commutation times are highly affected by the transportation mean.

Finally, the impact of the class of relevance of the departure PoI is independent of the scale of the scenario. As we can note comparing Figure 5.16e

and 5.16f, both datasets presents the same characteristic despite the different geographical areas they span. Even if the conditional probabilities are heavily affected by the great number of LILs, people commute to/from MILs from/to HILs, *i.e.* medium interest locations as pub or hobbies are related to home/work places (high interest locations). Clearly, even if people have to cover longer distances, they keep on moving between the places they stay the most (HILs: home and work), and some other MILs (e.g. gym), and distance affects only the transitions to LILs.

5.7 Conclusion

In this chapter I described the analysis performed on the trajectories and the continuous datasets presented in section 2.3, in order to evaluate a proper learning methodology for the user's mobility behavior, and some interesting properties.

A first analysis of the location data resulted in the identification of user's relevant Points of Interest. The learning procedure of the user's mobility model firstly identifies these PoIs and the connection between them, according to the consecutive visits of the user. A first learning algorithm develops a user prediction graph with this data, to be able to perform next location predictions according to the current location of the user in the graph. A second learning algorithm is presented, which includes the timing features, hence allowing the user's next location prediction given the location and the time context.

The user mobility is then analyses in order to evaluate the impact of three different features on the user's mobility choices. The first feature is the *relevance*: how important is a PoI for the user, and how this feature drive the user's movements. The PoIs can be classified according to their relevance value, into three different classes: HIL, MIL and LIL. According to the analysis described above, a user tends to have a very limited number of HILs, where it spends most of its time; instead the number of LIL is very high, and the user tends to spend there a very limited time. The MIL are also limited in number and the user tends to visit those PoIs regularly but for a short amount of time. The PoIs classification is important for the SLS because it helps in reducing the amount of used storage, differentiating between important PoI for the mobility prediction model (HILs and MILs) and not relevant ones (LILs).

The other analyzed features are the *geographical distance* between pairs of PoIs, and the *transfer time* between them. The presented analysis reported a very interesting result. Contrarily to most of the current mobility analysis available in the State of the Art, where the geographical distance is considered the

driving feature for the user mobility, I demonstrated above that the transfer time between PoIs is more significant for the user.

More details about the application of the presented models for the user's location prediction is presented in the next chapter.

5.8 Remarks

Part of the work presented in this chapter (sections 5.5 and 5.6) is the result of a collaboration with Matteo Zignani, Sabrina Gaito and Gian Paolo Rossi from the *Universitá degli Studi di Milano*, which has been published in Papandrea et al. [2013] and Zignani et al. [2014]. My contributions to the presented work consists in the *continuous dataset* collection and analysis, the identification of the existence of PoI relevance classes (Papandrea and Giordano [2014], Papandrea and Giordano [2012]) and the analysis on the relevance feature on both the datasets, the *continuous* and the *GeoLife trajectories* one.

Chapter 6

Mobility Prediction

The mobility model implemented by the SLS characterizes the movements of a user among its PoIs. In the following I refer to both the approaches presented in the previous chapter, to build this user mobility model. In section 5.4.1 I presented the location based mobility model; in this chapter I will use this model to perform movement predictions using the datasets presented in section 2.3, and I show that it allows to learn about the user habits and to perform a quite good prediction of regular actions. However, this model can be improved, in fact in section 5.4.2 I introduced the timing parameter to the model, and I will show in this chapter the better performances of the new model.

6.1 Introduction

A human being, by its nature, needs to have some fixed reference points in life. These are usually: home/family, work place and some hobbies and leisure activities. For this reason, people tend to be repetitive in their behavior, and visit more frequently few key locations (Isaacman et al. [2011]; Zhou et al. [2007]). In Isaacman et al. [2011], the authors show that it is possible to identify these reference points and in particular home and work locations, and present as possible applications of how this information can be used, the calculation of home-to-work commute.

New technologies are providing enormous possibilities to foster new systems for following and supporting people in their daily commutes, and several works in literature tried to capture and model this mobility for: empowering and simplifying new location search, best paths finding [Romoozi et al., 2009], quality of service [Akoush and Sameh, 2007], reducing the energy consumption of wireless network's communications Chakraborty et al. [2006], etc.

Accordingly to this tendency, I start from this observation to build a simple but powerful learning and prediction model, which can be easily implemented and used on a mobile device, which allows a user to dynamically update its own mobility model, in terms of key locations, periodicity at different time-scales, duration, and it gives the potentiality to self-predict where the user will be next.

Researchers have performed extensive work in the area of mobility prediction, especially in networking communities [Song et al., 2006; Nicholson and Noble, 2008]. The most common approach consists in comparing the current mobility pattern with historical data, in order to extract similar patterns to predict next location. Song et al. [2006] used a two-year trace dataset collected on a campus-wide wireless network, and found that complex prediction models were at best only negligibly better than the simple ones (i.e., Markov predictor) in practice. However, since the target environment of the presented thesis is a resource constrained environment, a simple prediction model is preferable because of its low computation overheads and low storage requirements [Nicholson and Noble, 2008] (evaluated in chapter 7).

An interesting related work is SmartDC, presented by Chon et al. [2014]. The authors applied a simultaneous mobility learning and prediction scheme to mobile phones, to gradually learn a user's mobility pattern, and optimize sensing schedule using the predictable regularity of individual behavior. The authors implemented two types of location predictor: Markov predictor as a location-dependent predictor, and nonlinear time series analysis as a location-independent predictor. The main purpose of this model is to predict the visiting time of a PoIs, when the user reaches it. However, this approach requires a quite long time to learn the user behavior (i.e., it requires three months to reach around $72 \pm 9\%$ residence-time predictability). The SLS solution instead is faster in learning the user mobility regularities and also provides a prediction algorithm which is location-based, However it is independent from the previously visited PoIs: this allows the SLS to be able to predict the user's next visited location also when there is no pattern repetition in the sequence of PoIs visited.

While in the SoA some other systems have been presented to predict next location, many of these techniques are not able to provide accurate predictions from a spatio-temporal perspective Krumm and Horvitz [2006]. Scellato et al. [2011] presented a location prediction system based on non-linear time series analysis of the arrival and residence time of a user in relevant places. This system focuses on the temporal predictability of users presence, when they visit their most important places.

An interesting work has been presented by Baumann et al. [2013], where the authors compared a set of 18 different next-place prediction algorithms,

able to reach a high prediction accuracy, but providing low reliable transition predictions. A set of performance features are identified and calculated over this set. They stated that a high average prediction accuracy can be obtained even by "naĭve" algorithms which are therefore largely unable to detect transitions between different places. Finally the authors presented a solution called MAJOR whose approach consists in exploiting the 18 algorithms and performing a majority vote to compute a final prediction, achieving both high prediction accuracy and reliably predict transitions, at the cost of a higher computational and memory overhead. Therefore this approach is not a good candidate for a mobile implementation, when the optimization in resource usage (including computational and memory costs) is essential. However, the authors focus on an important point, and in particular on the transition prediction accuracy: in my work I considered this feature and calculated it for the *Location-based algorithm*, where the next-place prediction is based only on the location context (section) and on the learned transition probabilities; the *Location and Time-based algorithm* instead is based on the PoIs visits repetitiveness of users, in order to perform next-place predictions.

In my work, the goal is to learn the user's behavior and to build a probabilistic mobility model, which allows the prediction of the current user's location independently from the location visit duration. However, an interesting and innovative aspect of this approach is that the model is meant to be built and run locally on a mobile phone. This opens to a huge number of location based applications, starting from simple localization Papandrea and Giordano [2012], which can be run locally without either global knowledge or access to remote information systems.

The rest of the chapter is structured as follow. Section 6.2 describes the application of the location-based model in the prediction of the user's next visited location. Experiments have been performed on the continuous dataset presented in section 2.3.2, and the results are reported in section 6.2.1. The time-location based mobility model introduced in section 5.4.2 is analyzed in more details here, and applied to the prediction of the user's location (section 6.3). The model has been evaluated on the GeoLife trajectories dataset presented in section 2.3.1, and the results are described in section 6.3.1; finally, section 6.3.2 discuss some open issues and the future work.

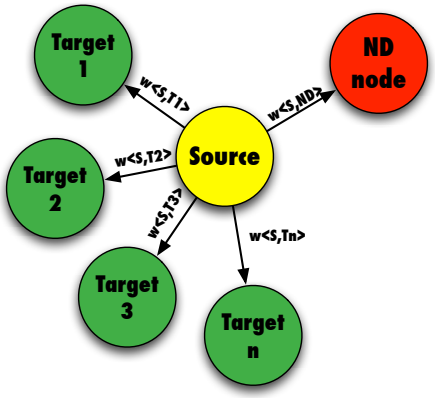


Figure 6.1. Example of Graph's Section

6.2 Location based model

A preliminary version of the SLS include the location-based model (introduced in section 5.4.1) to make predictions about the user's movements by means of the UPG. To predict the destination of the next movement the SLS calculates the conditional probability:

$$\mathbb{P}\left(\textit{moving to node } j \mid \textit{current node } i\right) \tag{6.1}$$

which is the probability of moving to *node j*, given the current one is *node i*. The value of this conditional probability is driven by the probability of the arc connecting *node i* to *node j*. The absence of an arc connecting two nodes, is equivalent to having an arc with probability equal to zero. Since the probability of each arc is calculated accordingly to the formula of equation (5.4), the sum of the probabilities of the outgoing arcs from a *node i*, is equal to 1.

Referring to the example in Figure 6.1, a generic source node has *n+1* outgoing arcs:

- *n arcs* directed to known PoIs (e.g., locations already visited by the user);
- *1 arc* directed to a *New Destination* node (*ND node*) representing locations which are still unknown for the user.

The weight associated to the arc which connects the current node to its *ND node*, is related to the transitions done by the user in the history of her/his movements, from this current node to unknowns nodes. Hence, each node has a different value associated to its outgoing arc to the *ND node*.

The prediction about the next movement, at this phase of the work, do not include any timing feature (e.g., time of the day in which the user is moving from one PoI to an other). Clearly, this penalizes the prediction, since time information is relevant to discriminate among PoIs with similar probabilities. Hence I introduced a new prediction model based on location and time in a second phase of the SLS development (section 5.4.2).

6.2.1 Experiment: next movement prediction

I used the *continuous dataset* presented in 2.3.2 to validate the performances of the proposed prediction method and in particular I focused on how the system can learn information about the user and about her/his frequently visited locations, to be able to perform personalized movement predictions. The localization service of the SLS collects data for 24 hours, and more precisely form 00:00 AM until 24:00 PM of each day, and, at the end of the day, it processes the data. Since I am now using for the current analysis the *continuous dataset* presented before, I divide the data accordingly in time interval of 24 hours, and I use the Density Based clustering algorithm (DBScan of WEKA) for identifying the PoIs (section 5.3.1) and construct the daily TPG . I empirically chose the value to apply to the parameters of the DBSCAN algorithm: *i)* ε (the radius distance of a neighborhood), *ii)* $minPoints$ (minimum number of points inside a neighborhood such that it can be considered a cluster). The value of the ϵ parameter is:

$$\varepsilon = 0.001DD \tag{6.2}$$

where DD stands for *decimal degrees*. Since, for privacy reasons, each user involved in the collection of the *continuous dataset* has been given the possibility to pause the localization service arbitrarily, the data collected is not always a 24 hours continuous data, but presents some discontinuities. The lack of continuous data has an impact on the choice of the value to assign to the *minPoint* DBSCAN parameter. To face this problem, I empirically defined some thresholds in the quantity of samples collected during a time period of 24 hours, and associated them a value for $minPoints$ accordingly (these values are reported in table 6.1).

Performing a clustering analysis on location data without introducing any temporal feature, may result in identifying also small clusters for areas visited frequently by the user, but every time for a short temporal interval. For example, if a user always drive on the same road from home to work and backward, and she/he frequently incurs in a point of the road characterized by traffic jam, that

num samples per day	corresponding δt	minPoint
(720-1440]	(12-24] hours	10
(360-720]	(6-12] hours	6
(180-360]	(3-6] hours	4
(0-180]	(0-3] hours	3

Table 6.1. MinPoints values

point may become a cluster[1]. During a first phase of the study [Papandrea and Giordano, 2012] I introduced an additional parameter to the clustering results: a threshold value for a **posterior filter**, to filter out irrelevant clusters identified by DBScan (section 5.5.3). In the current phase of the work, I face this problem by introducing a *temporal feature* in a *post clustering analysis*.

I use this new feature on the results of the clustering run over the data collected during a 24 hours time period. These results are a set of uncorrelated PoIs, essentially the vertexes of the user temporal prediction graph (TPG). Applying this new feature consists in adding edges to this graph by reading the data once again in the order it has been sampled (temporal order). This way the system reads sequences of consecutive location points belonging to the same clusters.

Definition 4. *When the user is in a generic cluster A, the system identifies a transition from cluster A to a cluster B, if and only if, it reads* $minPoints$[2] *consecutive samples associated to cluster B.*

Finally, the graph is further pruned of the vertexes without incoming and outgoing arcs. At the end of the 24 hours time period, once the related TPG has been built, it is used to update the global UPG with the information about the user's last movements history. Figure 6.2 shows an example of UPG for a User A of the *continuous dataset*, after 20 days of data location sampling.

As indicated in Section 5.4.1, the model compares the PoIs of TPG with the vertexes of UPG in terms of their euclidean distance and:

- if a PoI_{tmp} do not correspond to any PoI_{UPG}, it is added to the UPG, otherwise

[1]The location based model does not assign any semantic meaning to locations, hence a PoI may be situated wherever a user spends a relevant amount of time.

[2]The $minPoints$ value is the same used for the clustering procedure performed in that set of location data.

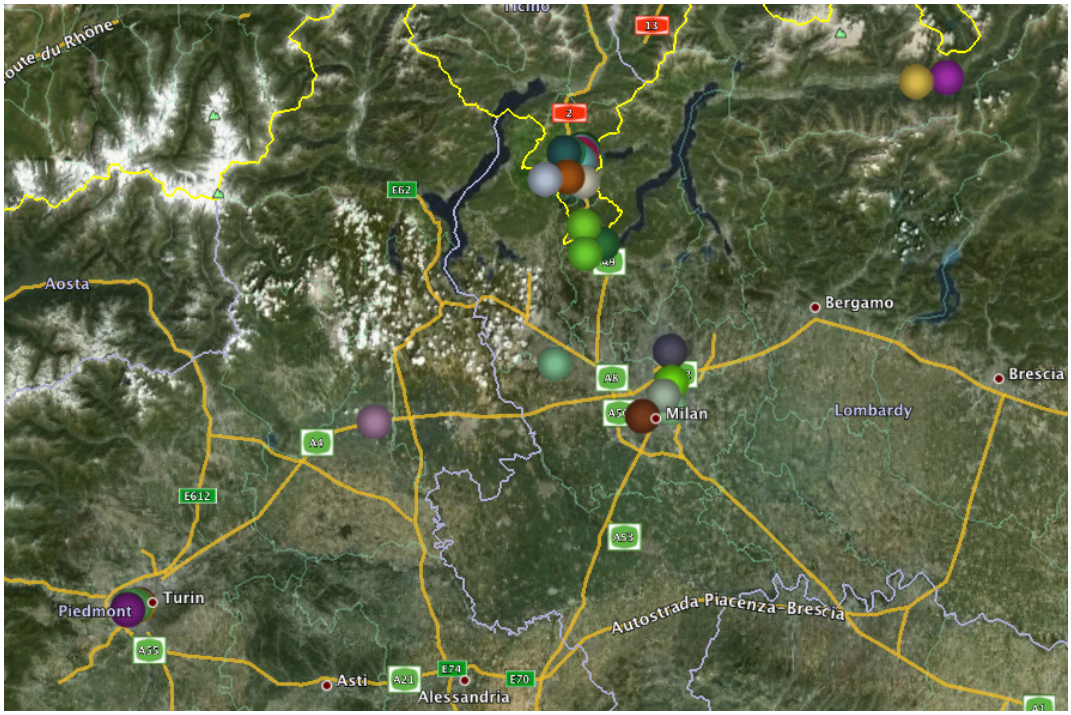


Figure 6.2. UPG's vertexes of User A, after 20 days

- it is merged to the corresponding PoI_{UPG}, updating its related values (latitude and longitude of the centroid, spatial extension, weight and density).

Figure 6.3 shows the growing cardinality of the UPG for different users with time, for the whole duration of the dataset collection experiment. It is visible from the figure that the number of PoIs in the graph increases considerably during the weekends or holidays, while it remains almost stable during working days. The overall number of PoIs for each user tends to grow infinitely with time. However, the number of real relevant PoIs remains limited: this is justified by the fact that the user tends to always visit new places, but only a few number of them are really relevant locations which the user visit regularly ([Khetarpaul et al., 2011; Papandrea and Giordano, 2012; Papandrea et al., 2013]). In a second phase of my study I faced the need to keep the number of PoIs per graph below a certain threshold, applying a criterion to prune the graph from the not relevant nodes. Looking at the weights assigned to the vertexes of the UPG, only few of them have an high value, while most of them are locations visited sporadically, hence with a small relevance value. A final version of the SLS will implement a pruning methodology which aims at removing the LILs from the mobility model data structure, to reduce the storage resource usage. A snapshot

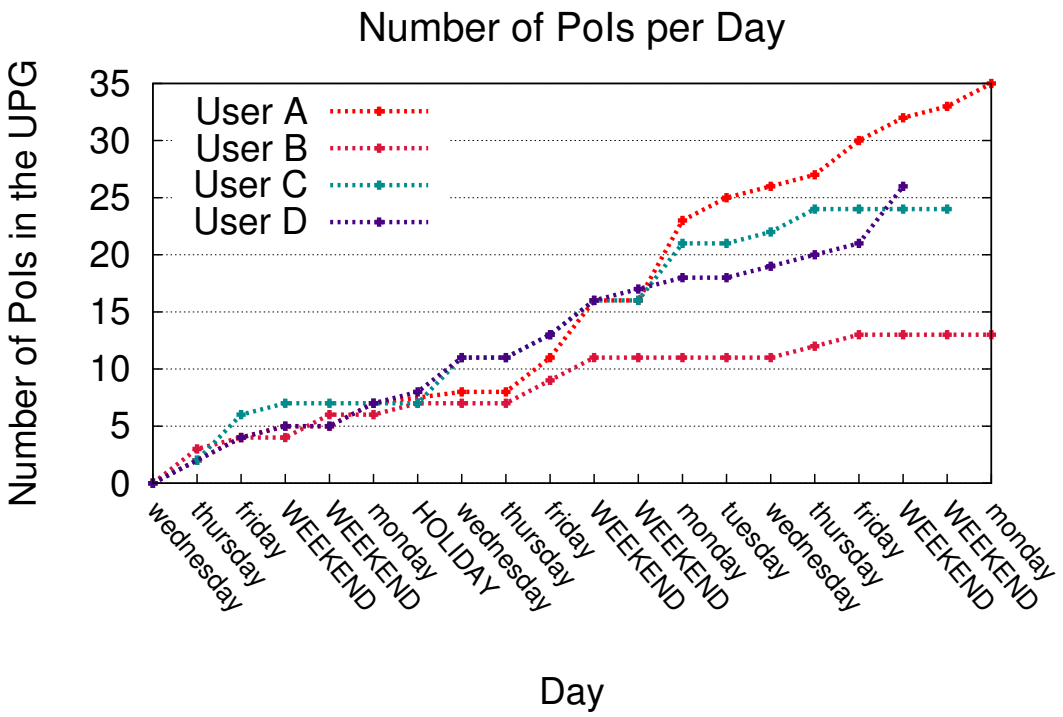


Figure 6.3. UPG cardinality increasing with time (User A)

of the UPG for *User A* after 10 days of sampling, is represented in figure 6.4.

At each k^{th} *day*, the system performs predictions about the target locations of the user's movements, using the related UPG updated at the end of the $(k-1)^{th}$ *day*. The predictions are made using the information learned and stored in the UPG. As expected, when the user moves toward unknown locations, the system is unable to perform correct predictions. For this reason, the SLS allows standard localization techniques for such situations. Also, as the learning phase never stops and the UPG continues being updated with newly learned data, this effect gets reduced with time.

I evaluated the error rate of the prediction procedure by measuring the number of errors over the number of predictions made by the *location based model*, when the user is moving from a known PoI, that is a vertex of the UPG.

In this experiment, I decided to give the same relevance to all the samples while updating the global probabilities, thus to use $\alpha = 1/n$. In fact, the factor α is basically used by the update algorithm to give different weights to the probabilities associated to the arcs of the graph, according to their age. However, the *continuous dataset* has been collected within a total duration of only

20 days, which is a quite short tracking duration to differentiate between old and new data, hence we decided to give the same weight impact to all the arc probabilities.

Figure 6.5 shows the prediction error for a sample user, User A, in each stage of the UPG growth. The number of predictions measured during the experiment and reported in the figure refers to the number of *predictable transactions*, going from a known PoI (a node already present in the UPG) to any other node; the number of errors is instead the number of wrong predictions. The prediction performed in this phase of the work is affected by a lack of information: in the prediction we do not consider the time when the user moves between two PoIs. In numerous situations the system has to make predictions about the next PoI visited by the user, where all the outgoing arcs from the current PoI have the same weight (such a situation is visible in figure 6.4 for *PoI 0* which is connected to *PoI 3* and *PoI 1* with the same weight): time information, in these situations, is

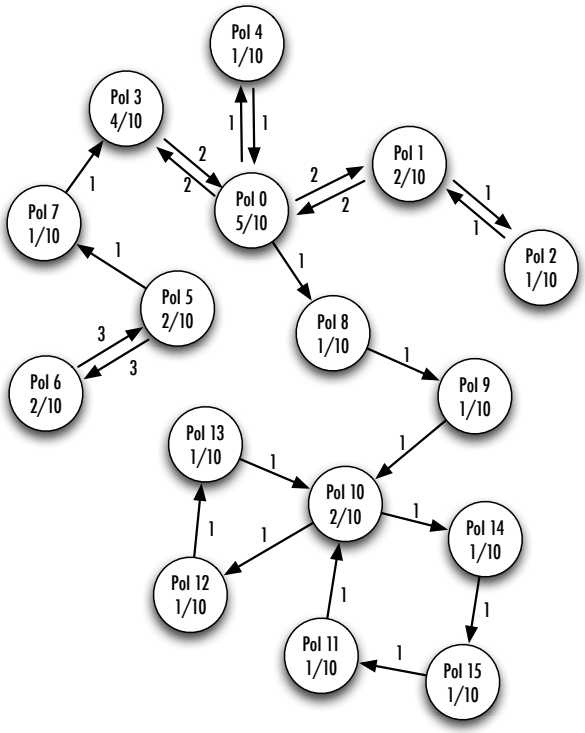


Figure 6.4. UPG of User A: snapshot after 10 sampling days. On each arc it is reported the number of crossings over it and in each vertex the number of days it has been visited, over the total number of days since the beginning of the learning procedure.

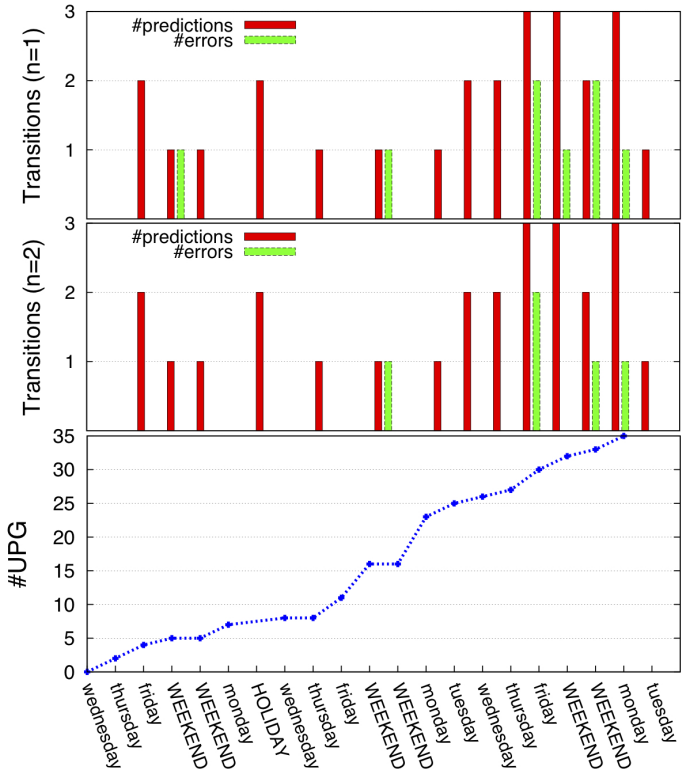


Figure 6.5. Prediction Error for User A

crucial to avoid prediction errors, because the outgoing arcs may have the same weight but in different times of the day (or of the week, month, year). For this reason in the second phase of my work I introduce a *timing feature* (section 6.3). In the error evaluation, I consider this factor by increasing the *confidence interval* of the prediction. If the user is situated in *PoI S*, the prediction of the next visited PoI is among the n reachable PoIs with the higher connecting arc weights where n is the *prediction confidence interval*. Figure 6.5 shows the results (related to User A in figure 6.3) for n equal to 1 and for n equal to 2 respectively. It is clearly visible how the error greatly decreases by increasing the confidence interval: this indicates that, by introducing the time information to situations in which a PoI has multiple outgoing arcs with the same weight, the error in the prediction might decrease accordingly.

Figure 6.6 instead shows the prediction and error rate for a different user (User D), whose movements are more regular and more repetitive than the ones already seen for User A. In fact, as we can notice from the figure, the number of vertexes of the UPG is smaller than the one of the previous user, and the system is able to perform more predictions since the movements are mostly among the

same set of regularly visited nodes.

As expected, for both users the number of predictions increases with time. This is an evidence of the fact that the system learns about the user habits and becomes able, with time, to predict the movements among learned PoIs.

The quantitative measure of the gain in terms of battery consumption is presented in the following chapter (section 7). I can however give a qualitative evaluation of the advantage in using the prediction within the SLS.

As an example, I report in table 6.2 the percentages of predictable transitions over the total number of user transitions (*recall*) for both User A and User D [3]: it is clearly visible that the number of predictable transitions for both users increases in the second half of the experiment, with respect to the first one, even if we have a limited amount of data. In accordance with what stated above, for User D the percentage of predictability is higher with respect to User A, because of the repetitiveness and regularity of his movements. User A instead is a great

[3]Similar or better percentage were found for all the users.

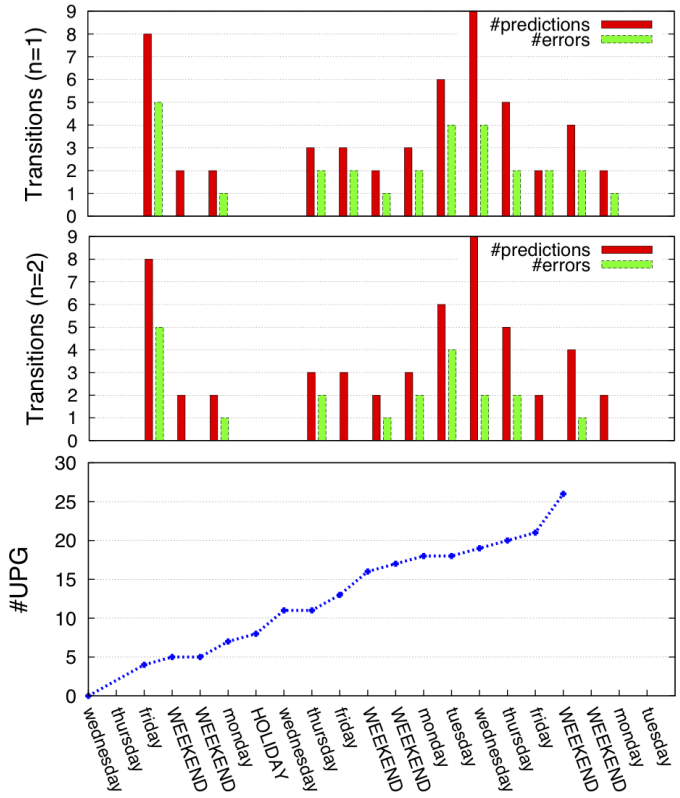


Figure 6.6. Prediction Error for User D

example of *globe trotter*, introduces above in section 5.5.4.

Considering that the prediction procedure requires a limited number of accesses to the data stored locally in the UPG, while the localization service requires the usage of expensive technologies, such as GPS, WiFi and GSM, and considering also the high percentage of predictable transactions (which do not require the usage of expensive technologies) reported in table 6.2, I infer that the advantage in battery consumption is reduced of several order of magnitude.

	First half of the experiment	**Second half of the experiment**
User A	37%	49%
User D	47%	63%

Table 6.2. Transition Predictability: Recall

Table 6.3 shows the prediction error percentage values over the total number of predictions (*precision*) for the same couple of users. The prediction error percentage has a different behavior for the two considered users. For User A, whose movements are not regular, the error prediction percentage increases from the first, to the second half of the experiment: this is due to the lack of regularity in the user movements and to the increasing number of PoIs in her/his UPG. The system frequently has to perform predictions in situation where all the PoIs reachable by the current PoI are connected by arcs with the same weights. In the case of User D instead, since his movements are more regular, the prediction error percentage decreases with time.

	First half of the experiment	**Second half of the experiment**
User A	11%	18%
User D	32%	25%

Table 6.3. Prediction Error: Precision

This analysis also shows that a generic user tends to move among a limited number of nodes, confirming the study described in section 5.5: even if the number of PoIs of a UPG increases with time, for most of the time her/his movements are restricted to a very limited set of PoIs. This is also visible in figure 6.4 where after 10 days of learning, User A visited 16 different PoIs: 10 of them have been visited for only 1 day, 4 of them have been visited for 2 days, and only 2 nodes have been visited for more than 3 days.

With this analysis I also show that the error performed by the prediction procedure decreases with time, while the system learns about the user's behavior. However, even after an initial learning phase there will be some prediction errors which are mainly caused by two factors: *(i)* I am not including the timing factor which is a discriminant in many situations, and *(ii)* the user can always visit new locations.

In this last section I assumed that the localization data is correct (the localization procedure performed by the system retrieves the correct locations), and accordingly, that the clustering procedure identifies the real relevant PoIs for each user. To evaluate the localization procedure I decided to ask for a feedback from the users which participated to the *continuous dataset* collection phase (in fact the dataset has been collected locally, by people working or studying at the university SUPSI). For privacy reasons we cannot associate a semantic meaning to the PoIs to evaluate their relevance, and nevertheless the semantic meaning of a certain location cannot give us a proof of its relevance for a certain user. The applied approach is to ask to each user to confirm or not the relevance of her/his PoIs identified by the system. On average, on the total number of users involved in the experiment, more than 90% of the PoIs identified for each user is actually a relevant location; the remaining PoIs have been identified as locations where the user spent a significant amount of time, but which cannot be considered relevant ones (i.e., an highway service area).

However it is difficult to calculate the localization accuracy in terms of the distance between a "true" location and the one identified by the system, because of its spatial extension. For example, if a person which uses to go frequently to the stadium to watch soccer matches, and every time she/he takes a different spot to sit, the system will identify a relevant cluster associated to the stadium, whose centroid is somewhere in the middle of the field. The calculated centroid of the stadium PoI cannot be compared to a real exact location, because the actual relevant location includes the whole stadium. This justifies our evaluation methodology based on user's feedbacks.

6.3 Location and Time based model

In this section I introduce the *timing feature* to the already presented user's mobility model. The learning procedure of this model has been presented above in section 5.4.2. Summarizing, each user keeps track of her/his history by updating a *Prediction Structure* with the actions data learned every 24 hours. This structure is then used to perform predictions. The SLS stores information about

the visited PoIs into a set of *Already Visited Clusters* (AVC) and when a new PoI is visited by the user, it is added to the AVC set.

The Prediction Structure is composed by a set of histograms (one per day) for each PoI, representing the distribution of the PoIs visiting probability during the days of the week. An example is reported in figure 7.2.

At each k^{th} *day*, the algorithm performs predictions about the user's current location, using the *Prediction Structure* updated at the end of the $(k-1)^{th}$ *day*. The system performs predictions about the current location of the user in two steps.

1 Firstly it checks the probability of each known PoI for the $day_{current}$ of the week, and the $dt_{current} \pm delayInterval$ of the day. The $delayInterval$ parameter is a time interval which let the algorithm tolerate shortly delayed predictions (i.e., action predicted few minutes after the real action occurred, $dt_{current} + 1$), or shortly early predictions (i.e., action predicted few minutes before the real action occurred, $dt_{current} - 1$). The n most probable PoIs for the current day and time are considered for the prediction. More details are described in section 6.3.1.

2 If the history of the user does not include yet any information about the habitual user's actions during $< day_{current}, dt_{current} \pm delayInterval >$, the system checks the probability of each known PoIs for every other days of the week day_i during the same interval of time $< day_i, dt_{current} \pm delayInterval >$. The daily most probable PoIs are selected, but only the m most frequent ones among them are considered for the prediction (details about the m parameter are explained in section 6.3.1).

6.3.1 Experiments and Results

The SLS system is thought to learn from the repetitiveness of the user's behavior in order to be able to predict habitual actions. Thus, to evaluate its performances, I need a dataset of regular and continuous location samples for a reasonable number of users. Similarly, the level of precision necessary for the evaluation is higher than the one we can obtain with traditional WiFi or GSM technologies. Unfortunately, for most of the available location datasets, the data corresponds to trajectories, or to WiFi or GSM continuous sampled data.

Therefore, I studied the whole *GeoLife dataset* presented in section 2.3.1 and selected a limited number of users whose related data presents only few discontinuities. For the selected subset it is easy to recognize the regularity of the user movements among a limited number of relevant locations.

The performances of the system are then measured on the basis of how well it can learn information about the user, and how fast it becomes able to perform personalized user location predictions.

In order to evaluate the system, firstly I calculated empirically all the parameters that take part in the algorithms introduced above (in section 5.4.2 and 6.3).

- The values assigned to the DBScan clustering algorithm parameters (described in section 5.3.1) are:

$$\varepsilon = 0.001DD \tag{6.3}$$

$$minPoints = 10 \tag{6.4}$$

  For both the DBScan parameters, the values have been chosen empirically considering not only the *GeoLife trajectory dataset*, but also the *continuous dataset* presented in section 2.3.2 (Papandrea and Giordano [2012]).

- After the clustering algorithm has been applied to the sample data, the system runs a *Temporal post-Filtering* procedure (section 5.4.2) to distinguish relevant PoIs from the complete set of identified clusters. The empirical value assigned to the *Temporal Threshold* parameter, applied for the filtering, is:

$$TempTh = 10minutes \tag{6.5}$$

- The *unit of time* value, which is used to convert the clock timestamp, to identify a certain interval of time inside the day duration, corresponds to:

$$dt = 10minutes \tag{6.6}$$

- For the learning algorithm, the value which characterizes the importance the system gives to the new data with respect to the historical data is chosen to be:

$$\alpha = 0.25 \tag{6.7}$$

  Table 6.4 shows different values of the location prediction error rate (formula 6.8) for different values of α, for a particular user ($User_A$). For all the values of α, the predictability rate (formula 6.9) is constant ad equal to 88.96%. As clearly visible from the values reported in Table 6.4, the error rate increases in two cases: when we assign a very large weight to the

$(1-\alpha)$	**error rate%**
0.95	13.69
0.90	13.69
0.85	13.42
0.80	13.37
0.75	12.05
0.70	12.32
0.65	12.33

Table 6.4. $User_A$: Prediction Error rate for different values of α

history, and oppositely also when we give the same weight to the history and to the current data. The best value of α in general is in between these two opposite situations. This is the reason which lead us to the assigned value of α. The *Prediction Error* (error rate) values which appear on table 6.4, corresponds to the rate of the wrong location predictions over the total number of predictions performed by the system.

$$PredictionError = \frac{number\ of\ prediction\ errors}{total\ number\ of\ predictions} \tag{6.8}$$

The empirical choice of the system parameters is also performed on the basis of its *Predictability Rate*, which corresponds to the number of predictable actions over the total number of actions performed by the user. The system considers unpredictable, the actions performed when the user moves toward unknown locations or when the movements are to known locations, during intervals of time when those locations have never been visited in the past.

$$Predictability = \frac{number\ of\ predictable\ actions}{total\ number\ of\ actions} \tag{6.9}$$

- Table 6.5 reports the values of the *Error Rate* by applying different values for n and m to the prediction algorithm. The predictability value does not change while changing the algorithm's parameters, and is equal to 88.96%, as already stated before. The values I show in the table are basically the confidence intervals, and in particular: n is the dimension of the set of most probable PoIs for the current day and time which are considered for the prediction; and m is the dimension of the weekly most probable PoIs for the current time which are considered for the prediction,

n	m	**error rate%**
1	1	24.18
1	2	16.80
1	3	14.67
2	1	19.43
2	2	13.95
2	3	12.05

Table 6.5. $User_A$: prediction error rate for different values of m and n, where $\alpha = 0.25$

when the system still did not learn information about the current day. The values used during this study are:

$$n\text{=}2,\ m\text{=}3 \tag{6.10}$$

- The interval of time corresponding to the acceptable delay or early timing for the activity prediction, corresponds to:

$$delayInterval = 2 * dt \text{ minutes (20 minutes)} \tag{6.11}$$

I evaluated the performances of the learning and prediction procedures by measuring the *Error rate* and the *Predictability* over the total number of actions performed daily by the user.

Figure 6.7-up shows the per-day-results of the learning and prediction algorithms run over the data collected by a user (*UserA*). The figure represents the number of dt over which the user collected locations data and the learning algorithm identified actual actions, which correspond to visits to relevant PoIs (blue color in the figure). Against this number we compare the number of dt related to predictable actions (green color). The red color on the graph represents the number of dt corresponding to wrong predictions. The lower part of figure 6.7-down represents the cumulative number of relevant PoIs, known by the user until the corresponding day.

Figures 6.8, 6.9 and 6.10 show the number of Prediction Errors over the number of Predictable Actions for other users in the dataset (UserB, UserC, UserD).

The GeoLife dataset over which I run the experiments, as already mentioned, has been firstly filtered such that the data selected as appropriate for the evaluation of the system is similar to a continuous sampling dataset. This selection has

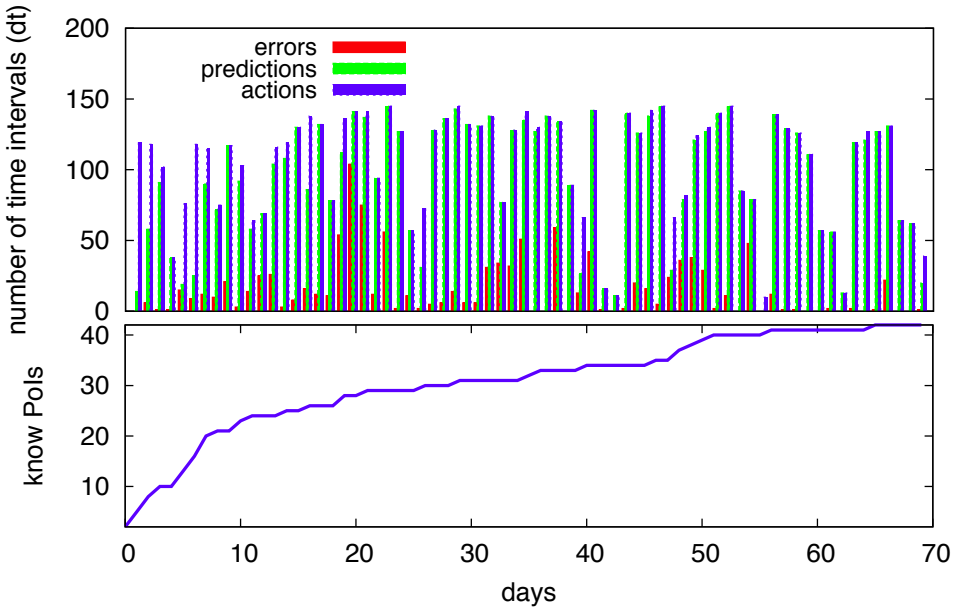


Figure 6.7. Prediction Error and Predictability Rate for User A (α=0.25, n=2, n=3)

been done on the users: only a small set of them has been considered, despite the large number of the participants in the collection of this dataset.

To summarize, by running the *prediction model* over the data related to the selected users, I applied the following values to the parameters.

$$\{n, m, \alpha\} = \{2, 3, 0.25\}$$

The corresponding error rate, averaged over all the users is:

$$\textit{Error Rate}=17.3\%$$

$$\textit{Standard Deviation}=5.1\%$$

The predictability rate best value for the available data is:

$$\textit{Predictability Rate}=88.96\%$$

The smallest prediction error rate I measured over the whole dataset, is the one related to the *UserA*, whose collected traces are very similar to a continuous sampled dataset. For this and other users I can easily recognize the regularity of their daily activities and then correctly perform action predictions.

However, in some situations, the system is still not able to predict the actions of the user. This may occur because of the dataset's lack of information

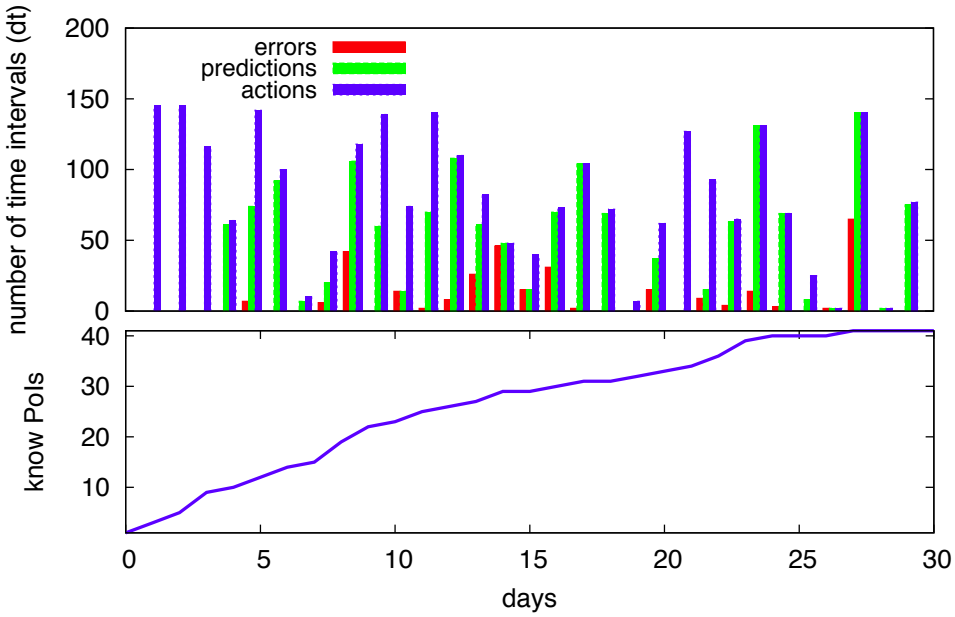


Figure 6.8. Prediction Error and Predictability Rate for User B (α=0.25, n=2, n=3)

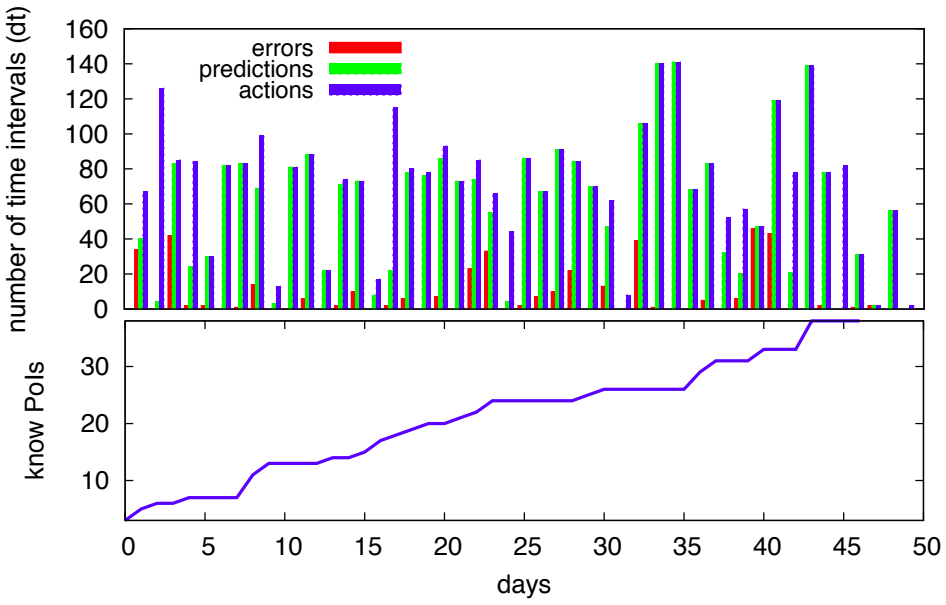


Figure 6.9. Prediction Error and Predictability Rate for User C (α=0.25, n=2, n=3)

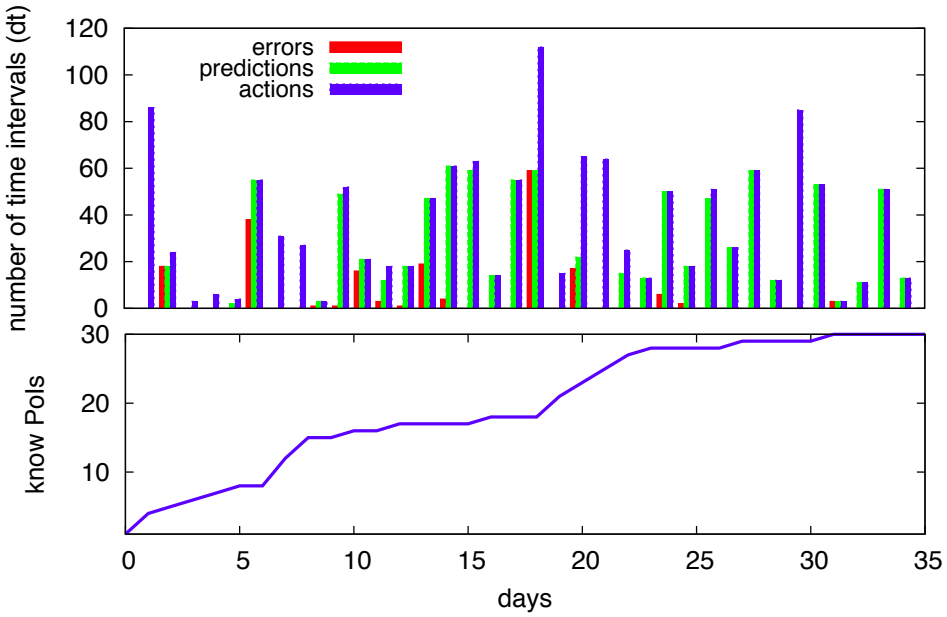


Figure 6.10. Prediction Error and Predictability Rate for User D (α=0.25, n=2, n=3)

and because of the presence of numerous data gaps. In fact, gaps and missing information affect the system, which then is not able to learn complete user's habits. Therefore, the system can fail to predict correctly the current activity of the user.

6.3.2 Discussion

History Offset

The prediction, as presented above, is affected by the absence of *near history* information. This is a source of error as, while performing predictions, the algorithm uses information learned until the day before, but does not take into account what happened the same day, or better, few minutes before the prediction. For example, if the algorithm has to predict where the user is at time 10.01.00 a.m. the *offset* of the history learned for prediction is exactly the time of the day (10 hours and 1 minute in this example). Considering a near history consists in setting the offset to a fixed and shorter value (i.e., 30 minutes), hence the algorithm can adapt its prediction to the very last activities of the user.

A solution to this issue is the implementation of an on-line algorithm which processes the data as soon as it is sampled and updates the prediction structure

immediately.

However, the final version of the SLS overcome this problem by adding a location context to the prediction. The idea is that it performs the prediction of the next location, knowing which is the current one.

Activity Periodicity

The prediction algorithm also considers the possibility to have user actions with different periodicity. A common action might be repeated with a period different from one week[4]. For example, the user might visit a PoI_A every 10 days. Furthermore, the periodicity of the user's activities might also correspond to multiples of the Prediction Structure duration (i.e., actions repeated every 3 weeks). In order to manage this issues, the system introduces two parameters:

- m: the m locations most frequently visited during all the week-days and at the same interval of time, are also considered in the prediction;
- n: defines the dimension n of the set of locations which are most probably visited during the current day of the week (for all the weeks in the user's history) and the current interval of time.

Acceptable Delay Interval

The regularity of the user's movements still could contain a factor of unpredictability: for example, a user which starts working every day at 8:00 a.m., may be sometimes delayed by the road traffic, and may arrive at her/his work place during the interval of time [7:40-8:20] a.m. Hence, while evaluating the error rate of the prediction algorithm, I introduce an other *tolerance temporal interval* ($delayInterval$), which corresponds to the amount of delay or advance in time which is still considered acceptable for the prediction.

User's Prediction and Predictability Rate graphs

As it is visible from the figure related to $User_B$ (figure 6.8), as well as for $User_A$, $User_C$ and $User_D$ (respectively, figures 6.7, 6.9, 6.10), at the beginning of the sampling the algorithm is mostly learning and is not able to perform correct predictions. At the beginning of the prediction phase, the error rate has a high

[4]The implemented user's Prediction Structure duration corresponds to 7 days, from Sunday to Saturday

value and it decreases with time, while the system learns about the user's behavior. The cases where we see a larger error rate and/or a lower predictability factor, mostly corresponds to an increment on the number of PoIs known by the user (visible step in the lower part of the figures); in the other cases, the errors or unpredictability is caused by the arbitrary of the users which can always visit new locations, or known locations with a timing different from its regular behavior. As expected, for all the users also the number of predictions increases with time. This is an evidence of the fact that the system learns about the user habits and become able, with time, to predict the movements among already known nodes.

6.4 Conclusions

The proposed prediction system for mobile users, might be applied to many application scenarios. Our main scenario is the optimization of a mobile localization mechanism, the SLS system. The SLS is a smart localization solution for mobile phones, whose goal is to provide a continuous and ubiquitous localization service, reducing the resources utilized with respect to traditional localization techniques (e.s., GPS). By using the presented prediction system, the SLS prevents performing continuous tracking of the mobile user by querying expensive localization provides. This allows SLS to offer a continuous localization service by reducing the frequency of those expensive queries, and substituting them with location predictions, especially when the user performs a limited number of actions with a certain regularity. Therefore the SLS acquires a certain smartness by using the presented prediction algorithm, in the evaluation of the resources usage.

This presented location prediction algorithm can also be used as a standalone system, extracting it from the SLS. It can be used as a context prediction algorithm where, knowing the location in advance may help in performing smart and adaptive decisions at the level of the applications (i.e., personalized Apps), and at the level of the data traffic and network loading.

Chapter 7

SLS Validation

In the previous chapters of this thesis I described in detail and evaluated the Inference, Learning and Prediction modules of the SLS. In this chapter I will describe how each module is working together with each other in the Smart Localization Service, with the purpose of enhancing the usage of battery resources, while providing continuous localization.

This chapter is structured as follows. In section 7.1 I provide a summary of the system, explaining in details how it works and how it incorporates the three different modules. I explain then the Mobility Learning procedure implemented by the SLS and the Context based Prediction algorithm, respectively in section 7.1.1 and 7.1.2. I finally present a final experiment which has been executed to validate the SLS, and the results obtained in terms of location availability (section 7.2.1), mobile device lifetime (section 7.2.2), network traffic exchanged (section 7.2.3), impact of the prediction in the localization (section 7.2.4), accuracy of the activity and visit inference (section 7.2.5) and in the identification of the PoIs (section 7.2.6). In section 7.3 I present a quantitative comparative study of the SLS with respect to two main related works. At the end, in section 7.4 I provide the final conclusions on the presented work and the obtained results.

7.1 Presentation of the SLS

The main goal of the SLS is to provide a continuous localization service, adapting its behavior to the user's mobility style, hence reducing the energy consumption costs compared to a standard continuous localization service. To get updated location data, the SLS periodically performs some reasoning and executes a location tracking only when strictly necessary, however guaranteeing continuously updated information.

The reasoning performed by the SLS has been presented above in section 2.2.1. More in detail: it involves the *Inference Module*, and consists in inferring the user's activity. If the user is *not moving* the SLS will not perform the localization, otherwise it tries to get a prediction of the user's next location from the *Prediction Module*. If a prediction is not available, or instead if the system can predict the user's location but with a reliability which is below a given threshold, then a location tracking is performed.

To prevent wrong predictions , the SLS never performs more than 3 consecutive predictions without checking the actual location of the user. Every time it performs three consecutive predictions (one every reasoning period), if the user is still moving, it is forced to execute a real location tracking using the Android location providers.

When the SLS infers the user is not moving, it stops the location updates, unless the last location update is older than 2 minutes. In this case, it tries to get a location update for a maximum duration of 4 consecutive periods in which the user is not moving. After this interval of time, the SLS stops any location updates, independently of the time of the last retrieved location information. This is done to avoid that the system continuously waits for a location update when the user is visiting an area where both GPS and Network providers are unavailable (i.e., indoor, basement, etc). Algorithm 1 reports the pseudo-code of the SLS periodic reasoning.

```
The SLS infers the user's activity;
if The user is not moving and (Not moving for at least 5 consecutive periods
or last location update is not older than 2 minutes) then
    skip location updates;
else
    if prediction available and reliability >= P_th and number of consecutive
    predictions < 3 then
        update the current location with the predicted one;
    else
        perform location tracking;
    end
    change the reasoning period according to the current activity;
    change the location update period according to the current activity;
end
```

Algorithm 1: SLS Periodic Reasoning

The *location prediction reliability* is given by the probability that a user is visiting a known location (PoI), at a certain time of a given day of the week. As already explained in chapters 5 and 6, the users mobility behavior is learned by the SLS in terms of "regular movements among known PoI". If a user has an high probability to be in a certain PoI_i at the current time t of the day, and day d of the week, and the distance between this PoI_i and the previous user's location is below a give threshold[1] $dist_{th}$,

$$\begin{gathered}(P(PoI_i|t,d) >= P_{th}) \\ and \\ (distance(location_{i-1}, PoI_i) <= dist_{th})\end{gathered} \tag{7.1}$$

then the SLS will decide not to perform the location tracking, but to use the predicted PoI_i as the current user's location. The probability thresholds used for the reliability study are evaluated empirically and correspond to

$$P_{th} = 40\% \tag{7.2}$$

$$dist_{th} = 2000meters \tag{7.3}$$

The condition about the distance is verified to select a subset of PoIs among the known ones, which are nearby the current user's location. The SLS then selects, among them, the PoI with the highest probability, above the defined threshold P_{th}. It is important to notice at this point the difference between the prediction availability threshold (P_{th}) and the PoI $radius_{max}$ (maximum distance between coincident PoIs): these two distances are different in the sense that, the second one refers to the PoI spacial dimension, while the first one identifies how long in advance (in terms of distance) the SLS is able to predict the next visited PoI. Figure 7.1 shows an example scenario in which the user (whose location on the map is identified by a green dot) knows 9 not-coincident PoIs (already visited ones and stored into the prediction data structure): the distance between these PoIs' centroids is greater than the $radius_{max}$. The SLS, in order to predict the user's next visited PoI, selects the ones located within the $dist_{th}$ distance; and among them, it selects the PoI with the higher probability for the related day/dt.

The period at which the SLS performs the *reasoning* described above is not constant, and strictly depends on the current user's activity. Table 7.1 contains

[1]The SLS calculates distances between GPS points by using the *Haversine formula*

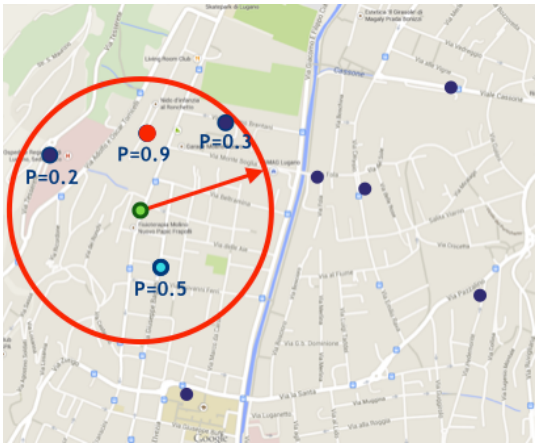


Figure 7.1. Prediction: example scenario

the values of the *adaptive reasoning periods* per each inferred activity. If the Inference Module cannot retrieve an activity or if the user is inferred to be moving with a *fast vehicle*, the SLS will perform the next reasoning after 60 seconds. If the activity inferred belongs to the class *slow vehicle* the consecutive reasoning is triggered after 90 seconds, while if it belongs to the class *moving by foot* it will start after 2 minutes. However, if the system infers the user is not moving, the reasoning period increases every time this activity is consecutively confirmed, starting from a first period of 2.5 minutes to a maximum value of 10 minutes, after the fourth consecutive time the system infers the user is not moving. The activity inference is then performed every 10 minutes until the user is moving again toward a new location.

Activity	**Period [sec]**
Not recognized	60
Fast vehicle	60
Slow vehicle	90
Moving by foot	120
Not moving	
1^{st} time	150
2^{nd} time	180
3^{rd} time	300
from the 4^{th} time	600

Table 7.1. Adaptive reasoning periods

As already described in section 4.6.5 the activity inference is performed by the SLS by means of a majority study on a set of consecutive windows of sampled data. Differently from my first experiments (for the training of the decision tree) the SLS considers 5 consecutive non-overlapping windows, corresponding to 5 seconds of accelerometer sampled data. This way the SLS is able to gather more information about the user's movements. The majority study may result in a "*Not recognized*" activity, in particular when there is no activity repeated for at least 3 sliding windows, among the inferred ones. In this case, the SLS performs its reasoning at its highest sampling frequency: which corresponds to a period of 60 seconds.

In chapter 5 I described the *Learning Module*, how it uses the location data collected to identify the user's PoIs and to learn how the user visits them. For this purpose, the experiments described in the previous chapters involved the usage of a Density Based clustering algorithm over the collected location data, to identify locations relevant to the user. In the final version of the SLS, the *Learning Module* implements a different and more lightweight algorithm to understand and learn the mobility behavior of the user. In fact, given the user's activity information provided by the *Inference Module*, the SLS is able to identify when the user is visiting a potential PoIs. In particular, the SLS identifies a visiting activity only when the user is not moving (the user is actually still or standing). The other classes of activities are identified as transitions among PoIs. Accordingly, the SLS stores location information of the user as *visits* data, only when the user is still or standing. Therefore the definition of PoI slightly changes with respect to the one given in section 5.3.

Definition 5. *A PoI is a location where the user spends an amount of time longer than a fixed threshold, performing very small or no movements.*

The threshold in time is:

$$T_{th} = 5minutes \tag{7.4}$$

If a user spends at least T_{th} time in a certain location without moving, the SLS considers this location area as a PoI, and the user's activity as a visit to this PoI.

7.1.1 Learning the user's mobility

The final version of the SLS implements the *location and time based learning algorithm* described in section 5.4.2. To avoid any impact on the battery consumption, it runs the *learning* algorithm every time the device is connected to the charger. The information which are learned by the SLS are

1. the set of PoIs known by the user;

2. the mobility behavior of the user among the known PoIs.

Algorithm 2 presents the pseudo code of the learning algorithm implemented by the SLS.

While running, the SLS collects two set of data:

1. timestamped GPS coordinates of the tracked and predicted locations;

2. timestamped activities of the user.

The *Learning Module* reads only once the activities data, in the order in which they have been inferred and it identifies the user's **visits**. A *Smoothing* algorithm is applied to the activities stream, which eliminates short (in terms of time) gaps between consecutive visits. If the *Inference module* infers an activity different from *not moving* between two visits, whose duration is shorter than a given threshold $dt_{th,GAP}$, than this activity is considered as *noise*: it is then discarded and the two adjacent visits are merged together. The value of this temporal threshold is:

$$dt_{th,GAP} = 90seconds \tag{7.5}$$

After all the user's visits have been identified in terms of *start time* and *end time*, the *Learning Module* proceeds with the identification of the related visited PoIs. Per each visit, it selects all the GPS coordinates collected during the visit duration and calculates their *centroid*. This calculation is very lightweight in terms of computational cost and usage of memory, in fact the number of samples is very limited: these are only collected during the first minutes of the visit; when the SLS understands the user is not moving, it stops the localization. The calculated centroid is compared with the already known PoIs. If there exist any already known PoI whose distance to the identified centroid is less than a given threshold $radius_{max}$, then the centroid and the selected known PoI are merged together. Otherwise the centroid is considered as a new relevant location for the user, and the list of known PoI is updated, adding the new one. The distance threshold between matching PoIs corresponds to:

$$radius_{max} = 150meters \tag{7.6}$$

```
Data: [start time, end time] of the learning period
select the GPS points collected during the interval [start time, end time];
select the activities identified during the interval [start time, end time];
identify the actual visits applying the smoothing algorithm;
/* selected list of visits <start time, end time>          */
for visit in visits list do
    select the GPS points collected during the visit;
    calculate the Centroid;
    if the Centroid is near to an already existing PoI then
        update the existing PoI;
    else
        insert a new PoI in the list of known ones;
    end
end
/* selected list of visits <start time, end time, PoI>     */
update the user mobility model data structure;
identify the list of Low Interest Locations;
if the list of LIL is not empty then
    eliminate them from the list of known PoIs and from the mobility
    model data structure;
end
delete the GPS points related to the learned interval;
delete the activities related to the learned interval;
```

Algorithm 2: Learning Algorithm applied to a limited interval of time $[start\ time, end\ time]$

At the end of the described procedure, the learning module retrieved the list of user's visits, with the related *start time*, *end time* and *visited PoI*, for the processed learning interval. This list is then used to update the user's mobility model (as already described in section 5.4.2), updating the user's mobility history data structure which is stored locally on the device. The dimension of this data structure is limited, in fact it contains the visit probability per each dt of the day (144 dt in a day), for each day of the week (1008 dt total in a week), per each known PoI, only if the related probability is greater than zero. A PoI visiting probability $P(PoI_i|day, dt)$ is greater than zero if and only if the PoI has been visited at least once, during the given day and time dt. As shown below, in the validation section (section 7.2.6), the number of PoIs is limited, and especially the number of locations in the classes of HIL and MIL does not

grow over a certain value (as already demonstrated in Papandrea et al. [2013]). Additionally, among these PoIs only few of them are visited for very long time intervals (HIL). However, only these PoIs visited for long intervals of time have a great impact on the dimension of the mobility history data structure. Storing only PoIs belonging to these classes of relevance, guarantees a limited usage of storage by the SLS.

After updating the stored user's mobility model, the *Learning Module* identifies the *Low Interest Locations* (section 5.5) among the known ones and eliminates them form the list of known PoIs, and form the mobility behavior data structure. The Low Interest Locations are identified by analyzing the PoIs visiting probabilities distributed during the week. And more in detail, a PoI is identified as a Low Interest Location if its highest visiting probability is lower than a give threshold $P(PoI)_{min}$, considering all the days of the week, and all the dt of the day. The value of this threshold is

$$P(PoI)_{min} = 0.08 \tag{7.7}$$

and corresponds to an interval of time of 4 weeks, without seeing the given PoI. If a PoI is visited once, and then it is not visited for a duration of four consecutive weeks, it is classified as LIL. The Low Interest Locations are not relevant for the purposes of the user's mobility modeling and prediction, in fact those are the locations which the user visits sporadically and without regularity. Hence, eliminating them from the mobility model supports the SLS in reducing the usage of storage resources without loosing important information about the mobility behavior of the user.

After the learning procedure, all the stored GPS coordinates and the activities data related to the processed time period are deleted, to avoid the used storage dimension to increase monotonically.

Tables 7.2 and 7.3 report a summary of all the parameters presented above, which have been empirically defined and used by the SLS Mobility Model learning and prediction algorithms.

7.1.2 Location and Time based Prediction

The SLS implements a context-based location prediction algorithm. It predicts the next user's location, given the current context in terms of time and location: "The user is at a certain location *(lat,lon)* at day d_j time t_i. Which is the most probable PoI that is going to be visited at time t_{i+dt} of the same day d_j?". To retrieve this information, the system first selects a subset of PoIs among the known

Parameter	Value	Definition
T_{th}	$5minutes$	minimum visiting time to a PoI
$dt_{th,GAP}$	$90seconds$	maximum noise time between adjacent visits
$radius_{max}$	$150meters$	maximum distance between coincident PoIs
$P(PoI)_{min}$	8%	maximum probability of a Low Interest Location
-	3	maximum number of consecutive predictions
-	$2minutes$	maximum delay for a location update

Table 7.2. Learning algorithm's empirical parameters

Parameter	Value	Definition
P_{th}	40%	minimum reliability probability for prediction
$dist_{th}$	$2000meters$	maximum reliability distance for prediction

Table 7.3. Prediction algorithm's empirical parameters

ones, which are the nearest to the user's location (*location-context evaluation*). Then it performs a *time-context evaluation* and checks the user's history, and in particular the regularity of the user's visits at day d_j of the week, time t_{i+1};

After this evaluation, the SLS decides whether it is able to perform a prediction, and eventually it predicts the next user location. The described context-based prediction supports the system toward its main goal of reducing the number of GPS tracking operations.

The location prediction service runs in parallel with the adaptive localization procedure. The Prediction Module evaluates the possibility to make a prediction periodically, and updates it every $dt = 10minutes$. At each interval dt the system checks the actual user's location and current activity, and updates accordingly its prediction for the next visited location. This data is provided to the SLS in the form of *candidate prediction* $< PoI, VisitingProbability >$. The SLS then utilizes this information, stopping the real location tracking, when appropriate (as described in section 7.1).

The SLS learning methodology and mobility model allows also a context-free location prediction. In table 7.4 I report the different levels of predictions which could be implemented by means of the SLS model. The last two rows of the table report the actual prediction performed by the SLS. The first two lines instead represent a context-free location prediction. In fact, the SLS mobility model allows a location/time user profiling, which identifies the most frequently visited locations and the time/day those are visited.

Figure 7.2 represents an example of the mobility history of a user (experi-

Information Provided	**Prediction**	**Question**
time/day	PoI_i	Which PoI will be most likely visited at a certain time/day?
PoI_i	time/day	When does the user usually visit PoI_i?
current time/day PoI_i	time/day	It is monday $10:00a.m.$, when will the user arrive at PoI_i?
current time/day current location *(lat,lon)*	time/day PoI_i	It is monday 10:00a.m. and the user is at location *(lat,lon)*, which PoI will the user visit next?
current time/day current location *(lat,lon)* PoI_i	time/day	It is monday 10:00a.m. and the user is at location *(lat,lon)*, is the user going to visit PoI_i soon?

Table 7.4. Possible Location Predictions

ment described in section 7.2), learned by the SLS. It shows a list of PoIs (one per each plot) and their related visiting probability distributions over a whole day. As it is visible from the figure, there are two PoIs characterized by a long daily-total visiting time: PoI_1 and PoI_3. Given their relevance calculated at the end of the experiment, they are classified as *High Interest Locations* (according tot heir visiting frequency). Additionally, their distributions probabilities reaches their maximum values at different times dt of the day: PoI_1 is mainly visited during the night, and PoI_2 is mostly visited during the day. A semantic inference for these PoIs given their visiting probability distribution is then straightforward: PoI_1 can be inferred as *home* location, while PoI_2 as *work* location[2].

Each PoI has a starting probability, in the user mobility data structure, equals to 0 (hence, basically, it is not present in the related database table). The first time a PoI is visited, its probability is updated to 0.25, according to the probability update formula 5.8 presented in section 5.4.2; with time, if the user visits this PoI regularly, the visiting probability increases monotonically and gets asymptotic to its maximum value. A PoI probability trend strictly depends on the regularity of the user in visiting it. In figure 7.3 I represent the visiting probability values of a PoI with time, for a specific day day_i and time dt_j, given different moving behavior of the user. Considering on the x-axis, instances of the same time dt_j for the same day day_i of consecutive weeks $week_k$, with $k = [1, \infty)$, on the y-axis I represent the evolution of the probability value for a given PoI_p:

[2]While writing this thesis I am currently working on this research direction, with the paper "*On Properties of Human Mobility*" under submission to *Computer Communications Journal*

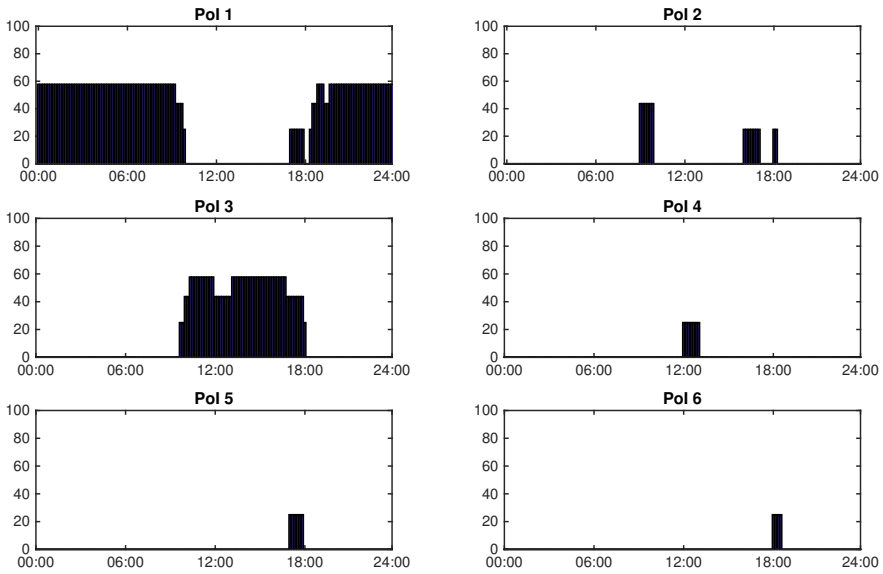
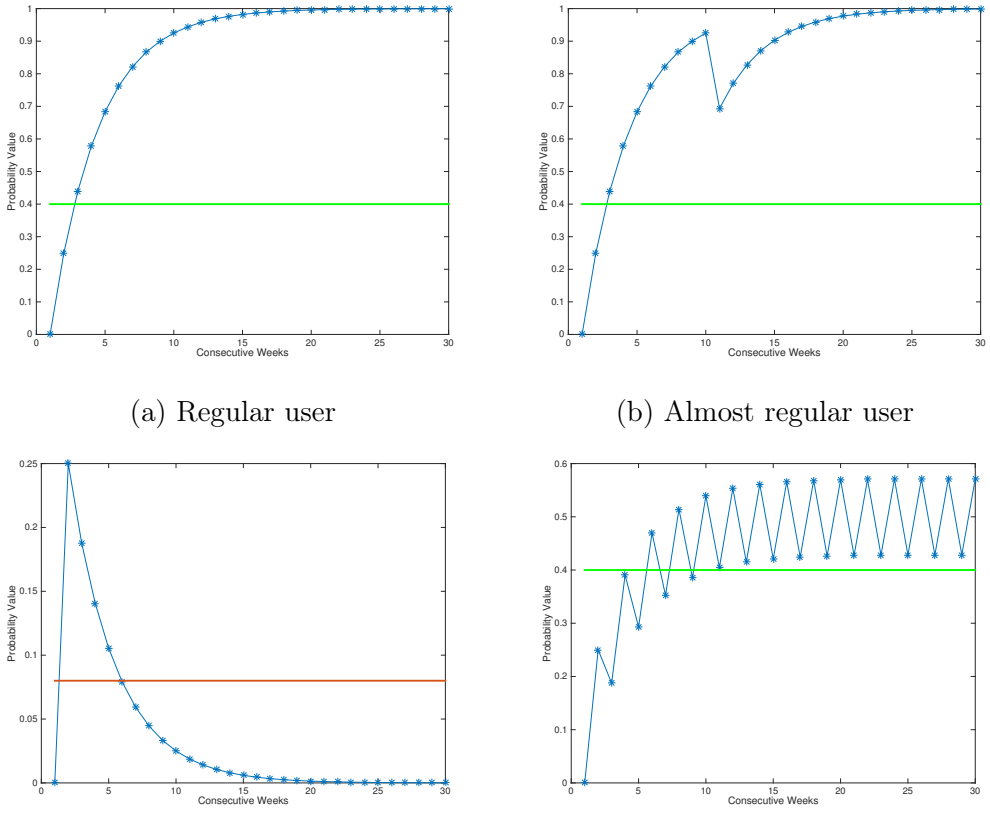


Figure 7.2. Probability distribution of the user mobility history for day 3 of the week (wednesday), after three weeks of learning.

$P(PoI_p|day_i, dt_j)$. Figure 7.3a represents the probability values for a very regular behavior of the user: when the user tends to visit regularly PoI_p every week, at the same time of the day $< day_i, dt_j >$. The green line shown when the SLS considers this probability reliable to make a prediction: the SLS could predict that the user is visiting PoI_p at time $< day_i, dt_j >$ if its probability values are above the green line. This PoI is then considered as a candidate for the prediction, before the location filtering (the distance from the current location to the candidate PoI) and the comparation with the other candidates. Figure 7.3b represents the probability values for a regular behavior of the user, who visits PoI_p every week, except one. When the user changes sporadically her/his regular behavior, the probability model is not affected heavily. The probability values provided by the model are above the reliability threshold (green line) and allow the SLS to make reliable predictions. If however, the user does not have a regular behavior in visiting a certain PoI, the probability distribution will not always allow the SLS to perform predictions. If the user visits a certain PoI only once (figure 7.3c), after a certain amount of time the SLS removes it from the user's mobility history, because not relevant for the mobility model. In the figure, the red line identifies the probability threshold for the identification of the Low Interest Locations: if a PoI has its highest probability value below this threshold,

it is considered as LIL and eliminates from the mobility model. This procedure prevents the list of relevant PoIs to increase infinitely, because of the number of LILs, and to keep it updated accounting new and past visiting events. Finally, figure 7.3d represents the probability values associated to a user performing regular visits to a PoI, but with a regularity period different than one week: the user is in fact visiting a PoI_p at time $< day_i, dt_j >$ every second week. As visible form the figure, after a certain number of visits (5 visits, with a 2 weeks period) the SLS starts considering the PoI as reliable per each $< day_i, dt_j >$, leaving the final decision to the distance selection.



(a) Regular user

(b) Almost regular user

(c) Low Interest Location

(d) PoI visited every second week

Figure 7.3. $P(PoI_P|day_i, dt_j)$ for consecutive weeks, given different mobility user's behaviour

7.1.3 SLS's working scenario

In order to better understand the SLS's working scenarios, I will present in this section an example use case. First of all, it is worth reminding here that SLS main goal is to provide a continuous localization service reducing the battery consumption with respect to the already existing solutions. The SLS does not aim at providing a highly accurate localization service, for example indoor localization with shop/room -level granularity. It instead provides a smart service which continuously track the user, utilizing the technologies already available on mobile phones.

Consider a scenario in which a user is running on his mobile phone a certain number of location-based applications. All these applications are usually interacting directly with the Operating System to get location information as soon as this is necessary: this implies that the applications always query directly the GPS or Network providers to retrieve location data. This is clearly energy expensive, and this is where the SLS applies his smartness. By using the SLS, the applications are not interacting anymore with the Operating System, but they query the SLS for up-to-date location information. The SLS in turn, by learning the movement habits of the user and analyzing his actual activity and context, is able to provide up-to-date location information without the necessity of a continuous tracking (then utilizing less energy). The SLS learns how the user is moving, which are the related PoIs where he spends most of his time, when he uses to visit them, and it calculates what the user is doing (activity performed). All these information are retrieved and learned by the SLS without the need of a remote computational support, hence without the need of sending private information out through the internet.

Hence, the SLS is a real-time user-specific smart location service, which is able to learn *who is the carrying user*: it does not only provide location information but it performs a user profiling and consequently adapt the performed localization procedure.

7.2 Validation

Each module of the SLS has been presented in details and evaluated independently in the previous chapters. In this section I will present the final experiment I carried out to validate the complete SLS system. The main contribution of my work consists in the smart usage of the battery resource while performing continuous localization with a mobile device. As I already explained in chap-

ter 3 there are many issues in the battery consumption evaluation, especially when the measuring scenario involves *mobility*. The are, in fact, many variables which may impact the life time of a mobile phone: for example, the device hardware, the operating system version, the activity of the users on the device (i.e., amount of phone calls, applications and services running on the phone), the active network interfaces (i.e., WiFi, Bluetooth, data connection), the intensity of the screen's light, the running services and application, etc. However for this final experiment, I analyzed the device battery consumption by measuring the battery discharge with time, while it was providing a continuous localization service. Furthermore, the devices used for this experiment are all the same: equal hardware and operating system version.

For this final experiment I used 3 Galaxy Nexus Google phones from Samsung, running the Android operating system version 4.2.1. Each mobile device involved in the experiment is set to the factory state and registered with a Google account (to be able to use the Google services for the localization); the WiFi interface is enabled (to support the localization) but never connected to any network and the Bluetooth is disabled (since it is not necessary for the experiment). Each phone is registered with the same, unique Google account: this way all the three phones transmit and receive a similar amount of data from the native Google services.

During this final experiment a user carried with him all the three above mentioned devices, for a duration of 22 days, while performing everyday activities. All the three devices are performing continuous localization, querying the GPS and Network providers by means of the Android Localization Manager. However, each of them implements a different localization methodology. More in detail:

- **Phone A** is running the SLS (as described above in this chapter) and starts learning about the carrying user at the beginning of the experiment;
- **Phone B** is running a Continuous Localization service without learning procedures, tracking the user's location every 60 seconds;
- **Phone C** is running a modified version of the SLS, which does not implement the activity inference procedure, instead it uses the Activity Recognition API introduced recently into the Google Play Services.

At the end of each day the user involved in the experiment was actively triggering the learning procedure, and a copy of each device's internal database was stored externally, to be able to perform the offline validation study. The results of

this experiment are strictly dependent on the user's activities and on his mobility behavior. However this validation methodology seems to be the most eligible to verify the goal fulfillment of the SLS.

The three phones are in the same state as described above (same device, operating system version, Google account registered, first usage after a factory state reset) and all of them start without knowing anything about the carrying user. To get updated location data, Phone B queries the Android location providers periodically. While Phone A and Phone C perform some reasoning periodically and queries the providers only when strictly necessary, however guaranteeing updated information. The reasoning performed by the SLS has been presented above in 2.2.1, and more in details in section 7.1. Together with these phones, the user carried with him a fourth device **Phone D** in order to collect the ground truth: an application running on this phone allowed the user to keep track of the performed activities (classes of activity) and the related starting and ending time; moreover, per each visit to a PoI, the user logged the actual location.

The performances of the phones involved in this validation experiment are measured in terms of six main parameters.

1. *Availability of the location information*: all the phones store locally the tracking location data, to be able to perform an offline evaluation of the location information availability [3]. This measure shows the capability of the SLS to provide continuous location information while performing not a continuous but an adaptive tracking of the mobile device.

2. *Mobile-phone life-time*: each device keeps track of the battery drain, storing the timestamp of the battery consumption with a granularity of 1%. This measure shows how the SLS improves the battery usage, extending the lifetime of a phone, while adapting its behavior to the user's activity and learning his mobility habits.

3. *Amount of network traffic exchanged*: each phone stores information about the total amount of data traffic exchanged (only for the evaluation purposes). With this measure I will compare the performances of the SLS, against the SLS modified version running the Google Activity Recognition, with respect to the amount of exchanged network traffic. The SLS does not need a support from a back-end server, hence it generates a reduced amount of network traffic which is usually expensive in terms of battery

[3] This is done only for the purpose of the off-line evaluation, in fact the SLS does not store the location information

consumption, and it does not transmit any sensitive information which could generate some privacy issues.

4. *Amount of predicted locations* against the number of visits to known PoIs. With this measure we have a quantitative evaluation of the impact of the prediction on the localization procedure.

5. *Accuracy of the Inference* performed to identify the user's activities and visits to PoIs. With this measure I am able to compare the performances of the SLS Inference Module against the Google Activity Recognition service.

6. *PoI identification*. With this measure I evaluate the capability of the SLS to identify the correct PoIs, comparing them with the collected Ground Truth. I also measure the growing number of PoIs with time, per class of relevance, in order to estimate the storage resources usage.

7.2.1 Availability of Location Information

The continuity of the location information is an important requirement for a tracking service. With this work I want to show that it is not necessary to perform a continuous localization procedure to ensure continuity in the information retrieved. The SLS in fact implements a smart usage of the devices embedded sensors, providing a continuous localization service which adapts its localization procedures frequency to the actual activity of the user.

In this section I will evaluate the continuity of the location information provided by the phones involved in the experiment. To perform this evaluation I will consider the time intervals in which the user is moving, according to the ground truth. In these time intervals I will verify the location information availability. During this evaluation I will not consider the intervals of time in which the user is not moving, because I assume during this time the user's location does not change. Figure 7.4 shows the percentage of time, over the total time during which the user was performing a moving activity for the whole experiment, in which a location has been stored in the internal database. To calculate this percentage I considered the total time divided in small time slots of 60 seconds, and I verified per each time interval, if the system stored at least one location data. As visible from the figure, Phone B which is running a continuous localization system, does not have a complete coverage. This is mainly caused by the unavailability of the localization android providers, especially in indoor environments. Considering PhoneB's percentage of time for the availability of the location information as our target (90%), the correspondent percentages

measured for PhoneA (SLS) and PhoneC (SLS+GoogleAR) have acceptable values. PhoneA provides a coverage of the location information which correspond to 82%, this means that for 8% of the time it is not able to provide a location information, compared to the location availability for Phone B (location information available for 91% of the target time). Phone C has a slightly lower value of location coverage which corresponds to 75%, 15% less than PhoneB (83% of the target time). We have however to consider that Phone A and Phone C are sampling the location with 1 minute sampling period only when the user is moving with a fast vehicle. When moving differently, the location update period gets longer and the location updates arrive to the system with a lower frequency. This way, the SLS ensures an always updated location information, but it does not guaranty one update per minute.

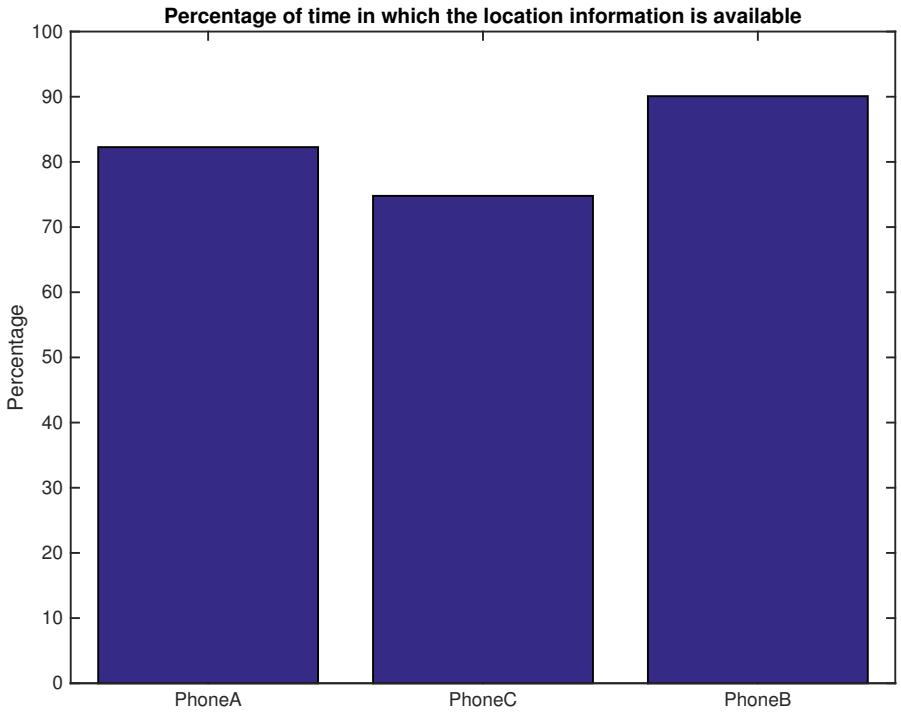


Figure 7.4. Availability of the location information

7.2.2 Mobile device life-time

There are several variables impacting the mobile phone life-time. During this experiment I tried to limit the number of free variables in this evaluation by using different instances of the same mobile device at its factory state, running the same version of the operating system, and registered with the same Google account. The mobility of the user heavily impact the device lifetime of Phone A and C, which are running two different versions of the SLS, both adapting their reasoning to the real user's activities. Phone A (running the SLS) and C (running the SLS with Google Activity Recognition) are increasing the location update frequency while moving fast (fast vehicle) and reducing it while moving slower (moving by foot), and finally stopping completely the user's tracking while not moving. This adaptive sampling increases the device lifetime when the user is mostly static; while it uses more battery resources when the user is constantly moving. Phone B, instead, implements a continuous localization service providing periodically a location update, independently from the user's mobility behavior.

Assuming the user is always on a fast vehicle, the SLS performs the reasoning and location tracking at its fastest rate (equal to the location tracking frequency of Phone B). It will however consume less battery than a continuous localization system (Phone B), because it performs, when possible, context based location predictions in place of direct locations tracking. Figure 7.5 represents the average lifetime of the three phones while running the final experiment. As visible from the plot, Phone A (SLS) has an average lifetime of 38.1 hours, with a standard deviation of 3.2 hours; Phone C (SLS+GoogleAR) has an average lifetime of 34.6 with a standard deviation of 2.2 hours; while Phone B (Continuous Localization) has an average lifetime of 18.6 with a standard deviation of 2.5 hours. This absolute measured life time strictly depends on the device (hardware) involved in the experiment. However, what is significant from this plot is the ratio between the average lifetime of each device. Phone A lifetime is more than twice the lifetime of Phone B: this shows the heavy impact of the location tracking on the battery resources. However, also the activity recognition performed by means of the Google services has a great impact on the battery consumption, and this is shown by the average lifetime of Phone C, which correspond to 10% less than Phone A lifetime.

To give an idea about the dependency of the lifetime to the mobility of the user, I report in figure 7.6 two examples of lifetime for Phone A and Phone C, considering two extreme cases of *user always still* and *user mostly moving* behaviors. In the graph I compare *Day A* which has the longest total daily moving

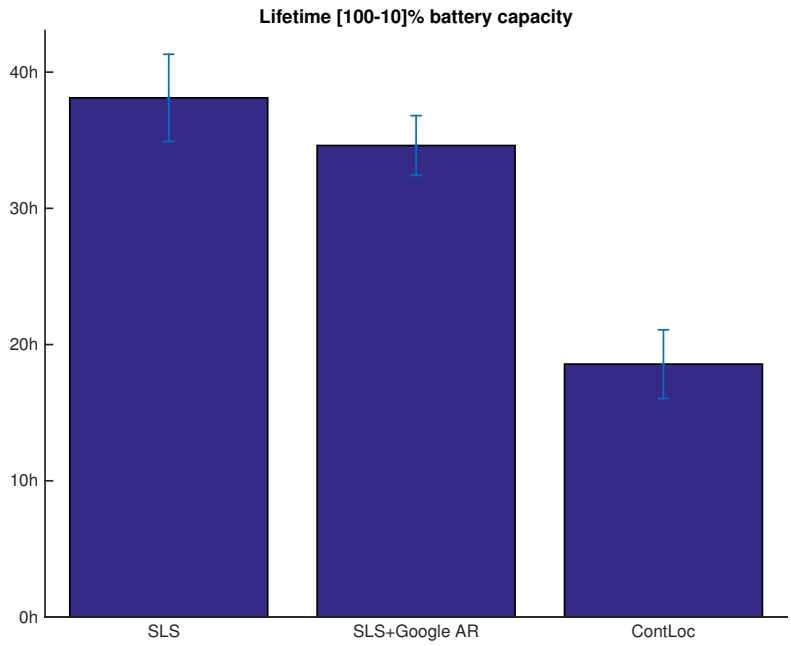


Figure 7.5. Lifetime

duration and *Day B* during which the user has been always still, visiting the same PoI for the whole day. In the graph I plot the amount of battery discharged during these two days, in percentage, by the two phones. As visible from the figure, both the devices are using more battery during Day A (*user mostly moving*) with respect to Day B. Phone C shows a larger difference in the amount of battery used during the two days: this is explained by the delay of the Google activity recognition service in sending the activity updates. In fact, by means of empirical evaluations, the Android activity update time resulted dependent on the activity itself, very delayed for some activities (more than 10 seconds to get an update, when the user is not still), and not always successful (returning an *activity not identified* result).

Phone B is not adapting its behavior to the user mobility, hence we expect an almost similar lifetime duration every day. During these specific cases, it discharged completely the battery in around 15*hours* during Day B, and around 14*hours* during Day A. This small difference may be caused by the availability of the GPS or Network location providers which may impact on the delay of the location updates (as explained in section 3.2, for the *Localization on-off* mechanism).

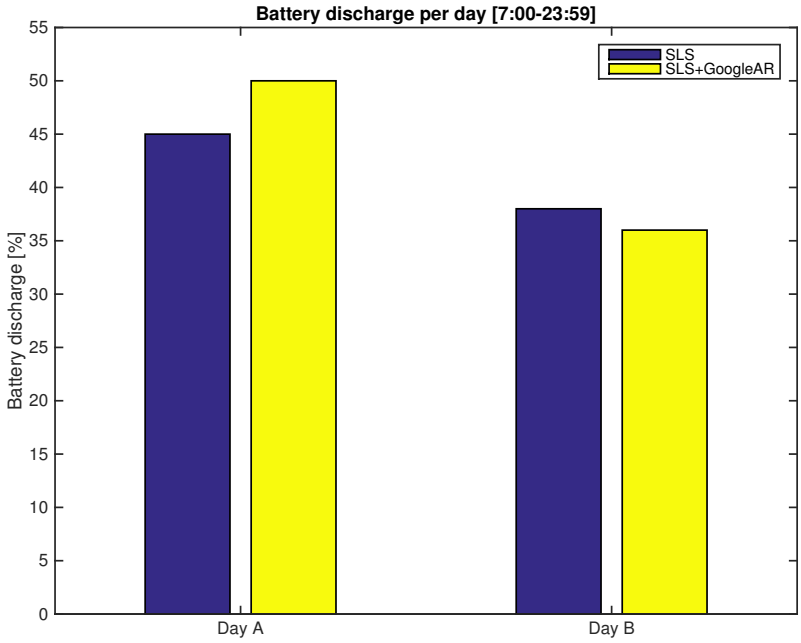


Figure 7.6. Battery discharge during a complete day

7.2.3 Amount of network traffic exchanged

In this section I will show the cumulative amount of traffic exchanged by each phone during the experiment. As already specified above, in this chapter, each device is registered with the same Google account, to ensure the comparability of the data traffic exchanged by each phone from the Android native services. Transmitting and receiving data traffic has a considerable impact on the battery resource. The SLS is a stand-alone system which does not need a back-end server either for the activity inference implementation and for the identification of the PoIs. The autonomy of the system is not only positive in terms of reduction of the usage of the battery, but also in terms of privacy. Since the SLS processes all the sensitive user's data (e.g., the activity and the location) internally, it does not face any privacy issue which involves the security of the data. All the user's sensitive data is not exchanged with any external server, and does not have to be protected, simply because it is only stored locally into the device, and only accessible by the SLS process and services.

Focusing on the battery consumption issue, given by the exchanged data traffic, I measured the cumulative amount of traffic exchanged by each phone

during the experiment. The result is shown in figure 7.7. What is shown in this plot refers to the total amount of data traffic exchanged by each phone, including the base network traffic exchanged by the Operating System and running Android native applications. All the data traffic is referred to the data connection of the phones, in fact for each of them the WiFi interface is active but there is no saved wireless network, and the phones did not connect to open networks automatically. As visible form the figure, all the three phones have a cumulative amount of traffic exchanged with is more than zero. This is explained by the localization itself. As already explained above, the location tracking mechanism implemented by the SLS makes use of the Android GPS and Network providers: this implies a communication with the Google servers (especially when the Network provider is involved).

As expected from this measurement, Phone C (implementing the SLS+GoogleAR) has the higher amount of data traffic exchanged, reaching a total amount of 51Mbyte for the whole duration of the experiment. What is really interesting from the figure is the amount of traffic exchanged by the SLS compared to the Continuous Localization system. The periodic requests to the Android localization providers performed by Phone B require an higher amount of data traffic (50Mbyte), with respect to the amount of traffic exchanged by the SLS (43Mbyte). Interpolating the measurements with a polynomial function of degree 1, Phone A has a slope of $1.18e+04$, while Phone B and Phone C have a slope of $1.439e+04$ and $1.432e+04$ respectively: this shows that the function associated to the SLS will stay below the other one, hence the amount of network traffic exchanged by the SLS will always be lower than the traffic exchanged by the Continuous Localization and the SLS+Google AR.

The high step in the figure, corresponding to the fifth day of sampling, is caused by the traffic exchanged by the Android native applications running on the devices (Google email in this case), and requiring traffic for downloading data (or in general for their periodic updates).

7.2.4 Amount of predicted locations

The location prediction performed by the SLS (both Phone A and Phone C) have an impact on the battery consumption given by the localization. In fact, when the user is moving and the system is able to predict the user's location, the location tracking is stopped. Since a location tracking involves the GPS and WiFi interfaces, while the prediction involves only querying a local database, the second procedure turns to be less expensive in terms of battery consumption.

The impact of the prediction on the overall localization, strictly depends on

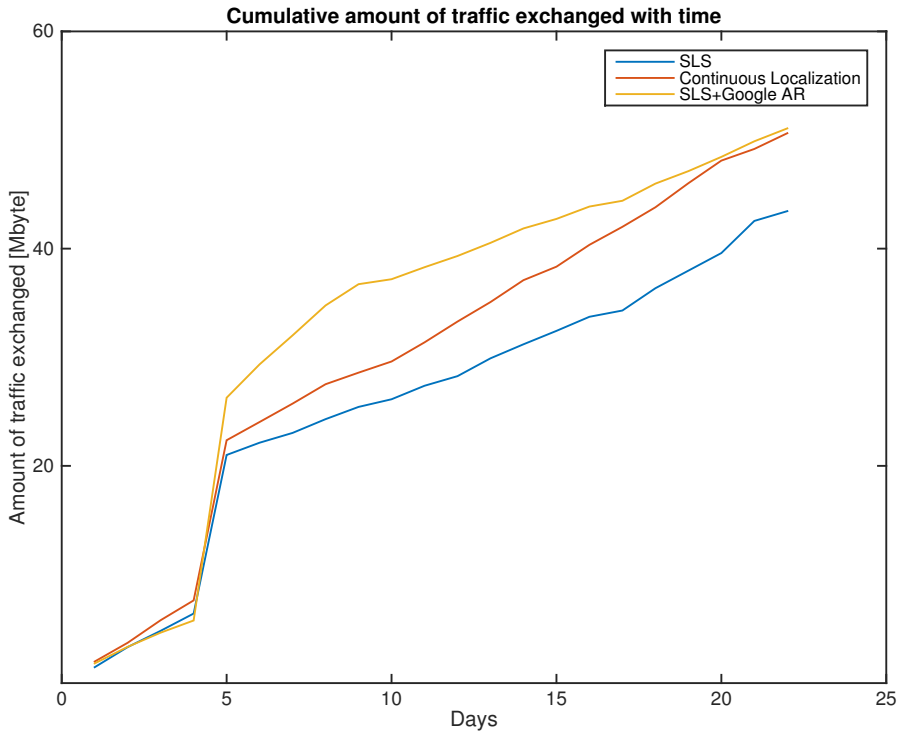


Figure 7.7. Amount of traffic exchanged

the mobility behavior of the user. In section 5.5.4 I defined *globetrotter* the user spending the majority of the time visiting LILs (Papandrea et al. [2013]), while the *creature-of-habit* is the user spending most of the time visiting her/his HILs. For the first user's category, the SLS location prediction has a very reduced impact, in fact the user tends to mostly visit new places, hence the visits are unlikely predictable. However, for the second user's class (*creature-of-habit*), and for users which have a quite regular mobility behavior, the SLS is able to learn their habits in terms of *when* and *where* the user tends to move and to build a mobility model over it. The prediction performed by the SLS by means of this mobility model have a significant impact of the localization of this class of users.

During the experiment, the first two weeks have been used for learning. The prediction procedure was then performed, during this first phase, considering the mobility of the users on the previous day of the week (i.e., prediction of tuesday, performed considering the movements of monday). During this first phase the probability threshold P_{th} defined above in section 7.1.2 have been

lowered, allowing early predictions. And in particular, has been set to 0.2. In the last phase of the experiment, from the third week, the prediction was working with a week periodicity (i.e., prediction of the movements of monday, according to the movements performed during the last mondays) as described in section 7.1.2.

To give an idea of the impact of the prediction on the localization during this final experiment, I measured the amount of time per day, the SLS performed a prediction instead of a direct location tracking. Figure 7.8 shows the total duration of the localization procedure per day, divided per provider. In the plots, the portions of the bars in yellow, represent the amount of time the system was performing a prediction: the portions in green are referred to the localization by means of the network provider; and the blu refers tot he GPS. As visible form the figure, the SLS is able to perform predictions already from the second day of the experiment. The vertical red line shows the separation between the first part and the second one of the experiment. The difference in the amount of predicted locations with respect to the direct tracking is not clearly visible, in fact the regularity of the movements of the involved user allow the system to perform early predictions, which have a great impact on the localization since the second day of the experiment. The system is able to predict locations almost every day, except for very few of them (this is the case when the user is visiting new places for all the movements of the day). In the figure, the days without data (no bar in the plot) correspond to the days in which the user was not moving out from the same PoI.

Figure 7.9 shows the global amount of time, for the whole duration of the experiment, the SLS used a certain provider, with respect to the total duration of the localization procedure (when the user is moving). As visible from the figure, there is a relevant amount of prediction in the overall duration of the localization. For Phone A, the total amount of predictions is almost the same as the total amount of locations retrieved by means of the GPS provider (about 40%). While the network provider only contributed with 20% of the retrieved locations. Phone C instead has an higher amount of location retrieved with GPS (51%) and a lower number of locations retrieved by means of the network provider (13%). For the prediction, the amount is similar as for Phone A and corresponds to 36% of the total localization.

In order to evaluate the reliability of the prediction, I also measured the distance between the predicted locations and the correspondent locations tracked by Phone B (continuous localization). This comparation does not give us a measure of the correctness of the location predictions because the location data tracked by Phone B cannot be considered as ground truth. In fact, when the

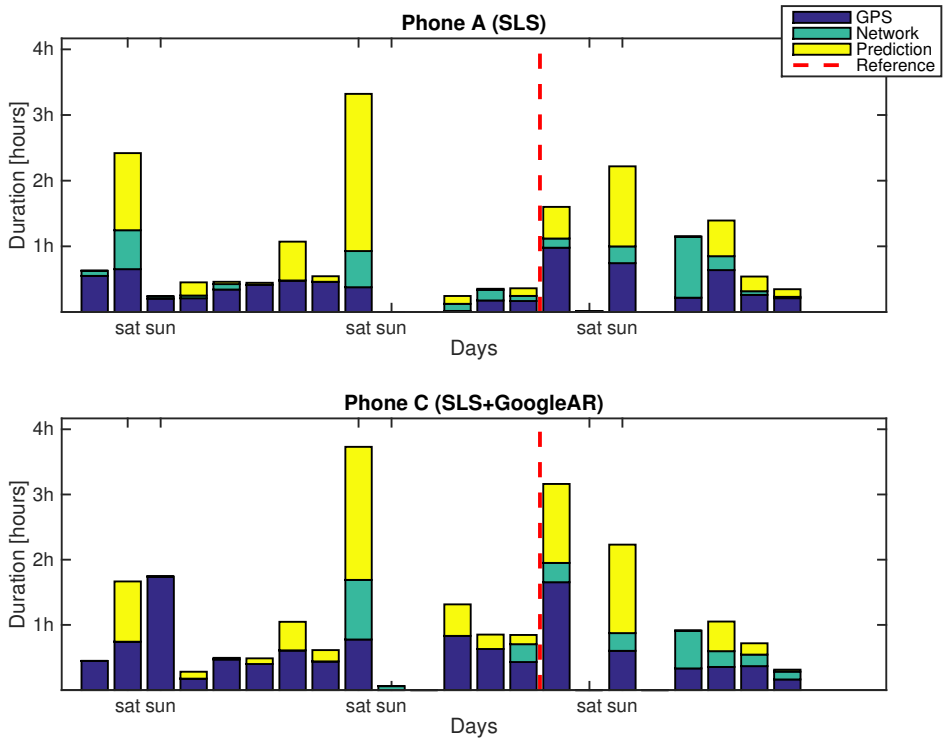


Figure 7.8. Duration of the localization per day

tracked location is retrieved by means of a network provider, the distance to the real one may reach the order of kilometers. However, this comparation gives us an idea of how close are the predicted locations from the best available ones. In table 7.5 I report the average distance between the predicted locations of Phone A and Phone C, compared to the correspondent locations tracked by Phone B by means of the Android GPS and Network providers respectively. The average distance between the predicted locations and the tracked ones is around $1.5Km$ in the case of tracking with GPS provider, and about 2 or 3 Km in case of tracking with the Network provider. The standard deviation values are quite high with respect to the mean values. This is explained by the fact that, the SLS could potentially start the prediction when the user is actually 2Km away from a PoI, and it can continue predicting the visit to this PoI while the user moves toward it, until it reaches a very small distance ($distance \sim 0$). Therefore we can read the values in the table as the average distance to the target PoI, at which the SLS starts predicting the next visiting destination.

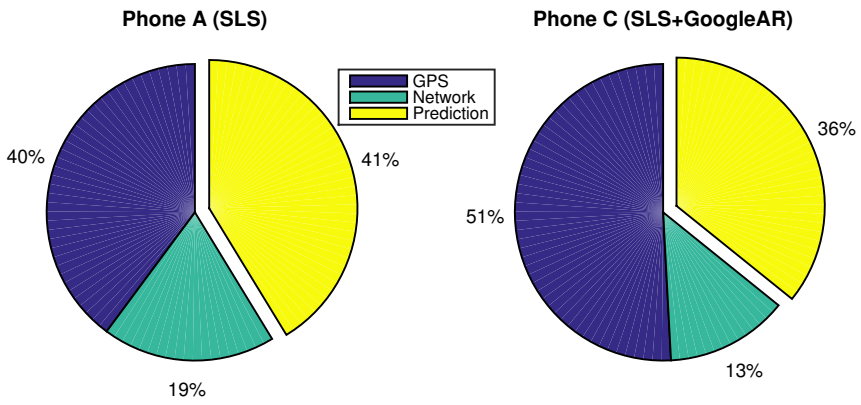


Figure 7.9. Global localization, per provider

		Mean [Km]	Standard deviation [Km]
GPS	**PhoneA**	1.607	1.097
	PhoneC	1.481	1.098
Network	**PhoneA**	3.001	1.283
	PhoneC	1.947	1.641

Table 7.5. Prediction distances to tracked locations

7.2.5 Visit and Activity Inference

The *Inference* performed by the SLS during this final experiment has been evaluated in terms of:

- capability of the system to identify a *visit*;
- accuracy of the activity inference, according to the classification in *not moving, moving by foot, slow vehicle, fast vehicle*.

Visit inference

Table 7.6 and 7.7 represent the confusion matrices related to the data collected during the experiment by Phone A (running the SLS), and Phone C (running the modified version of the SLS with Google Activity Recognition) respectively.

The unit of measure of the values reported in the matrices is the *minute*. And in particular, the *TP* (true positive) is the amount of minutes in which the user was actually visiting a PoI, and the system correctly inferred a visit. The *TN* (true negative) corresponds to the number of minutes in which the user was not visiting any PoI, and the system inferred correctly the user was not performing a visit. While *FP* (false positive) and *FN* (false negative) correspond to the incorrect inference of the system, when respectively the user was not visiting any PoI and the system inferred a visit, and when the user was really visiting a PoI and the system inferred he was moving.

ACTUAL\INFERRED	**Visit**	**Not visit**
Visit	TP = 28556	FN = 89
Not visit	FP = 738	TN = 1009

Table 7.6. SLS: Confusion Matrix for the Visits identification

ACTUAL\INFERRED	**Visit**	**Not visit**
Visit	TP = 28076	FN = 569
Not visit	FP = 490	TN = 1255

Table 7.7. SLS+Google AR: Confusion Matrix for the Visits identification

Since the ground truth has been collected manually, during everyday activities, by means of a *time scheduling* mobile application, there may be some small delays or anticipations in the logged activities, with respect to the real performed ones. To overcome this problem, I remove from the ground truth (only for this evaluation processing) the first and last minute of each activity, and consequently, also the activities lasting less than 2 minutes.

To be able to compare between the SLS and the modified version of the SLS with Google AR, I calculated the precision, accuracy and recall values for both of them (table 7.8). As visible from the table, both devices perform very well in inferring when the user is visiting a PoI, or not. In many situations the SLS performs mistaken inferences after the visit to a PoI. In fact, after a visit, when the user starts moving, the system could identify this mobility change with a delay of 10 minutes (in the worst case), according to the adaptive reasoning period values: most of the inference FP are performed during this interval in which the SLS does not check the activity of the user, and infers the user is still visiting the last seen PoI, while he is actually already moving away. Moreover, most of the FN are performed by the SLS when the user is actually visiting a

PoI, but within this visit he performs moving activities longer than the minimum $dt_{th,GAP}$ interval.

In the table the values of Accuracy, Precision and Recall are all very high (higher than 96%) for both Phone A and Phone C. I report in the table also the Balanced Accuracy, whose formula is reported in equation 7.8. Its value is low compared to the accuracy (for both phones), in fact it depends half on ratio of correct predictions when the user is performing a visit, and half on the ratio of the correct predictions while the user is not performing a visit. Since the amount of time while visiting a PoI is much higher than the time while moving between them, the balanced accuracy handles this issue and gives to both the cases the same weight. Its value reported in table 7.8 is quite low because it is affected by the low precision in identifying a "not visit".

$$0.5 * \frac{TP}{TP + FN} + 0.5 * \frac{TN}{FP + TN} \tag{7.8}$$

	SLS	**SLS + Google AR**
Precision	97.5%	98.3%
Recall	99.7%	98.0%
Accuracy	97.3%	96.5%
Balanced Accuracy	78.8%	84.9%

Table 7.8. Evaluation of the visit's inference

Activity class inference

The same methodology used for the visits, has been applied to calculate the accuracy of the activity inference. The SLS infers the activity of the user in classes, as already explained in chapter 4. Tables 7.9 and 7.10 represent the confusion matrices related to the data collected during the experiment by Phone A and Phone C respectively. Also in this case, the unit of measure for the values reported in the tables is the *minute*. Since the classes involved in the inference algorithms are four, I represented in each element (i, j) of the matrices the number of minutes in which the user was performing activity i and the system was inferring activity j.

The row corresponding to the *slow vehicle* activities is composed by all zeros elements, because the user was not performing this activity for the whole duration of the experiment. For the same reason, in table 7.11 this activity is

missing. For the *not moving* activity, both the devices are performing well: the accuracy on the identification of this class of activity corresponds to 97.3% for the SLS and to 97.1% for the Google Activity Recognition. The inference of the *fast vehicle* class is less accurate, and it presents many False Negative, inferring the class *Not Moving* (hence lowering the recall value). There are two main reasons for these FN. First, the device is often still while being in the car (e.g. at a traffic light, stuck in the traffic). The second reason is more technical, and in particular, the main discriminant feature for the differentiation of the *fast vehicle* and *moving by foot* activities is the standard deviation of the amplitude of the acceleration vector. In many situations, the stability of the car and the driving at a constant speed, gives an acceleration whose standard deviation is similar to the still activity. However the balanced accuracy (which takes into account the imbalanced dataset) has a value of 71.5% for Phone A and 71.6% for Phone C.

The main source of error is given by the *moving by foot* activity. This class of activities includes movements like *walking and strolling*, while the *standing and still* activities are included in the *not moving* class. While collecting the ground truth it was quite difficult to label the *standing activity*, which however was included in the *moving by foot* class, and the *not moving* class was used to label the only *still activity*. The other main source of FN for this class is the prediction of *fast vehicle* for both Phone A and Phone C. The balanced accuracy for this class correspond to 69.6% for the SLS and 70.7% for the Google Activity Recognition.

INFERRED / **ACTUAL**	**Fast vehicle**	**Slow vehicle**	**Moving by foot**	**Not moving**
Fast vehicle	363	12	26	419
Slow vehicle	0	0	0	0
Moving by foot	257	19	375	302
Not moving	90	1	19	28460

Table 7.9. SLS: Confusion Matrix for the Activity Inference

In table 7.11 I also show the Global performances of the two algorithms, which are very similar in terms of accuracy. It is worth mentioning once here that the SLS does not require a back-end server to perform the activity inference, while the Google AR does not work without internet connection. And, furthermore, the SLS response time is shorter than the Google Activity Recognition.

INFERRED / ACTUAL	Fast vehicle	Slow vehicle	Moving by foot	Not moving
Fast vehicle	359	0	6	458
Slow vehicle	0	0	0	0
Moving by foot	127	18	362	365
Not moving	11	3	18	28217

Table 7.10. SLS+Google AR: Confusion Matrix for the Activity Inference

7.2.6 PoIs identification

In this section I will show the results of the PoIs identification analysis. First of all I will analyze the growing number of PoIs for both the devices running the SLS (Phone A and Phone C) and consecutively I will measure the correctness of the PoIs, calculating the distances with the ones specified in the ground truth.

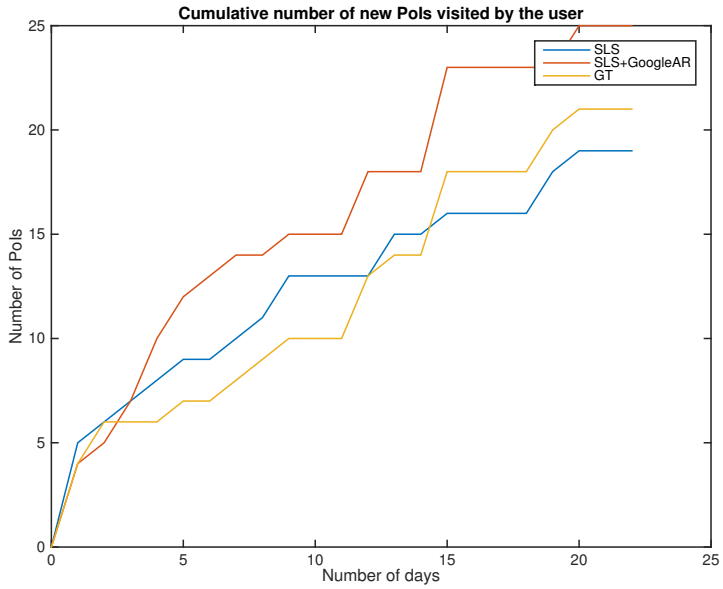


Figure 7.10. Cumulative number of new identified PoIs

Figure 7.10 shows the growing number of PoIs identified by Phone A (SLS) and Phone C (SLS + Google AR) for the whole duration of the experiment, compared with the ground truth. As visible from the figure, the total number of PoIs is constantly increasing, in fact the user tends to always visit new places. Especially during this experiment which is lasting for a limited amount of time,

		SLS	SLS + Google AR
Not moving	Precision	97.5	97.2
	Recall	99.6	99.9
	Accuracy	97.3	97.1
	Balanced Accuracy	79.5	75.7
Moving by foot	Precision	89.3	93.8
	Recall	39.3	41.5
	Accuracy	97.9	98.2
	Balanced Accuracy	69.6	70.7
Fast vehicle	Precision	51.1	72.2
	Recall	44.3	43.6
	Accuracy	97.3	97.9
	Balanced Accuracy	71.5	71.6
Global	Precision	96.2	96.6
	Recall	96.2	96.6
	Accuracy	98.1	98.3

Table 7.11. Evaluation of the activity's inference [%]

the system is constantly learning user's new relevant locations. After a certain amount of time (which may depend on the mobility category of the user), the slope of the cumulative number of new PoIs decreases.

This PoIs learning procedure may result in a massive usage of the storage resources, and consequently in the increment of the response time for the queries to the local database. To avoid this problem, the SLS eliminates from its internal storage the information about PoIs which become, with time, not relevant to the user, the LILs (as explained in section 7.1.2). Figure 7.11 gives an idea of the amount of PoIs which belongs to the LILs class (in yellow), at the end of the experiment. The separation of the PoIs in classes has been performed according to the methodology already explained before, in section 5.5.1. As visible from the figure, the LIL class includes the highest number of PoIs, compared to the other two classes of relevance.

In figure 7.12 I show the percentage number of PoIs, over the complete set of stored ones, per different values of relevance, at the end of the experiment. According to the previous figure, we can more clearly see from this graph that the majority of the PoIs (from $64\% to 81\%$) have a relevance lower than 10%, while only less than 5% of the stored PoIs have a relevance higher than 90%

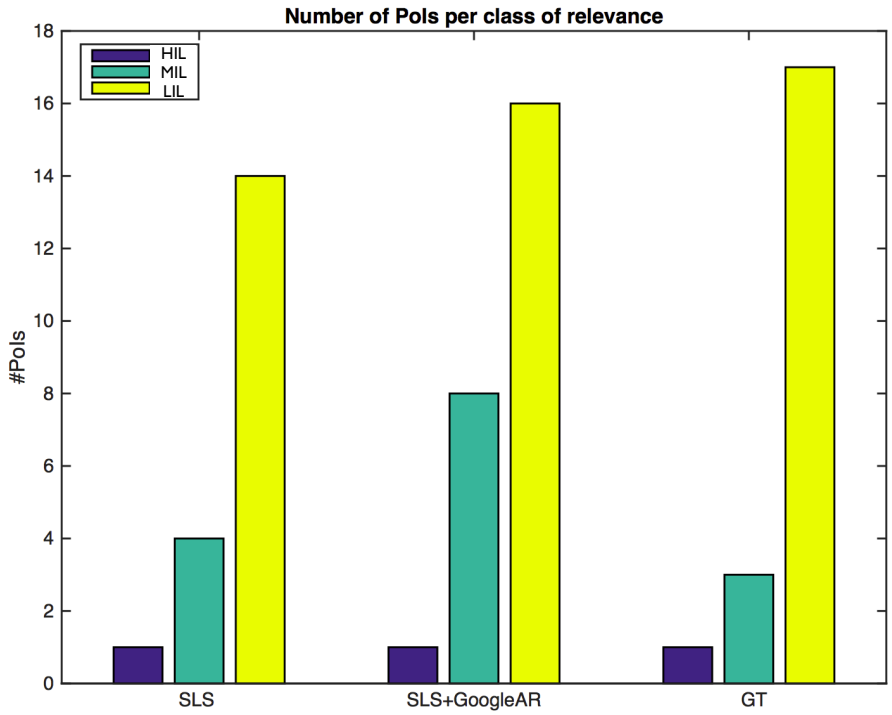


Figure 7.11. Number of PoIs per class of Relevance

(HILs). Therefore, this result justifies the SLS solution of deleting the LILs from the storage, to optimize the storage resource usage.

Table 7.12 reports the values, calculated for the whole duration of the experiment, about the average daily number of new identified PoIs, and the correspondent error. This error is calculated averaging the daily difference of the number of new identified PoIs by either the SLS and the SLS+GoogleAR, with the correspondent number from the ground truth. In the ground truth, the average number of new PoIs identified per day is 0.95, this means that the user tends to visit, in average, 0.95 new PoIs per day. Both Phone A and Phone C identify every day a number of new PoIs which differs from the ground truth of 0.54 and 0.63 PoIs in average, respectively. This error value is quite high, considering that it corresponds to 63% of the average for the SLS and to the 56% for the SLS+GoogleAR. To understand this values, I evaluated the error in the PoIs identification by measuring the distance between the wrongly identified PoIs and the correct ones from the ground truth.

I report in table 7.13 those distances, and in particular: for the correctly

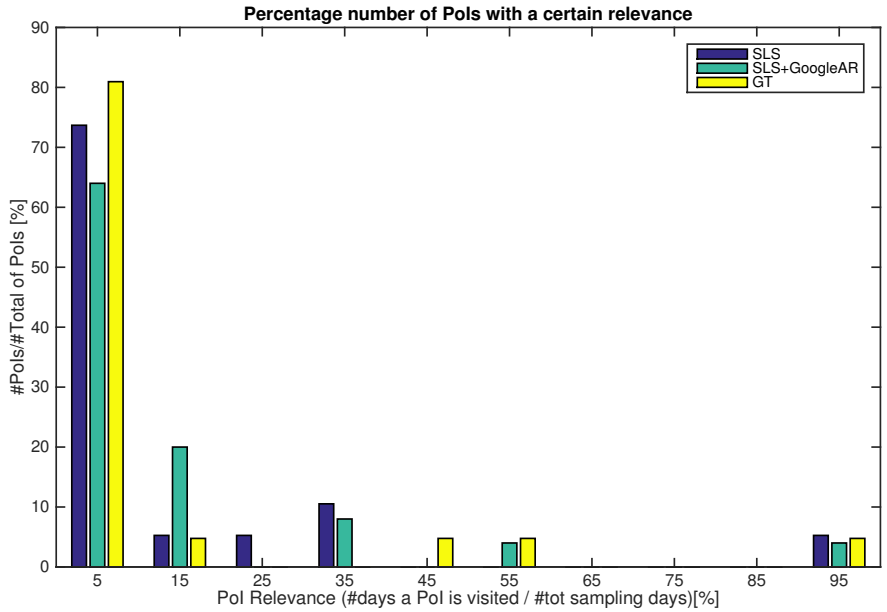


Figure 7.12. Percentage number of PoIs per value of relevance

	SLS	**SLS+GoogleAR**
Average Number of new PoIs	0.86	1.13
Average error in the number of new PoIs	0.54	0.63

Table 7.12. Average per day: number of new PoIs and error against the GT

identified PoIs, the average distance to the actual corresponding locations form the GT; for the wrongly identified ones, the distances to the nearest PoIs from the GT. The correctly identified PoIs correspond to the ones which lie within a circle area with radius $radius_{max}$ (whose value is reported in table 7.2), around the PoIs specified in the ground truth. And the wrongly identified ones, are all the other PoIs which lie outside these circles.

In table 7.13 I also specify the percentage of correct PoI identification for the whole duration of the experiment. From the values reported in the table, the SLS seems to work better than the modified version with Google activity inference. However, the two systems implement the same procedure for the localization. The only difference is in the inference of the activity, which drives the changes to the localization frequency. The lower accuracy of the *not moving* activity of the SLS+GoogleAR with respect to the SLS, implies the forcing of the location tracking for Phone C in situations when, for example, the user is actually still

and Phone A is not tracking him. In such situations, the user may be still while visiting an indoor location with GPS not reachable. The only provider available in this case is the Network one, which typically provides information with an high error in terms of location extension in space. Phone C could hence localize the user with an errors in the order of kilometers. This is the basically the reason for the high average distance for the wrongly identified PoI by Phone C.

For the correctly identified PoIs, both Phone A and Phone C perform very well, identifying PoIs with an average distance from the ones in the ground truth of less than $35 meters$.

		SLS	**SLS+GoogleAR**
Correctly identified PoIs	Percentage of PoIs	68%	44%
	Average distance	29.08meters	34.69meters
Wrongly identified PoIs	Percentage of PoIs	32%	56%
	Average distance	357.95meters	4613.6meters

Table 7.13. Average distance between identified and actual PoIs

To better understand the results presented above, I also measured the total amount of time spent by the user visiting each PoIs' class of relevance. With this measure I am able to identify the category of the user running the experiment, and in particular I am able to understand if he is a *globe-trotter* or a *creature-of-habit* one. Figure 7.13 shows the result of this measurement for both Phone A (SLS) and Phone C (SLS+GoogleAR) compared to the ground truth. All the instances confirm that the user is spending most of his visiting time in PoIs belonging to the HIL class. Globally, the user spent more than 70% of his visiting time in the Most Visited PoIs, not more than 8% of its visiting time in the Low Interest Locations and the remaining time in the Medium Interest Locations. This is the typical behavior of the *creature-of-habit* category of users.

7.3 Comparative Study

This section evaluates quantitatively how the SLS outperforms already existing solutions for efficient localization. Table 7.14 reports the main results achieved by the SLS, and compares them with two of the most relevant related works (Chon et al. [2014]; Kim et al. [2010]).

With respect to a periodic localization procedure, the SLS increases the mobile device lifetime of more than 100%; also the other two solutions contribute

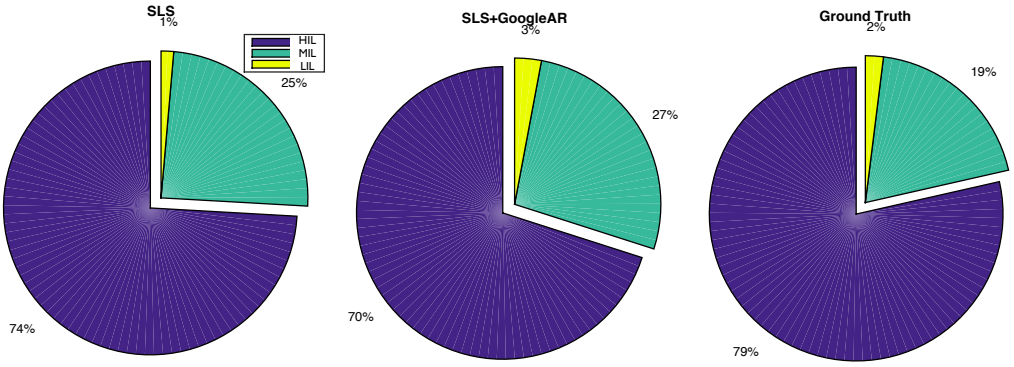


Figure 7.13. Percentage of the total visiting time per class of relevance

	SLS	**SmartDC [2014]**	**SensLoc [2010]**
lifetime increased with respect to periodic tracking	**>100%**	81%	87%
localization coverage	90%	90%	95%
activity computation complexity (time)	**80ms** 5 windows accelerometer (DecisionTree)	100ms for 95% prediction (Markov)	– 5 second accelerometer 1 feature
# visit inference accuracy	**97.3%**	93%	94%
PoI extension range	150m	500m	– WiFi fingerprinting
# PoI identification accuracy	68% avg mean error 400m	76%	–
bootstrap learning time (avg)	**2 weeks**	3 months	–
backend server computation offload	no	no	no
implementation	on mobile	emulation	on mobile

Table 7.14. Comparative study results.

to the increment of the battery lifetime compared to a fixed period procedure, corresponding to more than 80% for both of them. In terms of localization coverage, and comparing to a periodic localization procedure, all the three presented approaches perform similarly, being able to localize the user for 90% of the visited locations. While comparing the complexity of the activity inference

computation, the SLS outperforms the other solutions, being able to identify the user's activity within 80ms, while the other solutions requires more than 100ms. In terms of accuracy in the user's visits inference, the SLS achieves 97.3% precision in the number of visits correctly identified, while the other two solutions reach a precision of 93% and 94% respectively. The PoI extension range corresponds to 150m for the SLS, leading to a PoI identification accuracy of 68%, with an average distance error of 400m; for the Chon et al. [2014] instead, the PoIs identification accuracy reaches 76% while the PoI extension range is larger, corresponding to 500m. One of the strengths of SLS with respect to the related works consists in its average time for the learning bootstrap, that is the time the system needs in order to learn about the user mobility habits, before being able to reduce the resources usage. For the SLS the bootstrap time corresponds to 2 weeks, while for the Chon et al. [2014] it reaches 3 months. All the three compared solutions are built to work on mobile devices, without the support of any backend server. But only the SLS and the **?** have been really implemented and tested on-mobile; the Chon et al. [2014] instead have been evaluated on emulation environment.

7.4 Conclusion

In this chapter I validated the SLS against a continuous localization system. I performed a real experiment where a user was carrying with him three phones performing continuous localization for a total duration of 22 days. Each phone was running different versions of the localization service, and in particular: one phone was running the SLS system (Phone A), a second phone was running a continuous localization system with a fixed duty cycle (Phone B), and a third phone was running a modified version of the SLS running the Google Activity Recognition instead of the SLS Inference Module (Phone C).

The experiment showed that it is not necessary to perform continuous location tracking to ensure continuous location information. In fact, the measured location availability of Phone B is only 8% more than Phone A and 15% more than Phone C. However, the mobile device lifetime is strongly improved by using the SLS. While Phone B has an average lifetime of 18.6*hours* only performing the localization (less than a complete day), Phone A and Phone C greatly outperform it, with an average lifetime of 38.1 and 34.6 hours respectively. Since the phones were only running the localization service, in a normal situation when the phone is mainly used for phone calls, messaging and surfing the internet, the expected lifetime duration will cover the whole day. This validates the ap-

proach described in this thesis which aimed at reducing the battery resources consumption while providing continuous location information.

The measured performances of Phone A and Phone C are comparable in terms of accuracy of the visits inference, which correspond to 97.3% for Phone A and to 96.5% for Phone C. Also the inference of the class of activity is comparable, the SLS has a global accuracy of 98.1% , while the Google AR reaches a value equal to 98.3%. However, Phone A is independent from any back-end server, and in particular the activity inference algorithm may run without interned connection, while Phone C strictly depend on the Google Services. This factor causes a difference in the amount of network transmitted and received by the two phones: for instance, for this experiment, Phone A exchanged 14% less data traffic than Phone C.

Not only the activity inference and the adaptive localization duty-cycling contribute to the reduction of the battery consumption, but also the location prediction performed by the SLS. In fact, almost 40% of the locations retrieved by Phone A and C are provided by the Prediction Module. Moreover, the SLS performs well in identifying the user's PoIs: it identifies correctly 69% of the PoIs and for the remaining ones, it performs an average error of only 397.25 meters.

Moreover, I measured the impact of SLS in the storage usage, and showed that the amount of relevant location for the prediction model is very low (total number of HIL and MIL): 5 PoI for the SLS, over a total amount of 19 identified PoIs. Also, the user involved in the experiment has been classified as a "*creature-of-habit*" user, with an amount of 76% of the total time spent visiting his HILs.

Finally, I would like to mention that SLS is a solution without security holes, all the algorithms are run on the mobile, there is no exchange of information with other nodes/servers, hence no privacy issues.

Chapter 8

Conclusion and outlook

The main contribution of this dissertation is the design, implementation and validation of a **smart localization solution for mobile devices**. Considering the importance of the location as a context information in many scenarios, this thesis addressed the issue of the resources consumption due to the localization service. And in particular, an enhanced localization solution was proposed that efficiently reduces the overall energy consumption compared to standard localization techniques.

8.1 Summary and conclusions

The research work performed for the development of this thesis follows three main directions.

8.1.1 Human Mobility

The first research direction is *Human Mobility*: the key idea was to build a model which reflects the regularity of the user while visiting her/his relevant Points of Interest. Being able to model the mobility habits of the user in terms of locations visited and timing of the visits allows a smarter usage of the mobile devices resources while performing localization. And additionally, it gives the possibility to classify the user according to his mobility behavior and to predict his regular movements. In this thesis a user-centric mobility model was proposed based on the *relevance* property of the Points of Interest. I show that the PoIs relevant for a user can be divided into three main classes of relevance: High Interest Locations, limited in number, where the user spends most of the time and which are visited almost daily; Medium Interest Locations, where the user spends a

relevant amount of time and which are visited frequently; Low Interest Locations, which are location sporadically visited by the user. The mobility model presented in this thesis is based on the first two groups of PoIs (HIL and MIL), and in particular on the timing habits of the user while moving among these PoIs. Since the majority of the PoIs visited by a user are LILs, the identification of this class of relevance allows the model to neglect them, and it reduces the storage usage. Within the SLS solution, the *Learning Module* is in charge of this task: it learns the habits of the user in terms of *visits* to PoIs and builds the user mobility model. The studies performed in this direction of my work leaded to important results.

- *"Only few PoIs are really relevant for Human Mobility."*
- *"The main parameter governing human decisions on movements is travel time, and not travel distance."*.

8.1.2 Mobility Prediction

The second research direction of my work is *Mobility Prediction* and it is strictly related to the previously described one: modeling the user mobility behavior gives the possibility to also predict the users movements. In this thesis I implemented a *context based location prediction algorithm* whose task is to reduce the direct location tracking procedure while the user performs regular visits to relevant PoIs. The main goal of this algorithm is to reduce the usage of the battery resources while continuously tracking the user. In a final validation experiment I showed that, for a general user classified as *creature-of-habit* (hence showing a regular behavior), the SLS is able to predict the next locations up to 40% of the time, while the user is moving. This is a relevant result, in fact the majority of users present regularities in their mobility behavior. However this result strengthens the potential of the proposed algorithm.

8.1.3 Activity Inference

The third direction of this work focuses on *Activity Inference*, and in particular I investigated a methodology to adapt the localization duty cycling according to the activity of the user. The accelerometer sensor was exploited (nowadays available on almost every smartphone) to retrieve a set of 30 features in order to infer the activity of the user, among four different classes of activity: not moving, moving by foot, slow vehicle and fast vehicle. The inferred activity is then used to change the location tracking duty cycle, reducing it while the user is

moving fast, and increasing it while the user is moving more slowly; and finally stopping the location tracking while the user is static. As frequently proposed in the body of related work, I used the Decision Tree algorithm, which is known to be very lightweight in terms of resource consumption, hence a good candidate for running on a mobile platform. The inference algorithm is trained offline, and it runs completely on the mobile device, without the support of any back-end server. The benefit is twofold: it prevents any privacy issue raised by the transmission of private information; and it does not generate any data traffic, which might be expensive in terms of battery consumption. In the final validation experiment mentioned above, I measured the performance of the algorithm and compared it with the Google Activity Recognition service. The measured accuracy is comparable for both solutions and it corresponds to 98.1% for the presented algorithm and to 98.3% for the Google implementation. However, the network traffic exchanged during the experiment has been reduced by almost 15% with the SLS inference algorithm, and the localization does not have security holes while being comparable in terms of performance.

All the three specified directions of my work have been carried out in parallel, and they merged in the implementation of the SLS. In the final validation experiment I have shown that the lifetime of a mobile device may be doubled by using the SLS as opposed to a traditional continuous localization system, while losing only less than 10% of the localization coverage.

The remarkable results achieved in this thesis proved that machine learning and artificial intelligence are very efficient approaches to solve the resource consumption problem associated to the localization. This dissertation paves the way to further location-based services and optimizations, which will inherently improve the performances of mobile applications, lower their cost and complexity, and expand their potentials.

8.2 Directions for future research

I list here some interesting research directions that can be addressed to extend the work presented in this dissertation.

One research topic that can be addressed is the analysis of the user social behavior. Learning and modeling the user mobility gives the possibility to understand how the users moves and visits certain locations: for example, how long a user visits areas in the HIL class, how often he visits locations in the LIL class, etc. These information about how long and how often a user visits certain location reference classes could further the understanding of the user's behavior

and social category. Some initial results in this direction have been presented in this dissertation, classifying the mobility behavior of the user in *creature-of-habit* and *wanderer*.

Modeling the mobility behavior of the user in terms of PoIs, classes of relevance and visit probability distributions with time gives the inputs to perform a semantic inference of the visited locations. More specifically, given the regularity of the human behavior in visiting high interest locations, it is possible to infer their semantic meaning for the user (e.g., if a certain PoI is home or work place). Changing the perspective, and analysing the mobility of users from a social point of view, hence analysing the mobility of the crowd among the socially relevant PoIs, the relevance classes will change their semantic content (e.g., an High-Interest Location for a single user could be his home, an High-Interest Location for the crowd could be a popular restaurant). This allows the inference of the semantic meaning of crowd-relevant PoIs, according to their visiting behavior with time.

The user next-visit prediction algorithm implemented by the SLS allows the prediction of the user's movements toward High-Interest Locations and Medium-Interest Locations. However the SLS is not able to predict visits to Low-Interest Locations; in fact, by definition, such locations are seldom visited by each individual user. Also in this case analysing the mobility of the users from a social point of view would allow modeling the crowd mobility among those PoIs. Locations that are LILs for a particular user, may be much more relevant for the crowd, who visits them with a certain regularity.

The version of the SLS developed and analyzed within this thesis does not provide any query interface between the application layer and the SLS, through which individual location-based apps could specify specific requirements such as accuracy and latency. However this is one of the ongoing future work direction. When providing location information to many applications, the SLS will set its parameters in order to satisfy the strictest application requirements, specified in terms of location update frequency and location accuracy. This way, it will be able to satisfy all the other coarser specifications. This application requirement tuning is performed by the SLS modifying accordingly the reasoning period (for the information latency) and the P_{th} and $dist_{th}$ values (for the prediction accuracy). However, the SLS will offer a best effort service to the application layer, in order to adapt its behavior to the actual battery availability.

Publication List

8.3 Peer-reviewed Articles

1. M. Papandrea and S. Giordano. Location prediction and mobility modelling for enhanced localization solution. *Journal of Ambient Intelligence and Humanized Computing (Springer)*, 5:279–295, June 2014.

2. Matteo Zignani, Michela Papandrea, Sabrina Gaito, Silvia Giordano and Gian Paolo Rossi. On the key features in human mobility: relevance, time and distance. *In Proceeding of International Workshop on the Impact of Human Mobility in Pervasive Systems and Applications (PerMoby) at PerCom'14*, March 2014

3. M. Papandrea, M. Zignani, S. Gaito, S. Giordano, G. P. Rossi. How many places around you? *In Proc. of International Workshop on the Impact of Human Mobility in Pervasive Systems and Applications (PerMoby) at PerCom'13*, March 2013.

4. Michela Papandrea, Silvia Giordano. Enhanced Localization Solution. *In Proc. of International Workshop on the Impact of Human Mobility in Pervasive Systems and Applications (PerMoby) at PerCom'12*, March 2012.

5. Emiliano Miluzzo, Michela Papandrea, Nicholas D. Lane, Andy M. Sarroff, Silvia Giordano, Andrew T. Campbell. Tapping into the Vibe of the City using VibN, a Continuous Sensing Application for Smartphones. *In Proc. of First International Symposium on Social and Community Intelligence (SCI) at Ubicomp'11*, September 2011.

6. Emiliano Miluzzo, Michela Papandrea, Nicholas Lane, Hong Lu, Andrew T. Campbell. Pocket, Bag, Hand, etc. - Automatically Detecting Phone Context through Discovery. *In Proc. of First International Workshop on Sensing for App Phones (PhoneSense) at SenSys'10*, November 2010.

7. Piergiorgio Cremonese, Dario Gallucci, Michela Papandrea, Salvatore Vanini, and Silvia Giordano. PROMO: continuous localized and profiled multimedia content distribution. *In Proc. of MoViD'10*, October 2010.

8.4 Other Relevant Publications

8.4.1 Short Papers

8. Michela Papandrea. A Smartphone-Based Energy Efficient and Intelligent Multi-Technology System for Localization and Movement Prediction. *In Proc. of PerCom'12 (PhDForum)*, March 2012. **Best PhD Student Award**

9. Michela Papandrea. Opportunistically Supported Ubiquitous Localization: Machine Learning Enhancements. *In Proc. of WoWMoM'10 (PhDForum)*, June 2010.

10. Michela Papandrea. Opportunistically Supported Ubiquitous Localization. *In Proc. of MobiOpp'10 (PhDForum)*, February 2010.

11. M. Papandrea. Multimodal Ubiquitous Localization: a GPS/WiFi/GSM-based lightweight solution. *In Proc. of WoWMoM'09 (PhD Forum)*, June 2009.

8.4.2 Technical Reports

12. Michela Panadrea and Silvia Giordano. Four places where I can be. *NetworkingLab, ISIN, SUPSI*. November 2012.

8.4.3 Demo

13. Michela Papandrea, Silvia Giordano, Salvatore Vanini, and Piergiorgio Cremonese. Proximity Marketing Solution Tailored to User Needs. *In Proc. of WoWMoM'10 (Demo)*, June 2010. **Best Demo Award**

14. M. Papandrea, S. Vanini, S. Giordano. A Lightweight Localization Architecture and Application for Opportunistic Networks. *In Proc. of WoWMoM'09 (Demo)*, June 2009.

Appendix A

On-Mobile Feature Calculation

The SLS applies an Inferential algorithm to understand the movements of the user. As already explained in the thesis, it collects data from the virtual linear acceleration sensor and calculates a set of features over it. These features are then provided to a trained decision tree for the activity inference. To summarize, the set of features calculated by the system are reported in table A.1.

Feature	**Symbol**	**Formula**
Mean	μ	$\frac{1}{n}\sum_{i=1}^{n} f_i$
Standard Deviation	σ	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(f_i-\mu\right)^2}$
Signal to noise ratio	SNR	$\frac{\mu}{\sigma}$
Peak-Peak amplitude	$P-PA$	$\frac{1}{n}\sum_{i=1}^{n}\left(f\left(t_i\right)-f\left(t_k\right)\right)$ $\exists t_k : f\left(t_k\right) \leqslant f\left(t_j\right), \forall j \neq k$
Energy	E	$\frac{1}{n}\sum_{i=1}^{n} f_i^2$
Derivative	d	$\frac{1}{n}\sum_{i=1}^{n}\left\lvert\frac{df_i}{dt_i}\right\rvert$

Table A.1. List of features

Since the features are calculated online, it is extremely important to optimize the computational cost in order to have a reduced battery consumption. While it is straightforward the calculation of some of these features on the stream of the data (e.g., mean and energy), for some other features it may be required a second reading of the samples (e.g. standard deviation and signal to noise ratio). For example, the definition of the standard deviation reported in the table requires an a-priori knowledge of the mean value, which implies two passes over the data. This is not a feasible solution for the SLS online feature calculation, which need to produce incremental results after each sample becomes available.

In the following I report the development of the variance definition (applying the square root operation to the variance we retrieve the standard deviation), which results in equation A.5. It solves the "data double pass" problem since it allows the calculation of the standard deviation from two running sums.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu\right)^2 \tag{A.1}$$

$$\frac{1}{n}\sum_{i=1}^{n}\left(x_i^2 - 2x_i\mu + \mu^2\right) \tag{A.2}$$

$$\frac{1}{n}\sum_{i=1}^{n}x_i^2 - 2\mu\frac{1}{n}\sum_{i=1}^{n}x_i + \mu^2\frac{1}{n}\sum_{i=1}^{n}1 \tag{A.3}$$

$$\frac{1}{n}\sum_{i=1}^{n}x_i^2 - 2\mu\mu + \mu^2\frac{n}{n} \tag{A.4}$$

$$\frac{1}{n}\sum_{i=1}^{n}x_i^2 - \mu^2 \tag{A.5}$$

In the following I will report the pseudo code of the online calculation of all the features (algorithm 3). This calculation is applied to all the 5 values of the data input vector (described in paragraph 4.6.3). To simplify the code, I will refer to a general stream of data f. Each sample of the data is referred as f_i, and f_0 is the first sample. The number of samples in the data window is n. As visible from the algorithm, reading only once the stream data it is possible t calculate all the 30 features involved in the inference algorithm. This algorithm gives the possibility to perform the features calculation without storing the sampled data, hence without impacting the storage resources of the mobile device.

$A = f_0$;
$B = f_0^2$;
$D = f_0$;
$F = 0$;
while *read* f_i *in the sliding window* $[f_2, f_n]$ **do**
  $A = A + f_i$;
  $B = B + f_i^2$;
  $D = min(D, f_i)$;
  $F = F + \left|\frac{f_i - f_{i-1}}{t_i - t_{i-1}}\right|$;
end
$\mu = \frac{A}{n}$;
$E = \frac{B}{n}$;
$sigma = \sqrt{E - \mu^2}$;
$SNR = \frac{\mu}{\sigma}$;
$P - PA = \mu - D$;
$d = \frac{F}{n}$;

Algorithm 3: Online features calculation

Bibliography

Abdesslem, B., Phillips, A. and Henderson, T. [2009]. Less is more: Energy-efficient mobile sensing with senseless, *Proceedings of MobiHeld'09*, New York, NY.

Akoush, S. and Sameh, A. [2007]. Mobile user movement prediction using bayesian learning for neural networks, *Proceedings of the 2007 international conference on Wireless communications and mobile computing*, ACM, pp. 191–196.

Anguita, D., Ghio, A., Oneto, L., Parra, X. and Reyes-Ortiz, J. L. [2012]. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine, *Ambient assisted living and home care*, Springer, pp. 216–223.

Bamis, A. and Savvides, A. [2010]. Lightweight extraction of frequent spatio-temporal activities from gps traces, *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, IEEE, pp. 281–291.

Bao, L. and Intille, S. S. [2004]. Activity recognition from user-annotated acceleration data, *Pervasive computing*, Springer, pp. 1–17.

Bareth, U. and Kupper, A. [2011]. Energy-efficient position tracking in proactive location-based services for smartphone environments, *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*, IEEE, pp. 516–521.

Baumann, P., Kleiminger, W. and Santini, S. [2013]. The influence of temporal and spatial features on the performance of next-place prediction algorithms, *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, ACM, pp. 449–458.

B'far, R. [2004]. *Mobile computing principles: designing and developing mobile applications with UML and XML*, Cambridge University Press.

Bo, C., Li, X.-Y., Jung, T., Mao, X., Tao, Y. and Yao, L. [2013]. Smartloc: Push the limit of the inertial sensor based metropolitan localization using smartphone, *Proceedings of the 19th annual international conference on Mobile computing & networking*, ACM, pp. 195–198.

Bolliger, P., Partridge, K., Chu, M. and Langheinrich, M. [2009]. Improving location fingerprinting through motion detection and asynchronous interval labeling, *Location and Context Awareness*, Springer, pp. 37–51.

Bracciale, L., Bonola, M., Loreti, P., Bianchi, G., Amici, R. and Rabuffi, A. [2014]. CRAWDAD data set roma/taxi (v. 2014-07-17), Downloaded from http://crawdad.org/roma/taxi/.

Calabrese, F., Di Lorenzo, G. and Ratti, C. [2010]. Human mobility prediction based on individual and collective geographical preferences, *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, IEEE, pp. 312–317.

Chakraborty, S., Dong, Y., Yau, D. K. and Lui, J. C. [2006]. On the effectiveness of movement prediction to reduce energy consumption in wireless communication, *Mobile Computing, IEEE Transactions on* **5**(2): 157–169.

Cheng, H.-T., Sun, F.-T., Griss, M., Davis, P., Li, J. and You, D. [2013]. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning, *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, ACM, pp. 361–374.

Chon, J. and Cha, H. [2011]. Lifemap: A smartphone-based context provider for location-based services, *IEEE Pervasive Computing* **10**(2): 58–67.

Chon, Y., Talipov, E., Shin, H. and Cha, H. [2014]. Smartdc: Mobility prediction-based adaptive duty cycling for everyday location monitoring., *IEEE Trans. Mob. Comput.* **13**(3): 512–525.

Constandache, I., Gaonkar, S., Sayler, M., Choudhury, R. R. and Cox, L. [2009]. Enloc: Energy-efficient localization for mobile phones, *Proceedings of INFOCOM'09*, Rio de Janeiro, Brazil.

CRAWDAD data set cu/cu_wart (v. 2011-10-24) [2011]. Downloaded from http://crawdad.org/cu/cu_wart/.

CRAWDAD data set Microsoft/Vanlan (v. 2007-09-14) [2007]. Downloaded from http://crawdad.org/microsoft/vanlan/.

Cremonese, P., Gallucci, D., Papandrea, M., Vanini, S. and Giordano, S. [2010]. Promo: continuous localized and profiled multimedia content distribution, *Proceedings of the 3rd workshop on Mobile video delivery*, ACM, pp. 21–26.

Dunbar, R. I. [1992]. Neocortex size as a constraint on group size in primates, *Journal of Human Evolution* **22**(6): 469–493.

Dunbar, R. I. [1998]. The social brain hypothesis, *brain* **9**(10): 178–190.

Ester, M., Kriegel, H. P., Sander, J. and Xu, X. [1996]. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *Second International Conference on Knowledge Discovery and Data Mining*, KDD'96.

Ferrari, A., Gallucci, D., Puccinelli, D. and Giordano, S. [2014]. Detecting energy leaks in android app with poem, *Technical report*, Networking Laboratory, University of Applied Sciences of Southern Switzerland (SUPSI), Manno, Switzerland.

Flipsen, B., Geraedts, J., Reinders, A., Bakker, C., Dafnomilis, I. and Gudadhe, A. [2012]. Environmental sizing of smartphone batteries, *Electronics Goes Green 2012+(EGG), 2012*, IEEE, pp. 1–9.

Gambs, S., Killijian, M.-O. and del Prado Cortez, M. N. [2012]. Next place prediction using mobility markov chains, *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, ACM, p. 3.

Gaonkar, S., Li, J., Choudhury, R. R., Cox, L. P. and Schmidt, A. [2008]. Microblog: Sharing and querying content through mobile phones and social participation, *Proceedings of MobiSys'08*.

Giordano, S. and Papandrea, M. [2012]. Four places where I can be, *Technical report*, ISIN DTI SUPSI.
URL: *http://isin.dti.supsi.ch/NetLab/tech_rep_places2012.pdf*

Gonzalez, M. C., Hidalgo, C. A. and Barabasi, A.-L. [2008]. Understanding individual human mobility patterns, *Nature* **453**(7196): 779–782.

Gonzalez, P. A., Weinstein, J. S., Barbeau, S. J., Labrador, M., Winters, P. L., Georggi, N. L. and Perez, R. [2010]. Automating mode detection for travel behaviour analysis by using global positioning systemsenabled mobile phones and neural networks, *Intelligent Transport Systems, IET* **4**(1): 37–49.

Hemminki, S., Nurmi, P. and Tarkoma, S. [2013]. Accelerometer-based transportation mode detection on smartphones, *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ACM, p. 13.

Isaacman, S., Becker, R., Cáceres, R., Kobourov, S., Martonosi, M., Rowland, J. and Varshavsky, A. [2011]. Identifying important places in peopleâĂŹs lives from cellular network data, *Pervasive Computing*, Springer, pp. 133–151.

Jonathan Lester, Tanzeem Choudhury, G. B. [2006]. A practical approach to recognizing physical activities, **3968**: 1–16.

Keally, M., Zhou, G., Xing, G., Wu, J. and Pyles, A. [2011]. Pbn: towards practical activity recognition using smartphone-based body sensor networks, *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, ACM, pp. 246–259.

Khetarpaul, S., Chauhan, R., Gupta, S. K., Subramaniam, L. V. and Nambiar, U. [2011]. Mining gps data to determine interesting locations, *Proceedings of IIWeb '11*, Hyderabad, India.

Kim, D. H., Kim, Y., Estrin, D. and Srivastava, M. B. [2010]. Sensloc: sensing everyday places and paths using less energy, *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ACM, pp. 43–56.

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

Krumm, J. and Horvitz, E. [2006]. Predestination: Inferring destinations from partial trajectories, *UbiComp 2006: Ubiquitous Computing*, Springer, pp. 243–260.

Lim, L., Misra, A. and Mo, T. [2013]. Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams, *Distributed and Parallel Databases* **31**(2): 321–351.

Lin, K., Kansal, A., Lymberopoulos, D. and Zhao, F. [2010]. Energy-accuracy trade-off for continuous mobile device location, *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ACM, pp. 285–298.

Lin, M., Hsu, W.-J. and Lee, Z. Q. [2012]. Predictability of individuals' mobility with high-resolution positioning data, *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12.

Lu, H., Yang, J., Liu, Z., Lane, N. D., Choudhury, T. and Campbell, A. T. [2010]. The jigsaw continuous sensing engine for mobile phone applications, *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ACM, pp. 71–84.

Miluzzo, E., Lane, N. D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S. B., Zheng, X. and Campbell, A. T. [2008]. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application, *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ACM, pp. 337–350.

Miluzzo, E., Papandrea, M., Lane, N. D., Sarroff, A. M., Giordano, S. and Campbell, A. T. [2011]. Tapping into the vibe of the city using vibn, a continuous sensing application for smartphones, *Proceedings of First International Symposium on Social and Community Intelligence (SCI'11)*, Beijing, China.

Misra, A. and Lim, L. [2011]. Optimizing sensor data acquisition for energy-efficient smartphone-based continuous event processing, *Proceedings of the 2011 IEEE 12th International Conference on Mobile Data Management (MDM'11)*.

Muralidharan, K., Khan, A. J., Misra, A., Balan, R. K. and Agarwal, S. [2014]. Barometric phone sensors: More hype than hope!, *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, HotMobile '14, ACM, New York, NY, USA, pp. 12:1–12:6.

Musolesi, M. and Mascolo, C. [2006]. A community based mobility model for ad hoc network research, *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, ACM, pp. 31–38.

Nicholson, A. J. and Noble, B. D. [2008]. Breadcrumbs: forecasting mobile connectivity, *Proceedings of the 14th ACM international conference on Mobile computing and networking*, ACM, pp. 46–57.

Ofstad, A., Nicholas, E., Szcodronski, R. and Choudhury, R. R. [2008]. Aampl: Accelerometer augmented mobile phone localization, *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, ACM, pp. 13–18.

Oshin, T. O., Poslad, S. and Ma, A. [2012]. Improving the energy-efficiency of gps based location sensing smartphone applications, *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, IEEE, pp. 1698–1705.

Paek, J., Kim, J. and Govindan, R. [2010]. Energy-efficient rate-adaptive gps-based positioning for smartphones, *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ACM, pp. 299–314.

Papandrea, M. [2012]. A smartphone-based energy efficient and intelligent multi-technology system for localization and movement prediction, *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, IEEE, pp. 554–555.

Papandrea, M. and Giordano, S. [2012]. Enhanced localization solution, *Proceedings of International Workshop on the Impact of Human Mobility in Pervasive Systems and Applications, at PerCom'12 (PerMoby'12)*, Lugano, Switzerland.

Papandrea, M. and Giordano, S. [2014]. Location prediction and mobility modelling for enhanced localization solution, *Journal of Ambient Intelligence and Humanized Computing* **5**(3): 279–295.

Papandrea, M., Zignani, M., Gaito, S., Giordano, S. and Rossi, G. P. [2013]. How many places do you visit a day?, *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, PerMoby'13.

Patterson, D. J., Liao, L., Fox, D. and Kautz, H. [2003]. Inferring high-level behavior from low-level sensors, *UbiComp 2003: Ubiquitous Computing*, Springer, pp. 73–89.

Puiatti, A., Mudda, S., Giordano, S. and Mayora, O. [2011]. Smartphone-centred wearable sensors network for monitoring patients with bipolar disorder, *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, IEEE, pp. 3644–3647.

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

Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S. and Chong, S. [2009]. CRAWDAD data set ncsu/mobilitymodels (v. 2009-07-23), Downloaded from http://crawdad.org/ncsu/mobilitymodels/.

Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S. J. and Chong, S. [2011]. On the levy-walk nature of human mobility, *IEEE/ACM Transaction of Networking* .

Robles, R. J. and Kim, T.-h. [2010]. Review: context aware tools for smart home development, *International Journal of Smart Home* **4**(1).

Romoozi, M., Babaei, H., Fathy, M. and Romoozi, M. [2009]. A cluster-based mobility model for intelligent nodes, *Computational Science and Its Applications–ICCSA 2009*, Springer, pp. 565–579.

Scellato, S., Musolesi, M., Mascolo, C., Latora, V. and Campbell, A. T. [2011]. Nextplace: a spatio-temporal prediction framework for pervasive systems, *Pervasive Computing*, Springer, pp. 152–169.

Shek, S. [2010]. Next-generation location-based services for mobile devices, *Leading Edge Forum, Computer Science Corporation*, pp. 1–66.

Siirtola, P. and Röning, J. [2012]. Recognizing human activities user-independently on smartphones based on accelerometer data, *International Journal of Interactive Multimedia and Artificial Intelligence* **1**(5).

Sinnott, R. W. [1984]. Virtues of the haversine, *Sky and Telescope* **68**: 158.

Song, L., Kotz, D., Jain, R. and He, X. [2006]. Evaluating next-cell predictors with extensive wi-fi mobility data, *Mobile Computing, IEEE Transactions on* **5**(12): 1633–1649.

Song, W., Lee, J., Schulzrinne, H. G. and Lee, B. [2013]. Finding 9-1-1 callers in tall buildings.

Stenneth, L., Wolfson, O., Yu, P. S. and Xu, B. [2011]. Transportation mode detection using mobile phones and gis information, *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, pp. 54–63.

Su, X., Tong, H. and Ji, P. [2014]. Activity recognition with smartphone sensors, *Tsinghua Science and Technology* **19**(3): 235–249.

Ustev, Y. E., Incel, O. D. and Ersoy, C. [2013]. User, device and orientation independent human activity recognition on mobile phones: challenges and a proposal, *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing*.

Vanini, S. and Giordano, S. [2013]. Adaptive context-agnostic floor transition detection on smart mobile devices, *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, IEEE, pp. 2–7.

Wang, Y., Lin, J., Annavaram, M., Jacobson, Q. A., Hong, J., Krishnamachari, B. and Sadeh, N. [2009]. A framework of energy efficient mobile sensing for automatic user state recognition, *Proceedings of the 7th international conference on Mobile systems, applications, and services*, ACM, pp. 179–192.

Watts, D. J. and Strogatz, S. H. [1998]. Collective dynamics of âĂŸsmall-worldâĂŹnetworks, *nature* **393**(6684): 440–442.

Xu, C., Ji, M., Chen, W. and Zhang, Z. [2010]. Identifying travel mode from gps trajectories through fuzzy pattern recognition, *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, Vol. 2, IEEE, pp. 889–893.

Yan, Z., Subbaraju, V., Chakraborty, D., Misra, A. and Aberer, K. [2012]. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach, *Wearable Computers (ISWC), 2012 16th International Symposium on*, Ieee, pp. 17–24.

Zhang, L., Liu, J., Jiang, H. and Guan, Y. [2013]. Senstrack: Energy-efficient location tracking with smartphone sensors, *Sensors Journal, IEEE* **13**(10): 3775–3784.

Zheng, Y., Li, Q., Chen, Y., Xie, X. and Ma, W.-Y. [2008]. Understanding mobility based on gps data, *Proceedings of ACM conference on Ubiquitous Computing (UbiComp 2008)*, ACM Press, Seoul, Korea, pp. 312–321.

Zheng, Y., Xie, X. and Ma, W.-Y. [2010]. Geolife: A collaborative social networking service among user, location and trajectory, *Invited paper, in IEEE Data Engineering Bulletin*, pp. 32–40.

Zheng, Y., Zhang, L., Xie, X. and Ma, W.-Y. [2009]. Mining interesting locations and travel sequences from gps trajectories, *Proceedings of International conference on World Wild Web (WWW 2009)*, ACM Press, Madrid, Spain, pp. 791–800.

Zhou, C., Bhatnagar, N., Shekhar, S. and Terveen, L. [2007]. Mining personally important places from gps tracks, *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, IEEE, pp. 517–526.

Zhou, C., Frankowski, D., Ludford, P., Shekhar, S. and Terveen, L. [2004]. Discovering personal gazetteers: An interactive clustering approach, *Proceedings of ACMGIS*, pp. 266–273.

Zignani, M. [2012]. Geo-comm: A geo-community based mobility model, *Wireless On-demand Network Systems and Services (WONS), 2012 9th Annual Conference on*, IEEE, pp. 143–150.

Zignani, M. and Gaito, S. [2010]. Extracting human mobility patterns from gps-based traces, *Wireless Days (WD), 2010 IFIP*, IEEE, pp. 1–5.

Zignani, M., Gaito, S. and Rossi, G. [2012]. Extracting human mobility and social behavior from location-aware traces, *Wireless Communications and Mobile Computing* .

Zignani, M., Papandrea, M., Gaito, S., Giordano, S. and Rossi, G. P. [2014]. On the key features in human mobility: relevance, time and distance, *Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, PerMoby'14.