Department of Computer Science
Series of Publications A
Report A-2009-7

# Identifying Meaningful Places

Petteri Nurmi

*To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIV, University Main Building, on October 24th, 2009, at 10 o'clock.*

University of Helsinki
Finland

## Contact information

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

Email address: postmaster@cs.Helsinki.FI (Internet)

URL: http://www.cs.Helsinki.FI/

Telephone: +358 9 1911

Telefax: +358 9 191 51120


ISSN 1238-8645
ISBN 978-952-10-5789-2 (paperback)
ISBN 978-952-10-5790-8 (PDF)
Computing Reviews (1998) Classification: C.2.4, C.3, H.3.3, I.5.3
Helsinki 2009
University of Helsinki

**Identifying Meaningful Places**

Petteri Nurmi

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

PhD Thesis, Series of Publications A, Report A-2009-7
Helsinki, October 2009, 83 pages
ISSN 1238-8645
ISBN 978-952-10-5789-2 (paperback)
ISBN 978-952-10-5790-8 (PDF)

**Abstract**

Place identification refers to the process of analyzing sensor data in order to detect places, i.e., spatial areas that are linked with activities and associated with meanings. Place information can be used, e.g., to provide awareness cues in applications that support social interactions, to provide personalized and location-sensitive information to the user, and to support mobile user studies by providing cues about the situations the study participant has encountered. Regularities in human movement patterns make it possible to detect personally meaningful places by analyzing location traces of a user. This thesis focuses on providing system level support for place identification, as well as on algorithmic issues related to the place identification process.

The move from location to place requires interactions between location sensing technologies (e.g., GPS or GSM positioning), algorithms that identify places from location data and applications and services that utilize place information. These interactions can be facilitated using a mobile platform, i.e., an application or framework that runs on a mobile phone. For the purposes of this thesis, mobile platforms automate data capture and processing and provide means for disseminating data to applications and other system components. The first contribution of the thesis is BeTelGeuse, a freely available, open source mobile platform that supports multiple runtime environments.

The actual place identification process can be understood as a data analysis task where the goal is to analyze (location) measurements and to identify areas that are meaningful to the user. The second contribution of the thesis is the Dirichlet Process Clustering (DPCluster) algorithm, a novel place identification algorithm. The performance of the DPCluster algorithm is evaluated using twelve different datasets that have been collected by different users, at different locations and over different periods of time. As part of the evaluation we compare the DPCluster algorithm against other state-of-the-art place identification algorithms. The results indicate that the DPCluster algorithm provides improved generalization performance against spatial and temporal variations in location measurements.

**Computing Reviews (1998) Categories and Subject Descriptors:**
C.2.4 [Computer Systems Organization]:
Computer-Communication Networks - Distributed Systems
C.3 [Computer Systems Organization]:
Special-Purpose and Application-Based Systems
H.5.m [Information Interfaces and Presentation]:
Miscellaneous
I.5.3 [Pattern Recognition]:
Clustering

**General Terms:**
Algorithms, Design, Experimentation

**Additional Key Words and Phrases:**
location-awareness, place identification, spatial clustering, mobile systems, mobile platforms, ubiquitous computing, pervasive computing, mobile computing

# Acknowledgements

I am extremely grateful to my supervisor Patrik Floréen, who has supported me over the years and who has provided me the possibility to conduct research on topics that I have found personally interesting. The topics discussed in this thesis resulted from collaboration and discussions with Johan Koolwaaij, to whom I am extremely grateful. I am also very grateful to Wray Buntine whose knowledge about Bayesian statistics helped me enormously.

I thank all my current and former students, of whom Sourav Bhattacharya, Joonas Kukkonen and Eemil Lagerspetz have played an important part in the research towards this thesis. I also thank my friends and colleagues at the department and elsewhere, including (but not limited to) Jukka Suomela, Michael Przybilski, Taneli Vähäkangas, Niina Haiminen, Jussi Kollin, Petteri Kaski, Jukka Perkiö, Kasper Løvborg Jensen, Fabian Bohnert, Alexander De Luca, Novi Quadrianto, Gregor Broll, Christian Guttmann, Ákos Vétek, Péter Pál Boda and Esko Kurvinen.

I am grateful to my pre-examiners Rene Mayrhofer and Jeffrey Hightower for their comments and suggestions. I also thank everyone else who has given me advice and feedback on my work, including Greger Lindén, Petri Myllymäki, Marko Salmenkivi, Thomas Strang as well as numerous anonymous referees. I also thank the numerous people with whom I have coauthored papers during the last five years or so.

Last, but definitely not least, I am eternally grateful to my parents who have encouraged me in my studies and made everything possible.

# Contents

# Original publications

The thesis is based on the following original publications, which are referred to in the text as Articles I – IV. The articles are reprinted at the end of this thesis.

**Article I** Petteri Nurmi, Joonas Kukkonen, Eemil Lagerspetz, Jukka Suomela, and Patrik Floréen. BeTelGeuse – A Tool for Bluetooth Data Gathering. In *Proceedings of the 2nd International Conference on Body Area Networks (BodyNets, Florence, Italy, June 2007).* ACM, 2007.

**Article II** Joonas Kukkonen, Eemil Lagerspetz, Petteri Nurmi, and Mikael Andersson. BeTelGeuse: A Platform for Gathering and Processing Situational Data. *IEEE Pervasive Computing*, 8(2):49-56,2009.

**Article III** Petteri Nurmi and Johan Koolwaaij. Identifying Meaningful Locations. In *Proceedings of the 3rd International Conference on Mobile and Ubiquitous Systems (MobiQuitous, San Jose, California, July 2006.* IEEE, 2006.

**Article IV** Petteri Nurmi and Sourav Bhattacharya. Identifying Meaningful Places: The Non-Parametric Way. In *Proceedings of the 6th International Conference on Pervasive Computing (Pervasive 2008, Sydney, Australia, May 2008).* Lecture Notes in Computer Science 5013. Springer-Verlag, Berlin, 2008.

# Chapter 1

# Introduction

Mobile devices have fundamentally changed the way people interact with computing devices [30]. Nowadays people are no longer tied to a specific usage situation, but they can use computing services wherever, whenever and whatever they do. In mobile environments, the information needs of the user often depend on the user's situation [23, 105]. Hence, providing the appropriate information or assistance to the user requires taking into consideration the situation of the user.

Location is the most widely used source of situational information. Whereas other sources of situational information (e.g., activity or social context) are difficult to identify or measure, location information can be readily accessed [59]. Location also plays a fundamental role in our daily lives. For example, location information is widely used in human communication [116] and humans structure their daily activities around locations [48]. Location can also influence the user's information needs [37, 58, 76, 95, 105] or to give clues about other users' communication context [72, 87].

Contemporary mobile phones readily support at least one location technology (see Chapter 2). The location systems that mobile devices support typically provide location information as a pair of coordinates (e.g., latitude and longitude). However, humans do not refer to locations as a pair of coordinates, but using semantic expressions that are imbued with meanings, such as at home or in a library (see Sec. 3.1). Thus there is more to location than mere coordinates. The notion of place provides a way to represent location information that is consistent with the way people themselves refer to location information. Places are roughly defined as physical locations that are linked with semantical descriptions and meaningful activities (see Sec. 3.2). This suggests that place information could be used, e.g., in applications and services that support social interactions.

This thesis focuses on the process of providing place information to applications and services for mobile phones. The first two chapters of the thesis provide background information on topics that are relevant to the thesis and the original research contributions are discussed in the subsequent chapters. We begin in Chapter 2 by introducing the Global Positioning System (GPS) and the Global System for Mobile Communications (GSM), two commonly used technologies for providing location information to mobile devices. Chapter 3 describes human practices surrounding the use of location information in everyday situations and introduces the notion of place.

Chapter 4 discusses the process of providing place information from a system perspective. We describe mobile platforms, which are applications or frameworks that run on the mobile phone. For the purposes of this thesis, mobile platforms facilitate collecting suitable location data and providing information about places to applications, services and other system components. The chapter also introduces BeTelGeuse, an open source mobile platform[1] that has been developed during the research towards this thesis.

Chapter 5 shifts the focus to a data analysis perspective and surveys different approaches for identifying places from location measurements. The chapter also introduces the Dirichlet process clustering, a novel algorithm for place identification. Chapter 6 evaluates different place identification techniques, focusing on the accuracy and generalization performance of the techniques. The chapter also identifies weaknesses in current place identification algorithms and provides directions for future research. Finally, Chapter 7 summarizes the main contributions of this thesis, dicusses the limitations of the work and describes directions for further work on the topic.

## 1.1 Main Results of the Thesis

Articles I and II focus on mobile platforms and, in particular, the BeTelGeuse platform. The first version of BeTelGeuse, described in Article I, was designed to facilitate data collection from Bluetooth-enabled sensors. Since then we have extended the BeTelGeuse platform, e.g., by incorporating support for phone internal and Internet-based sensors, by building plug-ins that enrich the collected sensor data, and by providing additional mechanisms for accessing collected sensor data. The most recent version of BeTelGeuse is described in Article II and Chapter 4. Article II also presents a performance evaluation of the BeTelGeuse platform.

[1] Available from: `http://betelgeuse.hiit.fi`

Articles III and IV focus on algorithms for identifying places from location data. Article III introduces and compares four different algorithms. Two of the algorithms were designed for cell transition data, whereas the remaining two operated on coordinate data. Unfortunately these algorithms were sensitive to parameter values, which lead us to develop a novel place identification algorithm, the Dirichlet process clustering, that offers improved generalization performance. The Dirichlet process clustering algorithm is described in Chapter 5 and Article IV.

## 1.2 Contributions of the Author

In Articles I and II, the concept of BeTelGeuse is due to the present author and he has been responsible for leading the development team. The evaluation, write-up and illustrations are joint work.

All aspects of Article III are joint work with J. Koolwaaij.

In Article IV, the concepts and the main results are due to the present author; S. Bhattacharya has participated in the implementation and visualization.

# Chapter 2

# Location Systems

Enabling location-awareness requires technologies that provide information about the user's location. A large number of different location sensing technologies have been developed over the years, ranging from infrared sensing to satellite positioning systems such as GPS or Galileo[1]. Most location systems require some form of infrastructure investments and potentially also changes to the hardware of the device that is being located. For example, ultrasound or infrared systems require tags that the user carries around [115], whereas accurate network-based GSM positioning requires upgrading GSM cell towers with expensive location-measurement units [109].

Mass deployment of location-aware services requires location technologies that can be used on mobile phones without additional hardware. Current smart phones readily support GPS and GSM positioning. In the following sections we describe background information on these two technologies; for information about other location systems we refer to the survey in [51].

In comparison to GSM, the main advantage of GPS is that it provides more accurate location information. The main disadvantage of GPS measurements is that collecting them typically requires the user to carry an external GPS receiver with her. While increasingly many phones are equipped with integrated GPS receivers, high battery consumption of the receivers hinders using them for long term data collection [114]. In contrast to GPS, GSM can be used to provide location information also indoors and GSM can be used to provide location estimates without additional hardware. In terms of place identification, most algorithms for detecting places operate on GPS data, though also approaches that operate using GSM cell identifiers have been developed; see Chapter 5.

[1] `http://ec.europa.eu/transport/galileo/index_en.htm` [Retrieved: 2009-08-03]

## 2.1 Global Positioning System (GPS)

The Global Positioning System (GPS) is a satellite navigation system that was developed by the U.S. Department of Defense [35]. The first satellites were launched in 1970s and the system became fully operational in 1995. Originally GPS was developed for the needs of tactical bombers that require accurate three-dimensional position worldwide and that could only use passive receivers in order not to reveal their location to the enemy [46].

GPS is based on lateration, i.e., the idea that one's position can be determined given the distance to objects whose position is known [35]. The GPS architecture is based on a constellation of 24+ satellites[2] that orbit the earth. Each satellite knows its own orbital location and system time very accurately. The satellites regularly broadcast navigation messages that contain information, e.g., about the satellites orbital position and clock offset [79]. The signals that are broadcasted are relatively weak, but they can be heard if there are few radio frequency barriers between the receiver and the satellites. Accordingly, GPS measurements are mainly available when the user is outdoors, but measurements can be received also, e.g., inside wood frame buildings.

GPS receivers use time-difference-of-arrival measurements to determine their distance from satellites. If the receiver and satellite clocks are synchronized and there are no propagation delays, the distance from the satellite equals $c(t_r - t_s)$ where $c$ is the speed of light, $t_r$ is the system time of the receiver and $t_s$ is the system time of the satellite when the broadcast message was sent. Let $u$ denote the user (GPS receiver) and let $g$ denote a satellite. The range between the satellite and the user is given by the Euclidean distance between $u$ and $g$:

$$\rho_{u,g} = \sqrt{(x_u - x_g)^2 + (y_u - y_g)^2 + (z_u - z_g)^2}. \tag{2.1}$$

Knowing the range and location of (at least) three satellites defines a set of non-linear equations where the unknown variables correspond to the user's three-dimensional position. These equations can be solved, e.g., using non-linear least squares or Kalman filtering to yield an estimate of the receiver's position [70].

The formulation above assumes that the receiver and satellite clocks are synchronized and that the signals propagate without additional delays. In reality the receiver and satellite clocks contain errors and, e.g., ionospheric and tropospheric refractions, multipath effects and measurement

[2] Currently 31 satellites; for up-to-date information see `http://www.navcen.uscg.gov/navinfo/Gps/ActiveNanu.aspx` [Retrieved: [2009-07-01]


Figure 2.1: GPS estimates contain inaccuracies due to errors in pseudorange measurements (a) and satellite geometry (b,c).

noise delay the propagation of signals [36, 70]. Hence, the receiver can only calculate a biased estimate of the range. The biased range estimates are referred to as pseudoranges [31]. The basic pseudorange model can be written as follows:

$$r_{u,g} = \rho_{u,g} + c(\Delta t_u \quad \Delta t_g) + \epsilon_g. \tag{2.2}$$

Here $\Delta t_u$ denotes the clock offset of the receiver, $\Delta t_g$ denotes the clock offset of the satellite and $\epsilon$ is an error term that encapsulates other sources of error. The satellite clock offset can be approximated using information in the navigation messages, but the receiver clock offset must be solved from the pseudorange equations. The final set of equations thus contains four unknowns and requires information from a minimum of four satellites.

The accuracy of the estimated GPS position is proportional to the pseudorange measurement error, but it also depends on satellite geometry [36]. According to lateration principles, each distance measurement to a known reference point defines a circular curve and the position of the client is a point along this curve. When the distance measurements contain errors, the curve corresponds to a circular sector within which the client is located; see Fig. 2.1(a). When we combine measurements from multiple reference points, the intersection between the circular sectors defines the area where the client is located; see Fig. 2.1(b). The size of the intersection, and thus also the overall uncertainty in the position estimate, depends on the geometric relationships between the reference points. This is illustrated in Fig. 2.1(b) and Fig. 2.1(c). In the former the reference objects are almost orthogonal and the intersection is relatively small. In the latter example the

reference objects are closer and the resulting uncertainty in the estimates is higher.

The geometric dilution of precision (GDOP) is a metric that relates the pseudorange equations to an estimate of the goodness of satellite geometry. Let $A$ denote the matrix of partial derivatives of pseudoranges with respect to the unknown variables (longitude, latitude, altitude and clock offset) and define $Q = A'A^{-1}$, where $A'$ is the transpose of matrix $A$. The GDOP value is defined as the root of the trace of the matrix $Q$, i.e.,

$$\text{GDOP} = \sqrt{q_{11} + q_{22} + q_{33} + q_{44}}. \tag{2.3}$$

Rather than examining the goodness of all estimates, we can separate the different error components. These components are called DOPs (dilution of precision) and they cover a specific subset of the unknown variables. Commonly used DOP values include

$$\begin{aligned} \text{PDOP} &= \sqrt{q_{11} + q_{22} + q_{33}} \\ \text{HDOP} &= \sqrt{q_{11} + q_{22}} \\ \text{VDOP} &= \sqrt{q_{33}} \\ \text{TDOP} &= \sqrt{q_{44}} \end{aligned} \tag{2.4}$$

PDOP measures the overall dilution of precision in the position estimate, whereas the HDOP and VDOP measure horizontal and vertical dilution of precision. Finally, TDOP measures the dilution of precision in the clock offset estimates. Location-aware services typically require two-dimensional position information, which means that the HDOP value is the most relevant DOP value for our purposes.

The GPS satellite constellation has been designed to provide a good satellite geometry worldwide. However, tall buildings or other obstacles can block signals and decrease the accuracy of the location estimates. These situations can usually be detected from high dilution of precision values. As a general rule of thumb, with modern GPS receivers, measurements with HDOP values greater than 6.0 should not be considered due to potentially large error deviations; see, e.g., the experiments in [102]. In our case HDOP and satellite visibility information are used to filter out invalid GPS measurements from the place identification process; see Chapter 5.

When a GPS receiver is started or when it loses visibility of satellites, it must acquire information about the positions of satellites. The speed of the signal acquisition depends on when the receiver was last used and when it was last able to see sufficiently many satellites. When the receiver has no information about the satellites, the acquisition is called a cold

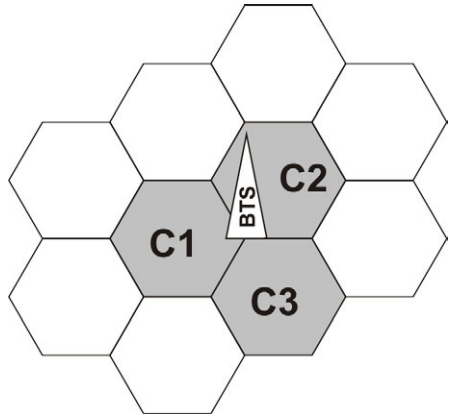



Figure 2.2: A simplified view of the GSM network architecture

start. With modern GPS receivers, a cold start typically takes around one minute. However, with older receivers a cold start can require up to 20 minutes; see, e.g., [19]. The situation where the receiver remembers its last location and it has coarse orbital information about the positions of the satellites (almanac data) is called a warm start. The time to position fix in a warm start is typically within 30 seconds. However, the acquisition time can be higher if the location of the receiver has changed from the previously known valid location. Finally, the situation where the receiver has accurate orbital data about the satellites is called a hot start. In a hot start, the acquisition typically takes only a few seconds.

## 2.2 Global System for Mobile Communications (GSM)

The Global System for Mobile Communications (GSM) is a worldwide digital cellular telephone standard. GSM was first deployed in 1992 and since then it has become the most widespread cellular system in the world with deployments in over 200 countries [85]. A simplified view of the GSM network architecture is shown in Fig. 2.2. The network is divided into base stations (BTS) and cells. Each cell has a unique identifier and it is served by one base station. One base station can serve multiple cells. The cells are grouped into clusters and cells belonging to the same cluster have the same location area identifier (LAI).

In addition to providing speech and data services, GSM supports positioning. Contrary to GPS, GSM signals can penetrate buildings and hence

GSM positioning works also indoors. The positioning is based on different signal measurements that can be made either on the client device or on the network side. Three main positioning techniques exist: cell identifier positioning, lateration and fingerprinting. In the following we discuss these techniques.

### 2.2.1 Cell Identifier Positioning

The cell identifier method is the simplest positioning algorithm for mobile phones. In the cell identifier method, the position of the client device is estimated using the coordinates of the base station to which the device is currently connected. If the exact coordinates of the base station are not available, the locations of the base stations can be estimated from empirical measurements; see, e.g., [21]. The accuracy of cell identifier positioning is relatively poor and depends on various factors such as cell size, cell density and environment characteristics. Trevisani and Vitaletti [109] have shown that the accuracy of this method is several hundreds of meters within densely populated areas and several kilometers within sparsely populated areas. The cell coverage areas typically overlap and location estimates can be improved using information from multiple cells. In the centroid method, the location of the handset is estimated as a weighted average of several base stations [69]. The cell identifier method can also be improved using timing advance (TA) measurements [109]. The TA is a discrete measure that gives rough estimates of the distance from the handset to the base station. One TA unit corresponds to approximately 500 meters and hence TA mainly helps positioning within large cells. While the accuracy of the cell identifier method is relatively poor, the advantage of the method is that it does not require any changes to existing mobile terminals or to the network infrastructure.

### 2.2.2 Lateration

Lateration is an extension of the cell identifier method that estimates the distance or angle between the mobile station and base stations. Each estimate defines a circle or hyperbola along which the client is assumed to be located. Measurements from multiple base stations are used to resolve ambiguity in the individual estimates.

Similarly to GPS, signal propagation time can be used to estimate the position of the client. When the clocks of the base station and the mobile receiver are synchronized, measuring the time it takes for a signal to traverse from the mobile client to the base station or vice versa is sufficient.

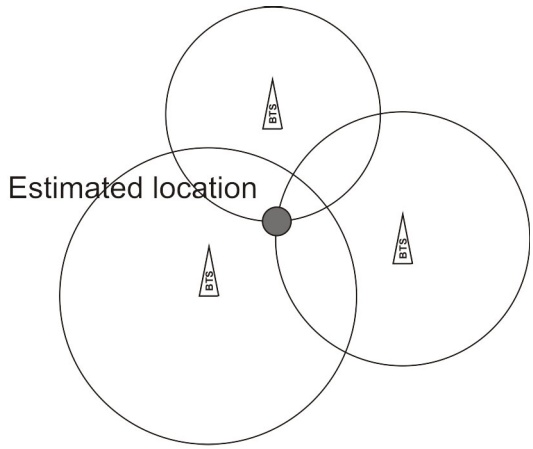



Figure 2.3: Example of distance-based lateration. Each estimated distance defines a circle and the interaction of three circles can be used to estimate the location of the handset unambigiously.

Otherwise the estimates must be based on round-trip times. Radio signals travel at the speed of light so by knowing the time the distance between the handset and base station can be estimated. Each distance measurement constraints the position of the mobile device along a circular locus centered around the base station. The ambiguity in the location estimates can be resolved by estimating distances to multiple base stations and using the intersection of the loci as the location estimate; see Fig. 2.3.

Distances can also be estimated using time-difference-of-arrival (TDOA) measurements [34]. TDOA measures arrival time differences between pairs of base stations. Each TDOA measurement defines a hyperbolic locus and multiple measurements can be used to resolve ambiguity in the estimates. Also observed signal strengths can be used to estimate distances. Related techniques include using angle of arrival or combination of angle and distance measurements to constrain the location estimates; see, e.g., [34, 85].

The accuracy of lateration depends on the accuracy of the distance and angle measurements. In practice, deriving accurate estimates is complicated due to a wide variety of random effects. For example, buildings and other obstacles cause signal decay and multipath refractions, other radio devices can cause interference that corrupts measurements, and so forth [85]. Furthermore, accurate time or angle measurements require costly upgrades to the network infrastructure, which makes these approaches unattractive.

### 2.2.3 GSM Fingerprinting

Instead of modeling radio propagation, fingerprinting exploits spatial variations in observed signal strengths for positioning. Fingerprinting operates by creating a database that maps pre-recorded network measurements with known locations. When the client needs to be positioned, the current network characteristics are compared to the measurements in the database and the position of the client is estimated, e.g., calculating a weighted average of the coordinates from the top $k$ measurements.

Fingerprinting is not limited to GSM, but it can be used with any radio technology (e.g., GSM, WLAN, FM radio). Fingerprinting was originally developed for indoor positioning and the first approaches used observed signal strengths from WLAN access points [8]. Typically the fingerprints that are used consist of radio source identifiers and observed signal strengths. However, also other types of measurements are possible. For example, the RightSPOT system operates on radio channel identifiers that are sorted based on signal strength [65], whereas hyperbolic fingerprinting operates using signal strength differences between pairs of radio beacons [61].

In GSM fingerprinting, the fingerprints typically consist of one to six cell identifiers and observed signal strengths for each cell. The use of multiple cells can improve positioning accuracy [21], though many handsets restrict the information to the cell the device is currently connected to. Further improvements can be obtained using wide signal fingerprints that contain readings from additional cells that are too weak for communication purposes [86, 112].

# Chapter 3

# From Location to Place

GPS and GSM positioning return location information in coordinate form. This type of location information is useful for a variety of applications and services. For example, disaster management can use coordinates to locate an emergency number caller [100]. Location-based games can change the state of the game according to the user's location in the physical world [14]. Mobile guides can provide information about restaurants, movie theaters etc. that are nearby [11, 64] and navigation systems [12] can provide instructions to reach the destination. However, as we discuss in Sec. 3.1, people themselves do not refer to locations using coordinates, but using semantic descriptions (at home, at the supermarket, at an opera performance etc.). Moving from coordinates to representations that are consistent with the way people themselves refer to location information can enable novel and more powerful opportunities for social coordination and interaction. In this thesis we focus on the notion of place, which aims to provide such a representation. Places can be roughly defined as a combination of a physical location, meanings and activities that relate to the physical location; see Sec. 3.2.

## 3.1 Users and Location Information

According to ethnomethodologist tradition, the design of technologies can be informed using observations about everyday practices; see, e.g., [33]. Accordingly, the design of location-aware applications that support social interactions can benefit from observations about how location information is used within everyday practices. Following this tradition, various studies have investigated location disclosure during mobile phone calls. For example, Laurier [71] analyzed how mobile workers talk about location while

traveling by car. The study indicated that mobile workers actively used location information and that the main use for location information was to establish a shared context with the other participant. Arminen [5] analyzed mobile phone calls within different contexts and found further uses for location information. According to Arminen, location can be used as a cue of interactional availability, as a precursor for mutual activity, as part of an ongoing activity (e.g., to coordinate), as a social fact or as an emergent feature that bears relevance to the current activity.

While location is widely used during conversations, it is seldom used in geographic terms, but it is made relevant as part of the joint activities in which the participants are involved [5]. Weilenmann and Leuchovius [116] studied the nature of information that is disclosed during phone calls and their results suggest that the type of location information that is disclosed depends on the role of the activity and the mutual context between the people communicating. For example, during coordination activities, location is disclosed in reference to what it means to move between locations, whereas familiar terms are used for other purposes (e.g., I'm at home).

Another important question is what kind of locations people name. This issue has been investigated using diary studies and data gathered from mobile applications that support labeling of locations [72, 122]. The results of the studies have been rather consistent and indicate that people tend to assign labels to both private (home, work) and public locations (library, train station). Furthermore, some labels relate to a shared context (e.g., referring to a friend's home or a regular place to meet friends) whereas some labels are related to a specific activity (e.g., gym, swimming hall).

In most studies, location disclosure has been investigated within a specific social setting (e.g., between friends or family). Consolvo et al. [24] investigated how the nature of the social relationship influences the willingness to disclose location and the granularity of information that people are willing to disclose. They found that people typically formulated their location information in a fashion that they though was useful for the other person. Typically participants returned specific location information and vague or blurred expressions were rarely used. The social relation between the persons also played a major role. While people were willing to disclose their location information practically always to significant others and to family, they were not willing to disclose location to colleagues outside work hours. Moreover, workers were even more hesitant about disclosing their location information to their managers.

According to the interactionist view of context [32], the use of context relates to the practices of the people and these practices change dynamically

over time as people invent new uses and become more familiar with the technology and its possibilities. Exposing people to novel technologies can thus result in novel ways of using context information. Oulasvirta et al. [87, 88] investigated the role of location as an availability cue by augmenting the contact book and recent calls view of a smartphone, e.g., with information about the location and phone profile of a contact. Location and profile information were found to be important cues for determining availability when people knew each other, but location information was not as useful for determining the interruptability of a stranger.

## 3.2 The Notion of Place

Place is a word that occurs frequently in daily communication and that is imbued with meanings of common sense. People talk about place in a variety of contexts, which suggests the notion of place pervades various aspects of daily life and that finding a single definition can be difficult. This is also evidenced by the variety of research fields that have investigated (some aspects of) the notion of place. For example, architects and urban planners try to evoke a sense of place, ecology and ecosystems management talk about ecological places and bio-regions, and artists and writers try to reconstruct places in their work [27].

The definitions of a place that are most relevant for computer science originate from the field of humanistic geography where place is considered an experiential entity [27, 63, 110]. For example, Relph [97] defines a place as a combination of a physical setting, the activities supported by the place and the meanings attributed to a place; whereas Tuan [111] defines places as spaces that are embodied with meanings. Note that, while places relate to a space, the existence of a physical space is not required but also virtual spaces exhibit place-related behavior. For example, people posting to a particular newsgroup adopt the norms of the specific group and people interacting in virtual environments form small-scale communities that adopt their own behavioral norms [50].

Meaningfulness is central to the definitions of place, yet nothing is said about what makes a place meaningful. According to Gustafson [49], the meanings can be related to a three-pole model where the poles correspond to self, others and environment. Meanings associated with places can relate to one of the poles or relationships between multiple poles. Also other aspects influence the meanings attributed to places. For example, Krämer [63] shows that places can be categorized into generic place-types based on their specificity, functionality and privacy.

Place information can be used in mobile applications in various ways. As discussed, place information can be used to support awareness by providing cues about the user's generic situation and interruptability. Place information can also be used to support place-centered information delivery. Jones et al. [58] investigated how places influence user information needs. They found that the information that people need in a place depends on how often the user visits the place and how stable the information is. For example, a user that takes the same train every morning does not normally need information about train schedules (stable information) unless there is a major delay (dynamic information).

Places correlate strongly with location and time information. As part of a study on human mobility patterns, Gonzalez et al. [48] showed that humans visit a relatively small number of locations during a day. This indicates that the activities of the users are necessarily structured around locations where humans spend significant amount of time. Lehikoinen and Kaikkonen [72], on the other hand, have shown that the time the user stays at a location is an important factor that influences whether the user is likely to label the location or not. However, users are unlikely to consider traffic jams or traffic lights meaningful, even if they are visited often and for long periods of time. In Chapter 5, we show how time and location information can be used to accurately determine meaningful places from user's location trajectories.

# Chapter 4

# Mobile Platforms

From a system perspective, the move from location to place requires interactions between location systems, algorithms that identify places from location measurements and applications and services that utilize place information. These interactions can be facilitated using a data collection platform that automates data capture and processing, and provides means for disseminating data to applications and other system components. This chapter introduces data collection platforms for mobile devices and describes BeTelGeuse[1], a mobile platform that has been developed as part of the research towards this thesis, and that is described in Articles I and II.

## 4.1 Survey of Existing Mobile Platforms

Frameworks are defined as computational environments that are designed to simplify application development and system management for specialized application domains [16]. Mobile platforms are frameworks that run on a mobile device. Mobile platforms can be categorized based on the nature of data they collect. First of all, platforms that support collecting objective data log different types of sensor information, e.g., about user interactions, device state, location and the user's environment. Platforms that collect only objective data are usually designed to support application development and, for this reason, these platforms usually also provide interfaces for disseminating data to other system components. These platforms usually also provide some form of support for automatically refining the sensor data, e.g., in the form of activity recognition (see, e.g., [67, 77]) or place identification. The second class of platforms focuses on collecting subjec-

[1] BeTelGeuse is freely available under the GNU Lesser General Public License (LGPL) from the project website: `http://betelgeuse.hiit.fi/`

tive self-reporting data from the user. The main goal of these platforms is to support field studies in mobile human-computer interaction. Most platforms that collect subjective data also collect objective data. However, contrary to platforms that focus on objective data, these platforms tend to have limited support for using sensor information in applications and services. In the following we describe existing platforms in these two categories. We limit our survey to platforms that run on a mobile phone and support, in addition to collecting sensor data, automatic processing of sensor data or collection of subjective data. Thus middleware, such as Muffin [118], and wearable platforms, such as Mobile Sensing Platform (MSP) [22]), are excluded from the following discussion.

### 4.1.1 Platforms and Toolkits for Objective Data Collection

Various toolkits that focus on specific types of data have been proposed. One example is the Place Lab open source toolkit for location sensing [53, 69, 104]. The architecture of a Place Lab client consists of three kinds of components: spotters, mappers and trackers. Spotters are modules that are responsible for collecting information about radio beacons in the user's vicinity. For example, a WLAN spotter would periodically scan for available WLAN access points. Mappers, on the other hand, are responsible for maintaining radio map information on the device. In the basic form, the radio maps consist of radio beacon identifiers and estimated locations for each beacon. Additional information can contain learned radio propagation models, antenna altitude information etc. Finally, trackers are responsible for calculating location estimates for the clients using the information stored by the mappers. Place Lab supports various platforms and it can be used on laptops, mobile phones and PDAs. Another example of a toolkit is the Context Recognition Network (CRN) [9, 10], which enables creating distributed, multimodal activity-recognition systems. The CRN supports collecting data from distributed sensors and it provides a collection of ready-to-use signal processing algorithms. However, the CRN supports only the Posix operating system and thus currently iPhone is the only mobile phone where the CRN can be used.

ContextPhone [92, 93] is a platform that collects various sensor data, provides system services that facilitate building and running custom applications, and provides an abstraction to the device's communication mechanisms. The sensor data that ContextPhone collects consists of location data (GSM identifier, Bluetooth GPS), communication behavior (calls, sent and received SMS), physical environment (nearby Bluetooth devices, optimal marker recognition) and user interaction data (active application, idle

or active status). ContextPhone also automatically detects places from GSM cell identifier data; see Sec. 5.2.5. In terms of system services, ContextPhone provides support for automatically launching applications and background services. ContextPhone also contains a watchdog mechanism that monitors running applications and restarts them if they have crashed. The main limitation of ContextPhone is that it only supports Nokia S60 smartphones. A related platform is the ContextWatcher [62], which also supports place identification and runs on Nokia S60 smartphones. The main difference between ContextPhone and ContextWatcher is that ContextPhone is a background service that is automatically started, whereas ContextWatcher is an application that the user must manually launch.

### 4.1.2 Platforms for Subjective Data Collection

Mobile phones are used in a wide variety of everyday situations [92, 106], which makes it possible to use mobile phones to collect rich data about the thoughts, feelings and behaviors of humans in a wide range of everyday situations. Experience sampling is a study technique that uses a signaling device to elicit subjective self-report data from participants over a longer period of time [28, 41]. Initially experience sampling studies were conducted using a pager and a paper-based self-report, but improvements in the capabilities of mobile phones have made it possible to conduct experience sampling studies using mobile phones [41, 55, 56, 92]. Experience sampling can also be used to study how people interact with mobile devices and applications [89, 105], and to evaluate mobile applications and services [25].

The benefits of subjective data collection have resulted in various mobile platforms that support collecting subjective data. While some of these tools support collecting both sensor data and subjective data, the focus of all of these platforms has been on supporting experience sampling studies. As a consequence, these platforms provide scant support for utilizing sensor data in applications. The first tools were designed for PDAs, but contemporary tools are exclusively targeted at mobile phones. Two examples of early tools are the Experience Sampling Program (ESP) [41] and the Context-Aware Experience Sampling tool (CAES) [55, 56]. The main difference between the two tools is that CAES supports collecting sensor data whereas ESP does not. CAES also enables event-based prompting, i.e., showing the questionnaires in pre-defined situations. The main limitation of these tools is that they were not designed to run on the user's personal devices. As a consequence, the tools require exclusive access to the device and may interrupt the user at inappropriate times [43, 54].

More recent platforms support also collecting objective data. An example is the Xensor [54], which is an extensible platform that runs on Windows Mobile smartphones. Xensor supports collecting data, e.g., about the user's situation (various Bluetooth-enabled sensors: GPS, accelerometer, heart rate monitor), device data (remaining battery, GSM information, WiFi access point information) and it also provides a socket interface that allows logging customized application data. Subjective data can be collected using interval-based experience sampling. The Xensor platform has been used, e.g., to study the influence of contextual factors on availability inferences [107].

MyExperience is another open source platform that supports logging sensor data and capturing subjective user data [43]. The sensor data that MyExperience collects from the phone is richer than what the Xensor collects. Among other things, MyExperience collects usage data (e.g., phone calls or application usage), user context information (e.g., calendar appointments) and environmental data (e.g., nearby devices or external GPS receiver). Subjective data is collected using questionnaires that can be trigged at certain intervals (i.e., interval-based experience sampling) or whenever a certain pre-condition is met (i.e., event-based experience sampling). MyExperience is implemented using a sensor-trigger-action model. The sensors are abstractions of hardware and software sensors which collect the objective data. The triggers, on the other hand, define an event mechanism, which allows specifying when to send data to other components or to perform an action. The actions themselves are code snippets that are executed on the phone, whenever the corresponding trigger condition is met. Similarly to the Xensor, MyExperience only runs on devices with the Windows Mobile operating system.

## 4.2 BeTelGeuse

BeTelGeuse is an open source mobile platform that has been developed during the research towards this thesis. The first version of BeTelGeuse was developed between August 2006 and February 2007. At that time, mobile phones had limited support for integrated sensors and they provided limited programming interfaces. However, Bluetooth support was becoming standard and many phones supported accessing Bluetooth using Java programming interfaces. Bluetooth-enabled sensors were also increasingly available on the market (e.g., GPS receivers, accelerometers and heart rate monitors). Because of these reasons, the first version of BeTelGeuse was developed using mobile Java and it focused on facilitating data collection

from Bluetooth-enabled sensors; see Article I.

Since our first version, the capabilities of mobile phones have rapidly increased. Contemporary mobile phones have ample processing power and storage capacity, which enables performing more processing directly on the phone. Sensors are increasingly integrated into mobile phones, for example the Nokia N95 contains an integrated GPS receiver, as well as tilt and three-dimensional acceleration sensors. Mobile data connectivity technologies have become faster, and relatively cheap flat rate subscription fees combined with improvements in mobile web browsers have resulted in widespread usage of mobile Internet. Mobile operating systems have opened up, which enables users to install custom third-party applications to the phone. Finally, the programming interfaces of mobile devices have improved, which has made it easier to access resources and sensors on the phone. These are among the issues that have influenced the latest version of BeTelGeuse, which includes support for accessing data from phone internal sensors, different mechanisms for accessing collected sensor data, and plug-ins that augment the sensor data by providing additional information or by inferring higher level context information. The BeTelGeuse platform supports different platforms and we have tested it on Nokia and Sony Ericsson mobile phones, desktop computers running Linux or Windows operating systems, as well as PDAs running the Microsoft Windows Mobile operating system. In the following we briefly describe the different components in the BeTelGeuse architecture. More detailed information, including a list of supported sensors and a performance evaluation, can be found in Article II. The datasets that are used in Chapter 6 to evaluate place identification algorithms have all been collected using BeTelGeuse and we are currently in the process of integrating place identification support into BeTelGeuse.

### BeTelGeuse Architecture

BeTelGeuse's high-level system architecture has been inspired by the microkernel architecture pattern. We have a separate core which offers a minimal set of functionalities that are needed to run the tool. The core also defines interfaces for components that provide extended functionalities. The core consists of a blackboard and three communication modules. The blackboard can be thought of as a shared message board where components can write new messages and read messages from other components [117]. The communication modules, on the other hand, encapsulate the communication mechanisms of the mobile device and provide mechanisms that enable applications to obtain sensor data.

The interfaces for obtaining sensor data differ across phone manufactur-

ers, which makes it difficult to have a single implementation of the sensor interfaces. In BeTelGeuse, the core defines only an interface for sensors and the actual sensors are abstracted as context parsers that are outside the core[2]. BeTelGeuse can also be extended with plug-ins that provide extra functionalities, e.g., high-level context inference or support for experience sampling studies. In the following we describe the different types of components.

**Blackboard:** The BeTelGeuse blackboard acts as a communication hub for the communications between different system components. Java components that run on the phone can communicate with the blackboard using direct method calls, whereas other components (local or external) can use a socket connection. The communications with the blackboard use a Simple Sensor Interface-like protocol[3] (SSI), which supports sending data packets as well as command messages that modify the current system configuration. The blackboard itself is data-type agnostic and components that read data from the blackboard are responsible for interpreting the data. By default, the interactions with the blackboard follow a publish-subscribe paradigm where the blackboard notifies the appropriate components when new data is available. The blackboard supports events, which enable refining when to send data.

**Context Parsers:** Context parsers are responsible for collecting data from sensors and for making the data available to the blackboard. Each parser can operate in streaming or periodic mode. In the streaming mode, data is continually read from a sensor, whereas in the periodic mode the sensor is polled for data at predefined intervals. BeTelGeuse supports collecting data from (i) phone internal sensors, (ii) external Bluetooth-enabled sensors, and (iii) sensors that provide data from the Internet. Developers can add new sensors to BeTelGeuse or they can extend the functionalities of existing parsers. The parsers for phone internal sensors need to be implemented using native code (e.g., Python S60 or Symbian C++ for Nokia S60 devices, and C# for Windows Mobile devices), whereas Bluetooth sensors can be implemented using Java. Internet sensors can be implemented either as plug-ins (typically Java) that pull data from the Internet or as services that run on external web servers and push data to the BeTelGeuse blackboard.

[2] A small set of parsers for common Bluetooth-enabled sensors is included in the core.

[3] `http://en.wikipedia.org/wiki/Simple_Sensor_Interface_protocol` [Retrieved: 2009-08-04]

**Communication Modules:** The current implementation of BeTelGeuse contains three communication modules: the Bluetooth manager, the data transmitter and the mobile HTTP server. The Bluetooth manager is responsible for scanning the device's Bluetooth environment and for creating and managing connections to Bluetooth-enabled sensors. The data transmitter is responsible for synchronizing sensor data with remote storage and for making the data available to external components. Similarly to the blackboard, the data transmitter supports events that can be used to define when to send data to the server. Finally, the mobile HTTP provides a callback mechanism that enables applications running on the mobile device's browser to access sensor data.

**Plug-Ins:** We have currently two plug-ins for BeTelGeuse. The location plug-in provides position information to the device using GSM fingerprinting, whenever GPS signal is not available, and it also provides semantic information about nearby locations that users have tagged; see [17] for information about the latter. The second plug-in, the activity plug-in, provides information about the user's activity based on accelerometer data [83]. We are also currently developing an experience sampling plug-in that provides support for running user studies with BeTelGeuse.

# Chapter 5

# Algorithms for Place Identification

Place identification can be understood as a data analysis task where the goal is to analyze (location) measurements and to identify areas that are meaningful to the user. In this section we first describe the place identification process and introduce techniques that are used in the different phases. After this we review existing place identification algorithms and describe the Dirichlet process clustering algorithm, a novel place identification algorithm that has been developed as part of the research towards this thesis, and that is described in Article IV. The Dirichlet process clustering provides improved generalization performance and is less sensitive to parameter values than the algorithms that we have developed in our earlier work, described in Article III.

## 5.1 The Place Identification Process

The steps of the place identification process are shown in Fig. 5.1. Four of the phases focus on analyzing the location measurements, whereas one phase, labeling, focuses on associating semantics with location information. The operations in the first data analysis phase, data preparation, depend on the particulars of the underlying location sensing technology and are common to all algorithms. The operations in the other analysis phases, preprocessing, cluster analysis and post-processing, are specific to the used place identification algorithm. The labeling step, on the other hand, is often considered the final step of the place identification process, but it can also be performed before data analysis. In the following we discuss each of the phases in more detail.

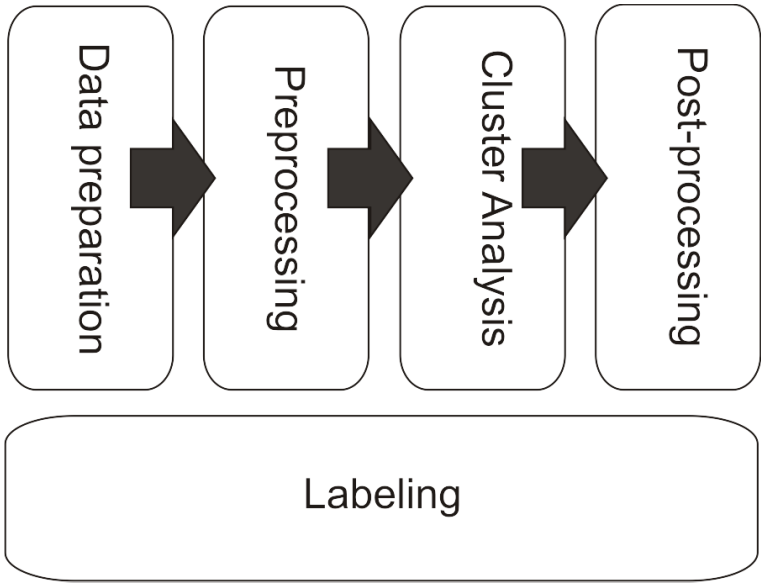



Figure 5.1: The place identification process.

### 5.1.1 Data Preparation

The data that we use consists of timestamped GPS measurements (see Sec. 6.1.1). In the data preparation phase we remove invalid measurements and transform the data into (`latitude, longitude, duration`) tuples where the duration values indicate the time the user has spent at each location. We also clean the data by removing invalid measurements that are caused by a warm start of the GPS receiver (see Sec. 2.1).

#### Duration Estimation

Most place identification algorithms use information about the time the user has spent at a location to identify meaningful places. As our data is collected non-continuously and as the data sampling rate varies, we need to perform some processing steps to estimate the time the user has stayed at each location. Our first processing step is to classify the points as valid or invalid. We consider a measurement valid, if the GPS receiver is able to see at least four satellites and if the HDOP value is below 6.0 (see Sec. 2.1). Otherwise the measurement is considered invalid. Most of the invalid measurements are from areas where the user has stayed indoors, but we also observed some inaccurate measurements; see Fig. 5.2. After classifying the points, we segment the data into sessions. The segmentation


Figure 5.2: Illustrating the notions of valid and invalid points. The left-most picture contains all measurements we have collected from Tokyo, Japan. The picture indicates that there are many invalid measurements which appear as straight lines originating from a frequent location. From the picture in the middle we have removed points where less than four satellites were visible and from the right-most picture we have removed points for which the HDOP value exceeds 6.0.

is necessary to ensure that missing measurements have no influence on the duration estimates. In our case data may be missing for various reasons. For example, as the data collection was based on voluntary participation, the participants might choose not to log data from a particular area. Other reasons include participants forgetting to start the data logging application and the mobile device or the GPS receiver running out of battery.

Similarly to segmentation algorithms used, e.g., in web log analysis [101] and driver trip analysis [44], our segmentation uses a threshold on the time difference between successive location measurements. If successive location measurements are at least 8 minutes apart, we assume they belong to different sessions. This threshold was selected based on two constraints. Firstly, Bluetooth scans require at least two minutes due to limitations in the J2ME Bluetooth API [81]. Secondly, many place identification algorithms use ten minutes as a threshold for detecting meaningful locations and thus the threshold value should be below 10 minutes to ensure that missing measurements cannot result in erroneous place detections.

In the actual duration estimation phase we consider each measurement in turn and compare it against the previous measurement. We only compare points that belong to the same session. The action to perform depends on whether the current and previous measurement are valid or invalid:

- **Current and previous point valid:** When both the current and the previous measurement are valid the user is outdoors. In this case

we compare the measurements and merge them if they are the same[1]. Otherwise we use the mode of the sampling rate as the duration estimate for the previous point.

- **Current point invalid, previous point valid:** When GPS connectivity is lost, we store the last seen valid point. If the session ends before a new valid point is seen, we use the mode of the sampling rate multiplied by the counter value as the estimate for the previously seen valid point.
- **Current point valid, previous point invalid:** If the points are the same, we merge them and use the difference in timestamps as the duration estimate. Otherwise we use the mode of the sampling rate to estimate the duration of the last seen valid point.
- **Current point invalid, previous point invalid:** When both measurements are invalid, we do nothing.

Most duration values are estimated using the difference in timestamps between successive measurements. When successive measurements are valid, there is usually some fluctuations in the measurements and we cannot evaluate exactly the time the user has stayed at a location. To reduce influence of sampling rate variations, in this case we estimate the duration using the mode of the sampling rate. In the processing phase we also discard invalid measurements. Accordingly, the final data set contains the coordinates of the valid measurements and a duration estimate for each point.

### Data Cleaning

Data cleaning (also known as cleansing or scrubbing) refers to the process of detecting and correcting errors and inconsistencies in data [94]. In place identification, the main source of errors is the used positioning technology. As all of our data has been collected using GPS, in this section we focus only on handling errors in GPS measurements.

The most common sources of errors in GPS measurements are signal shadowing and lack of GPS signal. These errors can be handled simply by examining the number of satellites and the estimated horizontal error, i.e., the HDOP value. In our case these measurements are removed in the duration estimation phase; see above. Another potential source of errors

[1] We consider two measurements the same, if the latitude and longitude coordinates are exactly the same. Alternatively a small radius threshold can be used to reduce fluctuations caused by uncertainty and errors in the location estimates. The latter approach is used in the algorithm of Ashbrook and Starner; see Sec. 5.2.1.

is related to GPS receiver warm starts, i.e., when the receiver is restarted after it has been off for a longer period of time and it has lost some of the orbital data that is used to estimate locations; see Sec. 2.1. In this case the first measurements are based on the receiver's last known position and coarse orbital parameters. If the receiver has moved significantly from the last known position, the first estimates can be in the wrong location. This can cause a sudden jump in the estimated location when the receiver obtains accurate orbital data from the satellites.

To correct errors due to receiver warm starts, we first use an outlier detection algorithm to detect jumps in the measurements. The simplest way to detect outliers from GPS measurements is to look at velocity information. We considered a measurement an outlier if the user's velocity exceeds 100 meters per second (360 km/h). As our velocity calculations are approximate (see Sec. 5.1.2) we selected a high threshold value to ensure that measurements would not be wrongly detected as outliers.

The outliers correspond to measurements that precede the point where the GPS receiver obtains accurate orbital data. Accordingly, the outliers define the last point that should be removed. To remove all invalid measurements, we combine the outlier detection with the session segmentation algorithm described in Sec. 5.1.1 so that points belonging to the same session and preceding the outlier point are also removed.

### 5.1.2 Preprocessing

Data preprocessing refers to tasks that are performed on the data before analysis [40]. We have made a distinction between the tasks that are common to all algorithms and the tasks that the individual algorithms perform on the data before analysis. In this section we focus on the latter issue and introduce velocity based pruning, which many place identification algorithms use to filter measurements.

#### Velocity Based Pruning

Areas where the user moves fast typically correspond to commuting and are unlikely to be meaningful to the user. This suggests that velocity information could be used to remove data from non-meaningful areas. Removing redundant data reduces the needed computations and potentially improves the resulting clustering accuracy. We approximate the actual velocity by considering the distance (in meters) and time (in seconds) between successive measurements. The velocity values are calculated during the duration estimation phase; see Sec. 5.1.1. While the duration estimates are not nec-

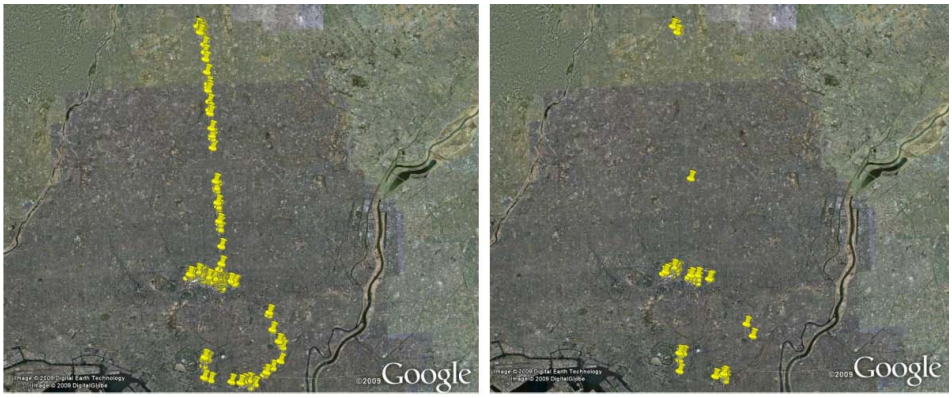



Figure 5.3: Illustration of the use of velocity information to prune data. The data in the example was collected in Tokyo, Japan. The original data set is shown on the left and the pruned data on the right. Most of the removed measurements correspond to points where the user was traveling by train.

essarily based on timestamps, velocity values are estimated using differences between timestamps.

The use of velocity information to prune data is illustrated in Fig. 5.3. In the example commuting traces have been removed from the data and the pruned data gives a better indication of the potentially significant areas. The figure also illustrates a potential pitfall as some of the unpruned points actually correspond to intermediate train stations. Accordingly, using only velocity and time information can leave areas that are insignificant, but where the user has stayed for a longer period of time; see Sec. 6.2.

### 5.1.3 Cluster Analysis

Cluster analysis focuses on detecting hidden groups, or clusters, among a set of objects [18]. Cluster analysis is the main phase in the place identification process. In place identification, the clusters typically correspond to frequently visited locations, or candidate places. The post-processing phase, discussed in the next section, then attempts to separate the meaningful clusters, i.e., places, from the non-meaningful ones.

Clustering can be performed using sequential or batch algorithms. Sequential algorithms analyze data one point at a time as new measurements arrive, whereas batch algorithms analyze data in larger chunks. Note that, while some place identification algorithms use sequential clustering, the actual place identification usually operates in a batch mode. The reason

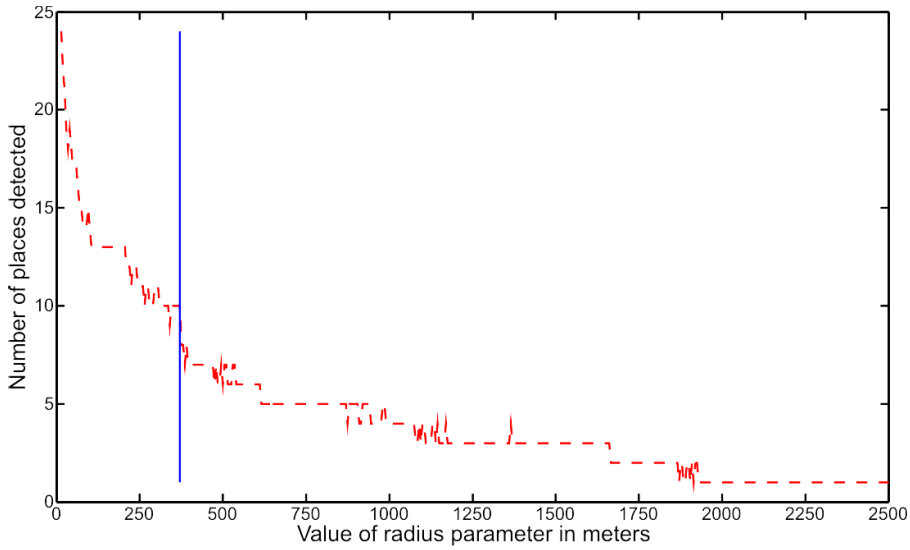



Figure 5.4: A plot of the number of places that the algorithm of Ashbrook and Starner detects as the radius parameter is varied. The solid line corresponds to the automatically selected knee value.

for this is that sequential clustering algorithms typically rely on various parameters and finding the optimal parameter values requires repeating the clustering with different parameter values. The cluster analysis step is discussed in Sec. 5.2. Next we introduce the scree criterion, a popular technique for determining optimal parameter values.

### The Scree Criterion

The scree criterion is a subjective method where the experimenter manually determines the appropriate parameter values. The idea in the scree criterion is to investigate graphically how variations in parameter values affect the quantity of interest, e.g., model fit or number of meaningful places. Originally the scree criterion was designed for multivariate statistics where it was used to determine the number of components in factor analysis [20]. In place identification, the scree criterion can be used, e.g., to determine the optimal value of the radius parameter for radius-based algorithms; see Sec. 5.2.1.

To illustrate the use of the scree criterion, in Fig. 5.4 we have plotted the number of places that the algorithm of Ashbrook and Starner (see Sec. 5.2.1) finds from the Buenos Aires dataset (see Sec. 6.1.1) as the radius parameter is varied. The figure does not indicate any clear cutoff points,

though the steepest point in the plot is at around 350 meters. As the example illustrates, applying the scree criterion in place identification can be difficult and the resulting interpretations can be highly subjective. For example, if we are told that the correct number of places in the Buenos Aires data set is six, we could argue that the knee point is at around 600 meters. On the other hand, if we are told that the number of places is eleven, we could argue that the knee point is at around 300 meters. To reduce biases from subjective interpretations, an objective criterion for selecting the knee point is needed.

From a mathematical point of view the knee point corresponds to a point where the slope of the function we are investigating changes significantly. Ashbrook and Starner [7] have suggested a way to automatically determine the knee point. In their scheme, the values of the function are examined from right to left. For each point, we calculate the mean of the next $n$ points to the right of the current point. If the mean point exceeds a predefined threshold $\psi$, the current value is selected as the knee point. In our experiments we have used this scheme for all algorithms that use the scree criterion. As the parameter values we use $n = 15$ and $\psi = 1.5$. The solid line in Fig. 5.4 indicates the scree point that this scheme selects for the Buenos Aires data.

### 5.1.4 Post-Processing

The post-processing phase refines the results of cluster analysis using spatial and temporal information. In this section we first discuss the merging of clusters, after which we discuss the use of temporal and spatial information to remove clusters that are assumed non-meaningful.

#### Cluster Merging

Variations in location measurements easily cause situations where multiple places are detected around a relatively small spatial area. In this kind of cases we might want to merge the different places into a larger place, especially if the activities or temporal patterns associated with the places are similar.

The simplest way to detect whether two clusters should be merged is to calculate the distance between the cluster means and merge the clusters, if the difference is smaller than a predefined threshold. For example, the algorithm of Kang et al. (see Sec. 5.2.1) uses a fixed threshold $d$ to detect clusters, and it merges clusters that are within distance $d/3$ from each other.

An alternative which considers also the spatial and temporal distribution of data near the places is to use the joint Kullback-Leibler divergence. The Kullback-Leibler divergence measures distance between two probability distributions $p$ and $q$ and it is defined as follows [26]:

$$D_{KL}(p||q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx. \tag{5.1}$$

As Kullback-Leibler divergence is not symmetric, the joint Kullback-Leibler divergence is defined as the sum $D_{KL}(p||q)+D_{KL}(q||p)$. When both $p$ and $q$ are modeled using Gaussian distributions, the Kullback-Leibler divergence can be calculated analytically using the formula:

$$D_{KL}(p||q) = \frac{1}{2} \log \frac{|\Sigma_p|}{|\Sigma_q|} + tr\left(\Sigma_p^{-1}\Sigma_q\right) + (\mu_p - \mu_q)\,\Sigma_p^{-1}(\mu_p - \mu_q)' - d. \tag{5.2}$$

Here $\mu_p, \mu_q, \Sigma_p$ and $\Sigma_q$ are the mean vectors and covariance matrices of the corresponding Gaussian distributions. The variable $d$ is the dimensionality of the corresponding Gaussian distribution and $tr$ refers to the matrix trace operation. When this criterion is used, clusters are merged whenever $D_{KL}(p||q) + D_{KL}(q||p) \leq \lambda$ where $\lambda$ is a predefined threshold value.

### Temporal Pruning

The most common post-processing step is to use temporal information for pruning the clustering results. As discussed in Sec. 3.2, the time the user has stayed at a location often serves as a strong indicator of the meaningfulness of a location. The common way to use temporal pruning is to fix a threshold $t$ and remove all clusters where the total stay of the user is less than $t$. A common threshold value is to use $t = 10$ minutes. Alternatively, the pruning can utilize information about the number of times the user has visited a location. For example, the comMotion system, discussed in Sec. 5.2.1, requires that a cluster is visited at least three times before it considers it a meaningful place.

### Spatial Pruning

Spatial pruning refers to the use of spatial information to prune clustering results. The underlying idea behind spatial pruning is illustrated in Fig. 5.5. The figure shows two clusters that have been detected from a user's location measurements. The first cluster corresponds to the person's home, whereas the second cluster contains mainly commuting measurements. As

Figure 5.5: Example illustrating two clusters detected from data. The first cluster corresponds to the person's home, whereas the second covers a commuting trace.

the figure illustrates, the distribution of points around the home cluster is dense whereas the distribution around the commuting cluster is spread out. Accordingly, the example suggests that cluster density and variance could be used to prune clustering results.

Spatial pruning is utilized by density-based clustering (see Sec. 5.2.2) and by the Dirichlet process clustering algorithm (see Sec. 5.3). As density-based algorithms utilize information about neighborhood densities in the clustering process, the spatial pruning is integrated into the actual clustering phase. With Dirichlet process clustering, spatial pruning is performed separately after the clustering phase.

As the example in Fig. 5.5 illustrates, clusters that correspond to meaningful places tend to have smaller variances than the non-meaningful clusters. The spatial pruning used in our Dirichlet process clustering is based on this idea. In the pruning phase, we calculate the maximum variance of each cluster[2]. The resulting cluster variances are clustered into two classes and an upper threshold is used to ensure that the size of the resulting cluster remains small enough. We use the k-means algorithm for clustering the maximum variances. As k-means is known to be sensitive to initialization, we repeat the clustering 1000 times and use the best clustering result. As the upper threshold we use 300 meters, or the median of the maximum

[2]The maximum of the variances along the longitude and latitude axes.

variances, if it is higher than the threshold.

### 5.1.5 Labeling

While frequently visited locations can be detected from location data, they should be considered as places only if the user can associate semantics with the location, i.e., if they can be labeled (see Sec. 5.1). In most of the existing work, labeling is considered as the final step of the place identification process. For example, the comMotion system (see Sec. 5.2.1) prompts the user to label newly discovered clusters, whereas the Opportunity Knocks system asks the user to take a photo (see Sec. 5.2.4). Wang and Canny [113] studied different annotation techniques from a usability perspective and found the combination of online photo and offline editable text to be the best method for assigning labels.

An alternative is to allow users to label arbitrary locations and to learn a suitable representation for the locations from data. In this case the place representation is initially tied to some location information and later refined using place identification. The main advantage of this approach is that users can immediately provide semantics instead of needing to wait for the place identification algorithm to detect them. The main problem, on the other hand, is that the original location representations may be inaccurate and linking the initial representations with the results of place identification can be difficult. For example, Lehikoinen and Kaikkonen [72] associated semantics with the current GSM cell. A similar approach has been used by Li and Jonsson [73]. As the size of a GSM cell can be several kilometers (see Sec. 2.2), one cell can cover multiple places and it might be difficult to later associate a place with the appropriate label. As part of our current work we are constructing SerPens [17], a tool that links the semantic labels with whatever location related information is available. Accordingly, location labels can be linked, e.g., to GSM cell identifiers, GPS coordinates, fingerprints of the radio environment or combinations of multiple sources.

Also the possibility to automatically associate labels has been investigated in the literature. These approaches either use additional information, such as time of day and point-of-interest databases, to determine the type of building, or attempt to assign labels by comparing places across users. For example, Adams et al. [2] use time of day information to classify discovered clusters as either home, work or other. A cluster is assumed to represent home, if it is the clusters with the maximal duration before 7 in the morning or after 7 in the evening. Respectively, a cluster is labeled as work, if it has the maximal duration during working hours on weekdays. As another example, Liao et al. [74] use time of day information together

with information extracted from geographic databases to associate place labels. They consider the following categories: work, home, friend, parking and other. The results in [74] indicate over 90% overall accuracy and 100% accuracy at detecting the user's home and workplace.

## 5.2 Survey of Existing Algorithms

In this section we review existing place identification algorithms. As discussed in the previous section, we assume that the data is represented as (`latitude, longitude, duration`) tuples. We use the variable $i = 1, \ldots, n$ to index data points and the variable $j = 1, \ldots, k$ to index clusters (i.e., candidate places). The coordinates of a data point are represented using the variable $y_i$ and the variable $t_i$ is used to represent the duration measurement associated with point $i$. Note that the variable $y_i$ corresponds to the (`latitude, longitude`) pair associated with measurement $i$. The variable $c_i$ is used to indicate the place to which data point $i$ is assigned. The variable $n_j$ is used to denote the number of points assigned to a cluster and $\mu_j$ is used to denote the geographic mean of the coordinates of the points that are assigned to cluster $j$. Finally, boldfaced variable names are used to represent vectors, i.e., $\mathbf{y} = (y_1, \ldots, y_n)$ denotes the vector of all data points.

### 5.2.1 Radius-Based Algorithms

The idea in radius-based algorithms is to fix a radius and to merge all points that are within the radius into a cluster. Significant places are detected using temporal heuristics to prune either the data or the results of the clustering. Radius-based algorithms have been a popular choice for place identification because they are simple to implement and because they are relatively fast. The main disadvantage of radius-based algorithms is that they rely on multiple parameters whose values are determined using various heuristics. The effectiveness of these heuristics often depends on the properties of the underlying dataset and hence it can be difficult to generalize the algorithms to other datasets; see Sec. 6.3.

#### comMotion

The comMotion system was one of the first applications to utilize place identification. The place identification in comMotion is based on GPS signal loss [76]. The algorithm compares the location where GPS signal is lost with the next valid GPS measurement, i.e., the point where the user

re-appears. If these two measurements are within a pre-defined radius, the location is considered to be a building. If the user visits the building several times (three in the original comMotion system [99]), the algorithm considers the building as a place and prompts the user to label it. Only locations the user has labeled are considered meaningful. The main disadvantages of the comMotion algorithm are that it can only recognize places that are indoors and that are visited at least three times. As most of our datasets contain places that are outdoors and that are visited infrequently, we do not consider the comMotion algorithm as part of our evaluation in Chapter 6.

### Ashbrook and Starner

The algorithm of Ashbrook and Starner [6, 7] is a variation of the comMotion algorithm. In the original approach, Ashbrook and Starner log measurements only when the user moves at a speed of at least one mile per hour [6]. A preprocessing step is used to segment the measurements into a set of candidate places that consists of areas where we observe a continuous gap of at least 10 minutes. Finally, a clustering step is used to merge nearby places. The authors have later modified the algorithm so that speed information is no longer considered, but candidate places are created at locations where the GPS signal is lost [7]. The reliance on GPS signal loss makes the algorithm unable to detected outdoor places. Toyama et al. [108] have modified the algorithm by performing a clustering step that merges nearby points before creating candidate places. The radius parameter for this clustering step should be relatively small and depend on the accuracy of the underlying positioning technology. We have adopted a similar approach to Toyama et al. In our case the radius parameter for the first clustering step was set to 20 meters.

The modified algorithm of Ashbrook and Starner is shown in Alg. 1. The first step is to preprocess the data (line 2) by merging nearby location measurements and dropping measurements with a duration less than 10 minutes. After the preprocessing step, a radius-based clustering algorithm is used on the remaining data. The clustering iterates over the remaining data points (line 3), and, during each iteration, a random data point is selected and used to initialize a new cluster mean (lines 4 and 6). Next all data points within a predefined radius $\epsilon$ from the current mean are discovered (line 7) and the mean of these points is used as the new cluster mean (line 8). This process is repeated until the cluster mean stabilizes (line 9). The resulting cluster is added to the set of places and the points that belong to the cluster are removed from further consideration (lines 10 and 11). This process is continued until all points have been processed.

**Algorithm 1** C = AshbrookStarner($\mathbf{y}$, $\epsilon$)

1: C = $\emptyset$
2: T = `preprocess`($\mathbf{y}$)
3: **while** $T \neq \emptyset$ **do** ▷ Repeat until all points have been processed
4: Select random data point $z \in T$
5: **repeat**
6: $\mu = z$ ▷ Store the current cluster mean
7: $Q = \{w \mid w \in T,\ d(w, z) \leq \epsilon\}$
8: $z =$ `mean`($Q$) ▷ Calculate new cluster mean
9: **until** $d(z, \mu) <$ MIN_ERROR ▷ Loop until mean does not change
10: C = C $\bigcup\{Q\}$ ▷ Store cluster
11: $T = T \setminus Q$ ▷ Remove points from consideration
12: **end while**

The clustering step relies on a radius parameter $\epsilon$. Ashbrook and Starner determine the optimal value of $\epsilon$ automatically using the scree criterion, i.e., the clustering is repeated with different values of $\epsilon$ and the value corresponding to the knee-point of the resulting graph is considered the optimal choice. In the experiments in Chapter 6 we have used the algorithm described in Sec. 5.1.3 to automatically determine the knee value.

### Kang et al. Iterative Radius-Based Clustering

Kang et al. [60] have proposed an iterative radius-based algorithm for place identification. The basic idea is similar to the modified Ashbrook and Starner algorithm: nearby points are clustered into candidate places, a time threshold is used to determine which candidate places are meaningful, and a cluster merge step is used to combine multiple visits to the same place. The main difference to other algorithms is that Kang et al. use a temporary buffer to reduce the effects of minor fluctuations in location measurements. The temporary buffer ensures a new cluster is created only once enough successive measurements are outside the cluster.

In the experiments we use a batch version of the algorithm; see Alg. 2. The algorithm iteratively compares data points against the current cluster mean (line 8). If the point is within distance $\epsilon$ from the current cluster, the point is added to the cluster and the temporary buffer is cleared (lines 9 and 10). Otherwise the algorithm either adds the point to the temporary buffer (line 38) or, when the temporary buffer is full, creates a new cluster (lines 13 - 36). Before creating a new cluster, the algorithm checks whether the user stayed long enough within a cluster (line 14). If the stay was

**Algorithm 2** $(c_1, \ldots, c_n) = \text{ClusterKang}(\mathbf{y}, \epsilon, \psi, \gamma)$

```
1:  q = 1                                              ▷ Initialize cluster counter
2:  c_1 = q                                            ▷ Initialize first cluster
3:  T = ∅                                              ▷ Create temporary buffer
4:  i = 2
5:  while i ≤ n do
6:      Q = {w | c_w = q}                              ▷ Points in cluster q
7:      μ_q = mean(Q)                                  ▷ Calculate the mean of the cluster
8:      if d(y_i, μ_q) < ε then                        ▷ Compare point i with current cluster
9:          c_i = q                                    ▷ Add point to cluster
10:         T = ∅                                      ▷ Clear temporary buffer
11:         i = i + 1
12:     else
13:         t_T = Σ_{w:w∈T} t_w                        ▷ The duration of points in T
14:         if t_T > γ then                            ▷ If point outside and temporary buffer full
15:             t_q = Σ_{w:c_w=q} t_w                  ▷ The time the user stayed in the cluster
16:             if t_q > ψ then                        ▷ If user stayed long enough in the cluster
17:                 for j = 1, ..., q − 1 do
18:                     if d(μ_q, μ_j) < ε/3 then
19:                         ∀w s. t. c_w = j : c_w = q     ▷ Merge clusters
20:                     end if
21:                 end for
22:                 q = q + 1                          ▷ Start new cluster
23:             else                                   ▷ User did not stay long enough in the cluster
24:                 ∀w s. t. c_w = q : c_w = −1        ▷ Label as noise
25:             end if
26:             w = first(T)
27:             c_w = q
28:             T = T \ w
29:             for all w ∈ T do
30:                 z_w = 1                            ▷ Mark point as processed
31:                 if d(w, μ_q) < ε then
32:                     c_w = q
33:                     Z = {w ∈ T | z_w = 1}
34:                     T = T \ Z                      ▷ Remove all processed points
35:                 end if
36:             end for
37:         else                                       ▷ Temporary buffer not full
38:             T = T ∪ {y_i}                          ▷ Add point to temporary buffer
39:             i = i + 1
40:         end if
41:     end if
42: end while
```

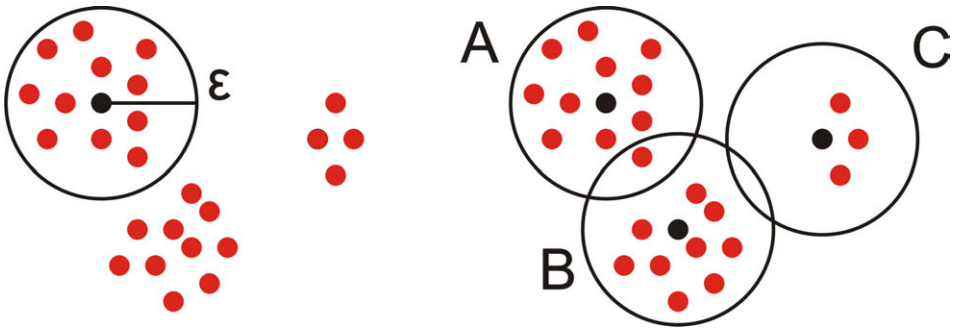



Figure 5.6: Illustrations of the concepts of $\epsilon$-neighborhood (left) and density-joinability (right).

long enough, the cluster is kept; otherwise the cluster is discarded and the points are labeled as noise (lines 16 - 25). If the cluster is kept, the algorithm checks whether the cluster can be merged with an existing cluster (lines 17 - 21). Two clusters are merged if the distance between the cluster means is no greater than $\epsilon/3$. After the merge step the algorithm initializes a new cluster using the first point in the temporary buffer (lines 26 and 27). The point is removed from the temporary buffer (line 28) and the remaining points are processed as if they were new location measurements. Accordingly, the points are added to the newly created cluster, if they are close enough to the cluster mean, and otherwise the points remain in the temporary buffer (lines 31 - 35). When a point from the temporary buffer is added to the cluster, points that precede it are removed from the temporary buffer (lines 33 and 34).

### 5.2.2 Density-Based Algorithms

Density-based algorithms utilize topological properties between points to cluster data. The idea is to detect areas within which points are closer together than outside of it, i.e., where the density of points is high. Before discussing the use of density-based clustering in place identification, we define some basic concepts related to density-based clustering.

The central notion in density-based clustering is the $\epsilon$-neighborhood of a point. Let $x$ denote an arbitrary point. The $\epsilon$-neighborhood of point $x$ is defined as $N_\epsilon(x) = \{z \,|\, d(x,z) < \epsilon\}$, i.e., as the set of points that are within distance $\epsilon$ from $x$. The actual shape of the neighborhood depends on the used distance function. The most common choice is the Euclidean distance, which leads to circular clusters. The notion of $\epsilon$-neighborhood is illustrated on the left-hand side of Fig. 5.6.

Density-based clustering requires that the neighborhood of a point is dense. This requirement is encoded into a second parameter, $MinPts$, which specifies how many points the $\epsilon$-neighborhood of a point must contain for it to be considered dense. If the neighborhood of a point is not dense, the point is labeled as noise.

The final notion we consider is density joinability. Points $p$ and $q$ are density joinable, if there exists a point $x$ such that $x \in N_\epsilon(p)$ and $x \in N_\epsilon(q)$, i.e., $x$ belongs to the $\epsilon$-neighborhood of both $p$ and $q$. If two points are density joinable, the clusters to which they belong to can be merged [121]. This is illustrated on the right-hand side of Fig. 5.6. Clusters $A$ and $B$ share a common point and hence they can be merged, whereas cluster $C$ cannot be merged with cluster $A$ or $B$.

Density-based algorithms can be seen as a generalization of radius-based algorithms. The $\epsilon$-parameter serves a similar function as the radius parameter and $MinPts$ inherently encodes the requirement that the user must have stayed long enough within an area. The main difference between the two classes of algorithms is in the way the places are represented. Radius-based algorithms typically use circles or ellipsoids to represent clusters whereas density-based clustering allows arbitrary shapes. This also influences the way clusters are merged: radius-based clustering merges clusters based on distance properties whereas density-based clustering relies on topological connectivity properties. As we show in Chapter 6, density-based algorithms are often superior to radius-based algorithms. In the following we describe two density-based algorithms, DBScan and DJCluster, which have been utilized for place identification. Also the spectral clustering algorithm described in Article III belongs to the class of density-based algorithms.

### DBScan

The Density-Based Spatial Clustering of Applications with Noise (DBScan) [39] is probably the best known density-based clustering algorithm. The DBScan algorithm is described in Alg. 3. The algorithm considers each point in turn (line 3). If the current point is unlabeled (line 4), the algorithm attempts to create a cluster around the point. The first step in the expansion is to calculate the $\epsilon$-neighborhood of the current point (line 5). If the neighborhood is not dense, the point is labeled as noise (lines 6 and 7). Otherwise a new cluster is created and the current point and all points in its neighborhood are added to the cluster (line 9). The algorithm attempts to recursively expand the new cluster by considering the points in the neighborhood of the current point as seed points (line 11). For each seed point, the algorithm forms the $\epsilon$-neighborhood (line 12). If the neigh-

**Algorithm 3** $(c_1, \ldots, c_n) = \text{DBScan}(\mathbf{y}, \epsilon, MinPts)$

```
1:  q = 1                                          ▷ Initialize cluster counter
2:  ∀i = 1, ..., n : c_i = 0                       ▷ Initialize cluster indicators
3:  for i = 1, ..., n do
4:      if c_i = 0 then
5:          T = N_ε(y_i).                          ▷ Form the ε-neighborhood of the point
6:          if #T < MinPts then
7:              c_i = −1                           ▷ Label the point as noise
8:          else
9:              ∀x ∈ T : c_x = q                   ▷ Create a cluster
10:             while T ≠ ∅ do
11:                 x = first(T)                   ▷ Select the first item in T
12:                 Q = N_ε(x)                     ▷ Form the ε-neighborhood of x
13:                 if #Q ≥ MinPts then
14:                     for all w ∈ Q do
15:                         if c_w = 0 then
16:                             T = T ⋃ {w}        ▷ Add unlabeled points to T
17:                         end if
18:                         c_w = q                ▷ Merge clusters
19:                     end for
20:                 end if
21:                 T = T \ {x}                    ▷ Remove x
22:             end while
23:             q = q + 1                          ▷ Update cluster counter
24:         end if
25:     end if
26: end for
```

borhood is dense, the points in the neighborhood are added to the current cluster and all previously unlabeled points are added to the seed set (lines 13 - 20). This process is continued as long as the cluster can be expanded.

The values of the parameters can be either fixed beforehand or selected according to some heuristic. Ester et al. [39] use a heuristic where the MinPts parameter is set to a predefined value $k$ and the distance from each point to its $k$th neighbor is calculated. The value of $\epsilon$ is then selected using the scree criterion (see Sec. 5.1.3). After experimenting with different values, Ester et al. use $k = 4$ in their experiments.

The basic DBScan algorithm suffers from some limitations. First, forming the $\epsilon$-neighborhood of a point can be slow. Unless suitable indexing schemes are used, the time complexity of a single $\epsilon$-neighborhood query is

$O(n)$. The situation can be improved using a spatial index. For example, using an $R^*$-tree [13] reduces the complexity to $O(\log n)$. In practice the complexity of neighborhood queries is an issue only if GPS data is sampled continually. A more severe limitation is related to the memory requirements of the algorithm. The recursive nature of the cluster expansion (lines 10 - 22 in Alg. 3) can require large amounts of memory and significantly slow down the algorithm [121]. Finally, the performance of the algorithm can be sensitive to the values of the parameters $\epsilon$ and $MinPts$ [119].

Adams et al. [1] have applied the DBScan algorithm for place identification. To overcome the limitations in cluster expansion, Adams et al. use velocity filtering to prune the data before clustering; see Sec. 5.1.2. The velocity threshold for filtering is set to the median of the user's velocity[3]. In more recent work, Adams et al. [2] use an incremental variant of the DBScan algorithm, proposed in [38]. The incremental DBScan limits the cluster expansion operations to a small subset of the data, which often leads to significant performance improvements. Adams et al. fixed the values of the parameters beforehand: $\epsilon$ was set to approximately 60 meters[4] and $MinPts$ was set to correspond to 5 minutes of data. After experimenting with different values, we used $\epsilon = 150$ meters in our experiments.

### DJCluster

Zhou et al. [120, 121] have developed a modified DBScan algorithm, Density-and-Join based clustering (DJCluster), for place identification; see Alg. 4. The main difference to DBScan relates to the operations that are performed for an individual point. If the $\epsilon$-neighborhood of a point contains enough points, instead of expanding the $\epsilon$-neighborhood, DJCluster compares the $\epsilon$-neighborhood with existing clusters (lines 9 - 15) and merges the neighborhood with all density joinable clusters (lines 10 - 12). This can significantly reduce the memory requirements of the clustering and hence the algorithm is better suited to mobile devices than the standard form of DBScan.

The original version of DJCluster [120] applied velocity pruning to reduce the amount of data that is processed. In their more recent work the filtering was not applied to the data [121]. For this reason we did not use

[3]Originally Adams et al. [1] used the mean velocity during a day as the velocity threshold and in their more recent work [2] two different velocity thresholds are used to prune out points so that the resulting data sets does not contain points that exceed slow walking speed. We use the median velocity as it gave slightly better results for our datasets.

[4]The value of $\epsilon$ was specified in degrees. When converted to meters, it corresponds to approximately 60 meters.

**Algorithm 4** $(c_1, \ldots, c_n) = \text{DJCluster}(\mathbf{y}, \epsilon, MinPts)$

```
q = 1                                          ▷ Initialize cluster counter
∀i = 1, ..., n : c_i = 0                       ▷ Initialize cluster indicators
for i = 1, ..., n do
    if c_i = 0 then
        Q = N_ε(y_i)                           ▷ Form the ε-neighborhood of a point
        if #Q < MinPts then
            c_i = −1                           ▷ Label as noise
        else                                   ▷ Check if we can merge clusters
            for j = 1, ..., q − 1 do
                if Q density joinable with cluster j then
                    ∀w s.t. c_w = j : c_w = q  ▷ Merge clusters
                end if
                ∀x ∈ Q : c_x = q               ▷ Create a cluster
                q = q + 1
            end for
        end if
    end if
end for
```

velocity pruning with DJCluster in the experiments. After experimenting with various values, we set $\epsilon = 50$ meters and determine the value of $MinPts$ using the scree criterion (see Sec. 5.1.3).

### 5.2.3 Probabilistic Clustering

Probabilistic clustering assumes the measurements are a sample from a set of random variables. Each random variable $j$ corresponds to a cluster which can be represented by some (parametric) density function $f_j(\theta_j)$, where $\theta_j$ denotes the parameters of the density function. The parameters of the density functions are assumed unknown and clustering is considered as an inference problem where the goal is to estimate the parameters $\theta_j$ from data [15, 18]. A common choice is to assume that the density functions are Gaussian, in which case the parameters $\theta_j$ correspond to the mean vector and covariance matrix of the underlying Gaussian. In this section we describe the agglomerative Gaussian clustering proposed by Aipperspach et al. [3]. Also our Dirichlet process clustering, described in Sec. 5.3, and Article IV belongs to the class of probabilistic clustering algorithms.

**Algorithm 5** $(c_1, \ldots, c_n)$ = AgglomerativeGMM($\mathbf{y}$, $\lambda_1$, $\tau$, $\lambda_2$)

1: $\forall i = 1, \ldots, n : d_i = (y_i, \sum_{q=1,\ldots,i} t_q)$ ▷ Initialize data
2: $\forall i = 1, \ldots, n : c_i = i$ ▷ Assign each point to its own cluster
3: **for** $i = 1, \ldots, n$ **do**
4: **for all** $k$ **do** ▷ Merge adjacent clusters
5: $T_i = \{d_w : c_w = i\}$, $T_k = \{d_w : c_w = k\}$
6: $\mu_i = \texttt{mean}(\texttt{T}_\texttt{i})$, $\mu_k = \texttt{mean}(\texttt{T}_\texttt{k})$
7: $\Sigma_i = \texttt{covariance}(\texttt{T}_\texttt{i})$, $\Sigma_k = \texttt{covariance}(\texttt{T}_\texttt{k})$
8: $KL = \texttt{JointKLDivergence}(\mu_\texttt{i}, \mu_\texttt{k}, \Sigma_\texttt{i}, \Sigma_\texttt{k})$
9: **if** $KL < \lambda_1$ **then**
10: $\forall w$ s.t. $c_w = k : c_w = i$
11: **end if**
12: **end for**
13: **if** $\sum_{w:c_w=i} t_w < \tau$ **then** ▷ Check if duration of stay long enough
14: $\forall w$ s.t. $c_w = i : c_w = 0$
15: **else**
16: **for all** $k$ **do** ▷ Merge spatially adjacent visits
17: $T_i = \{(y_w) : c_w = i\}$, $T_k = \{(y_w) : c_w = k\}$
18: $\mu_i = \texttt{mean}(\texttt{T}_\texttt{i})$, $\mu_k = \texttt{mean}(\texttt{T}_\texttt{k})$
19: $\Sigma_i = \texttt{covariance}(\texttt{T}_\texttt{i})$, $\Sigma_k = \texttt{covariance}(\texttt{T}_\texttt{k})$
20: $KL = \texttt{JointKLDivergence}(\mu_\texttt{i}, \mu_\texttt{k}, \Sigma_\texttt{i}, \Sigma_\texttt{k})$
21: **if** $KL < \lambda_2$ **then**
22: $\forall w$ s.t. $c_w = k : c_w = i$
23: **end if**
24: **end for**
25: **end if**
26: **end for**

### Agglomerative Gaussian clustering

Aipperspach et al. [3] have developed an agglomerative probabilistic clustering algorithm, which can be interpreted essentially as a probabilistic variant of radius-based algorithms. Originally the algorithm was developed for detecting places from high precision location measurements within a home. The algorithm represents candidate places using three-dimensional Gaussian distributions. Two of the dimensions correspond to position and one to time. The algorithm is described in Alg. 5.

Initially the algorithm assigns each point to its own cluster (line 2). Next the algorithm merges clusters that are spatially and temporally adjacent (lines 4 - 12). As the merge criterion the algorithm considers a

threshold on the joint Kullback-Leibler divergence between the Gaussian distributions of the clusters (Eq. 5.2 in Sec. 5.1.4). Similarly to radius-based algorithms, the next step is to drop clusters that do not explain a sufficiently long duration of the data (line 13). The remaining clusters correspond to continuous stays within a sufficiently small area. The final step in the algorithm merges clusters that correspond to visits to the same place (lines 16 - 24). Also the second merge operation is based on the joint Kullback-Leibler divergence. However, whereas the first merge step considers both position and time information, the second merge step considers only position information.

The algorithm relies on three parameters, two merge thresholds $\lambda_1$ and $\lambda_2$, and one duration threshold $\tau$. The data in our setting differs from the original use of the algorithm and thus the same parameter values cannot be used directly. Our data is collected periodically and the distances between measurements in our data are much larger than within a home. After experimenting with different values, we set the duration threshold $\tau$ to 10 minutes. As the merge thresholds we used $\lambda_1 = 10$ and $\lambda_2 = 10$.

### 5.2.4 Grid Based Clustering

Grid-based algorithms are closely related to radius-based clustering approaches. Though, instead of comparing successive measurements, grid-based algorithms distribute measurements to discrete patches and compare adjacent patches. When a street map is available, the patches usually correspond to continuous street segments. Otherwise rectangular grid cells are used. The place identification phase then measures the time the user has spent at each patch and, if the time exceeds a predefined threshold, the patch is considered a candidate place. Finally, spatial clustering is used to merge recurring visits and places in adjacent patches.

Opportunity Knocks is a transportation assistance system that has been targeted at people with cognitive disabilities [91]. Opportunity Knocks uses a hierarchical dynamic Bayesian network (see, e.g., [82]) for learning and reasoning about the user's transportation routines, to predict likely destinations and to recognize anomalous behavior. The underlying model considers the world as a graph where edges represent road sections and nodes correspond to intersections [75, 90]. Opportunity Knocks also integrates a grid-based clustering approach, which follows the generic pattern described above, i.e., a time threshold is used to determine significant nodes and clustering is used to merge multiple visits and nearby nodes [75, 91].

Liao et al. [74] use conditional random fields (see, e.g., [68]) for simultaneous place and activity detection. The first step of the algorithm is

to associate location measurements with patches. This is done using a conditional random field that considers consistency and smoothness of the mapping. Consistency ensures that successive measurements are mapped to patches that are next to each other, whereas smoothness ensures users do not frequently switch between different streets. The next step in the algorithm is to associate activities with patches. Two types of activities are considered: navigation activities (commuting, walking, driving a car) and other activities. The activity detection relies on time and speed information and on information extracted from geographic databases. Patches where the user performs other activities than navigation are marked as places and, finally, spatial clustering is used to merge multiple visits and nearby patches.

In our work, we have experimented with grid-based algorithms within the Context Watcher application [62] and Article III. In practice the grid-based algorithms suffer from the same problems as radius-based algorithms, i.e., the performance is sensitive to the time thresholds and the algorithms easily make mistakes, e.g., at traffic lights and at tram stops; see Sec. 6.2. Though, the results of Liao et al. [74] indicate that considering also activity information can be used to overcome some of these weaknesses.

### 5.2.5 Radio Beacon Algorithms

Instead of operating on coordinates, radio beacon algorithms use information about the radio environment to identify meaningful places. In the simplest case the beacons correspond to the identifier of the current GSM base station. For example, Laasonen et al. [66] recursively merge GSM cells into clusters and detect places by considering the number of visits and the time the user stays in a cluster. The algorithm of Laasonen et al. has also been integrated into the ContextPhone platform; see Sec. 4.1.1. Meneses and Moriera [80] use the frequency of GSM transitions to filter the measurements. The underlying intuition is that, when the user is mobile, cell transitions occur at a more frequent rate than when the user is stationary. Each stationary period, i.e., period between successive mobility detections, is represented as a fingerprint. The fingerprints contain the cell identifiers that were seen during the stationary period and the relative percentage of time that the phone was connected to each cell. Finally, a clustering step is used to merge stationary periods with similar fingerprints. The results indicate that the use of cell identifier fingerprints and corresponding transition patterns enables detecting places that are smaller than the diameter of the corresponding GSM cells [80].

In Article III, we describe a heuristic graph clustering algorithm that

operates on GSM cell identifiers that are augmented with GPS coordinates. The algorithm calculates the mean of the location measurements collected from a single GSM cell and it merges multiple cells together if the corresponding means are within a predefined radius. Due to the large size of GSM cells, the clusters that the algorithm recognizes are often large. Moreover, the size of the clusters tends to grow over time as more cells are merged together. Article III also describes an algorithm that uses GSM cell transition patterns to merge GSM cells into clusters. More specifically, let $A$ and $B$ denote two GSM cells, the algorithm compares the transition probabilities $P(A, B)$ and $P(B, A)$ against a predefined threshold and merges the two cells into a cluster if both probabilities exceed the threshold. This algorithm suffers from the same problems as the other GSM identifier based approaches, i.e., that the size of the clusters can be relative large due to large GSM cell size. In addition, the algorithm is sensitive to the threshold value.

The BeaconPrint [52] algorithm continually gathers information about GSM and WiFi beacons and uses this information to identify places. The basic idea is to fix a time window and to assume the user is in a place if the scan remains sufficiently stable for the duration of the time window. Stability is controlled by a certainty parameter that ensures the fraction $f$ of scans with the same set of visible beacons is large enough over the time window. Each place is represented as a fingerprint that contains a so-called response rate histogram (see [69]) of beacons that were visible in at least $f$ percent of the scans over the time window. Finally, fingerprints that share at least 68% (one standard deviation of a Normal distribution) of beacon identifiers are merged together. Once the system has been deployed, the same heuristic is used to recognize when the user is visiting the place.

## 5.3 Dirichlet Process Clustering

The Dirichlet Process Clustering (DPCluster), described in Article IV, is a probabilistic place identification algorithm that is based on Dirichlet process mixture models [4, 42, 84, 96], which are a special case of finite mixture models [78]. The algoritm can automatically learn the correct number of places from data and it offers good generalization performance against spatio-temporal variations. In the following we first discuss the underlying statistical model, after which we discuss how the model can be applied in practice. We end the section with a discussion about performance issues.

| Symbol | Description |
|---|---|
| $y_i$ | Coordinates of an individual data point |
| $\mathbf{y}$ | The vector $(y_1, \ldots, y_n)$ of data points |
| $c_i$ | Cluster indicator for data point $i$ |
| $t_j$ | The total time the user has stayed at cluster $j$ |
| $\mathbf{c}$ | Vector $(c_1, \ldots, c_k)$ of cluster indicators |
| $k$ | The number of active clusters |
| $n$ | Number of data points |
| $\overline{y}$ | Sample mean |
| $\Sigma$ | Sample precision |
| $\mu_j$ | The mean vector of cluster $j$ |
| $S_j$ | The precision matrix of cluster $j$ |
| $\lambda$ | Mean vector for the prior on cluster means $\mu_j$ |
| $R$ | Precision matrix for the prior on cluster means $\mu_j$ |
| $\beta$ | Degrees of freedom for the prior on cluster precision matrices $S_j$ |
| $W$ | Inverse scaling matrix for the prior on cluster precision matrices $S_j$ |
| $\alpha$ | Concentration parameter of the Dirichlet process prior |

Table 5.1: A summary of the notation that is used.

### 5.3.1 Statistical Model

The DPCluster algorithm is based on the statistical model shown in Fig. 5.7. The notation we use is summarized in Table 5.1. In the model, each cluster $j$ is modeled as a multivariate Normal distribution with unknown mean $\mu_j$ and precision[5] matrix $S_j$. The distribution of a single data point $y_i$ is thus given by

$$y_i | c_i = j, \mu_j, S_j \sim \mathcal{N}\left(\mu_j, S_j^{-1}\right). \tag{5.3}$$

To infer the mean vectors and precision matrices from data, we assign prior distributions for them. We use conjugate priors since they provide a good balance between computational simplicity and clustering performance. The conjugate prior for the multivariate Normal distribution is to assign a Normal distribution on the mean vector and a Wishart distribution

[5]Inverse covariance

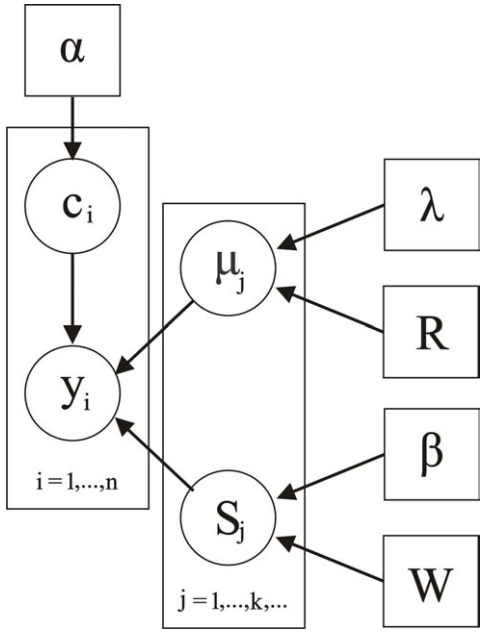



Figure 5.7: The statistical model underlying the Dirichlet process place identification algorithm.

on the precision matrix; see, e.g., [45]. Accordingly, we have

$$\mu_j \sim \mathcal{N}\left(\lambda, R^{-1}\right) \tag{5.4}$$

$$S_j \sim \mathcal{W}i\left(2\beta, \frac{1}{2}W^{-1}\right), \tag{5.5}$$

where $\lambda, \beta, R$ and $W$ are hyperparameters. Following Rasmussen [96], we use a hierarchical model and assign priors to all hyperparameters. If we would want clusters that are on average of specific size, we could fix the values of $\beta$ and $W$ beforehand. This can be done, for example, by assigning $W$ to be the sample covariance and setting $\beta$ so that the product $\beta W^{-1}$ corresponds to the desired cluster size[6].

Let $\overline{y}$ denote the sample mean and $\Sigma$ the sample precision. We assign $\lambda$ a Normal distribution whose mean equals the sample mean and whose covariance matrix corresponds to the sample covariance, i.e.,

$$\lambda \sim \mathcal{N}\left(\overline{y}, \Sigma^{-1}\right). \tag{5.6}$$

[6]The product $\beta W^{-1}$ corresponds to the expectation of the Wishart distribution on the cluster precision matrices.

The distribution of $\lambda$ has full support over the set of data points. This implies that samples from the prior on cluster means $\mu_j$ also have full support over the data points. The prior also implies that values of $\mu_j$ that are near the sample mean are most likely. A potential problem with this prior is that it puts more weight on the center (sample mean) of the data points, but this does not necessarily correspond to a place. For example, in large cities people often commute for a long period of time to get to work. In this case, the sample mean corresponds to the midpoint of the travel route and samples from the prior on cluster means rarely fall near the actual clusters (home or work). Thus, the algorithm can take longer time to convergence. An alternative is to assign $\lambda$ a uniform distribution over the set of data points.

The matrix $R$ specifies the precision matrix for the cluster means. Intuitively, we would want the expectation of the distribution on $R$ to correspond to the sample precision $\Sigma$ as in this case the values for $\mu_j$ are on average drawn from a distribution that is specified by the sufficient statistics of the data. However, we also have to ensure that the resulting Wishart distribution is well defined[7]. These goals can be achieved by assigning the following distribution on $R$:

$$R \sim \mathcal{W}i\left(2, \frac{1}{2}\Sigma\right). \tag{5.7}$$

The hyperparameters for the prior on precision matrices are more complicated. We start from the variable $\beta$, which defines the degrees of freedom for the Wishart distribution on $S_j$. We do not want to limit the size of clusters beforehand and hence we need to assign a vague prior on $\beta$. Again we need to ensure that the Wishart distribution over $S_j$ remains well defined. These two goals can be achieved by assigning $\beta$ a flat, continuous distribution over the interval $[1, \infty)$. In order to achieve this, we consider the variable $(\beta - 1)^{-1}$ and assign a Gamma prior for it:

$$(\beta - 1)^{-1} \sim \mathcal{G}\left(\frac{1}{2}, 2\right). \tag{5.8}$$

Samples for $\beta - 1$ follow a flat inverse-Gamma distribution and they are within the interval $(0, \infty)$. Thus the distribution of $\beta$ is as desired.

For the hyperparameter $W$, i.e., the inverse scale matrix of the prior on $S_j$, we assign the following Wishart prior:

$$W \sim \mathcal{W}i\left(2, \frac{1}{2}\Sigma^{-1}\right). \tag{5.9}$$

[7] A Wishart distribution $\mathcal{W}i(b, W)$ is well defined whenever the $p \times p$ matrix $W$ is positive definite and $b \geq p$ holds for the degrees of freedom parameter $b$.

The expectation of $W$ equals the sample covariance and, since the expectation of $S_j$ equals $\beta W^{-1}$, samples from $S_j$ are on average scaled variants of the sample precision matrix.

The cluster indicators $c_i$ are assigned a Dirichlet process prior. Following Neal [84], the prior distribution of $c_i$ can be written in the following form:

$$\begin{aligned} c_i = j|\mathbf{c_{-i}} &\sim \frac{n_{-i,j}}{n-1+\alpha} \\ c_i \neq q|\mathbf{c_{-i}} &\sim \frac{\alpha}{n-1+\alpha} \quad (\forall q \in \{1,\ldots,k\}) \,. \end{aligned} \tag{5.10}$$

Here $n_{-i,j}$ denotes the number of data points that belong to cluster $j$ when the data point $i$ is ignored. The variable $\alpha$ is the concentration parameter of the Dirichlet process prior that, together with the priors on $\mu_j$ and $S_j$, governs the rate at which new clusters are created and $\mathbf{c_{-i}}$ is a vector that contains all other cluster indicators except $c_i$, i.e., $\mathbf{c}_{-i} = (c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n)$. The support of the prior on $c_i$ is the countably infinite set $\{1, 2, \ldots, k, \ldots\}$ where $k$ denotes the number of clusters that have currently data points associated with them. For each of the represented clusters $j \in \{1, \ldots, k\}$, the prior assigns a probability mass of $n_{-i,j}/(n-1+\alpha)$. A probability mass of $\alpha/(n-1+\alpha)$ is assigned for all of the unrepresented clusters combined. Thus, although the number of clusters is potentially infinite, only some of them are represented at a given time and we do not need to make a distinction between the clusters that are unrepresented.

To finalize our model specification, we need to assign a prior on the concentration parameter $\alpha$. We assign a vague inverse-Gamma prior for this purpose so that

$$\alpha^{-1} \sim \mathcal{G}\left(\frac{1}{2}, 2\right). \tag{5.11}$$

This prior results in a flat distribution that has support over $(0, \infty)$.

### 5.3.2 The Dirichlet Process Algorithm

The actual place identification consists of two steps. In the first phase we estimate the parameters of the statistical model from data (i.e., perform clustering) and in the second phase we prune the clusters. A common way to estimate the parameters is to use Markov Chain Monte Carlo (MCMC) techniques [45, 47]. In our case we use Gibbs sampling (see, e.g., [47, 84]), which is an MCMC technique that sequentially updates each parameter

in turn. The updates are sampled from probability distributions that are conditioned on the values of other parameters. Thus, when sampling a new parameter value, the values of other parameters are assumed fixed. The sampling process and the required conditional distributions are described in more detail in Article IV.

As the first step of the post-processing phase we re-estimate the cluster indicators. While the parameter estimation phase provides estimates also for the cluster indicators, some assignments necessarily have a small probability due to the model's tendency to favour a small number of compact clusters. We take advantage of this property to identify non-meaningful areas and to label some measurements as noise. To re-estimate the cluster indicators, for each data point $y_i$ we find the cluster $q$ which has the largest likelihood of generating the point, i.e.,

$$q = \arg \max_{j=1,\ldots,k} p(y_i|\theta_j, S_j^{-1}). \tag{5.12}$$

If this probability exceeds 0.01, we assign the point to cluster $q$ and otherwise we label the point as noise. Typically the likelihood values are either extremely small (order of magnitude of $10^{-32}$) or extremely large, which means the re-estimation phase is not sensitive to the used threshold value. We also tested this empirically by re-creating the evaluation results with different parameter values. Larger threshold values slightly improve the results, and thus the threshold value we use in the experiments is conservative. Once the algorithm is deployed within a real world system, the same procedure can be used to recognize visits to a place.

After the cluster indicators have been re-estimated, we apply temporal and spatial pruning to the results. The temporal pruning drops all clusters where the user stayed less than five minutes, whereas the spatial pruning uses an upper threshold on the maximum cluster variance of a cluster. The threshold for the cluster variance is determined automatically using the constrained clustering procedure described in the spatial pruning part of Sec. 5.1.4.

### 5.3.3 Performance

The performance of the Dirichlet process algorithm depends, among other things, on the number of points and on the spatial distribution of data. When the data is relatively evenly distributed, the cluster indicators mix properly and the algorithm converges rapidly. However, when the data is spread out, i.e., it has long and narrow commuting traces, the mixing is much slower. The mixing is also slow when there is a large number of

dense areas within a relatively small area. In general, for a given number of clusters, the cluster parameters converge in a few hundred (100 - 500) iterations, but the cluster indicators require 1000 to 10000 iterations to converge. With our current implementation, running 100 iteration for 1000 data points takes around 5 minutes on a desktop computer. The development of more efficient inference algorithms for Dirichlet process models is an active research area and many improvements have been recently suggested in the literature [29, 57].

# Chapter 6

# Comparison of Algorithms

Existing evaluations of place identification algorithms tend to focus on datasets that have been collected only by few participants [52] or that exhibit little spatial and temporal variation [7, 58, 121]. Moreover, these evaluations seldom compare more than two or three algorithms. This chapter presents an evaluation of the DPCluster algorithm using twelve datasets that have been collected at different locations and over different periods of time. The dataset contain a wide variety of different activities including tourism, business visits as well as everyday activities of people living in the location. We also compare the DPCluster algorithm against five other place identification algorithms. The algorithm of Ashbrook and Starner and the iterative radius-based clustering algorithm of Kang et al. were selected as representative examples of radius-based algorithms; see Sec. 5.2.1. Density-based algorithms are represented by the DBScan and the DJCluster algorithms; see Sec. 5.2.2. Finally, we consider the agglomerative Gaussian clustering as an example of probabilistic algorithms; see Sec. 5.2.3. As grid-based algorithms (see Sec. 5.2.4) are practically identical to radius-based algorithms when no additional information is considered, and as radio-beacon algorithms (see Sec. 5.2.5) require additional data, these classes of algorithms were excluded from the comparison.

## 6.1 Experiment Setup

### 6.1.1 Description of Data Sets

The data that was used in the evaluation was collected using BeTelGeuse (see Sec. 4.2) and an external Bluetooth GPS receiver. Five out of the twelve datasets were collected in Helsinki, Finland by different individuals and over different periods of time. For these datasets, the GPS receiver

| Dataset | Measurements | Duration | Rate | Participant |
|---|---|---|---|---|
| Buenos Aires | 378 | 12h | 60s | A |
| Canberra | 1500 | 2d 8h 50min | 60s | A |
| Calcutta | 2042 | 2d 16h 55min | 60s | B |
| Gran Canaria | 465 | 8h 30min | 60s | C |
| Innsbruck | 350 | 23h 15min | 60s | A |
| Petra | 743 | 15h 55min | 60s | A |
| Tokyo | 1279 | 4d 1h | 60s | A |
| Helsinki 1 | 676 | 1d 1h 20min | 20s | D |
| Helsinki 2 | 914 | 3d 7h 10min | 20s | E |
| Helsinki 3 | 1241 | 6d 20h | 20s | F |
| Helsinki 4 | 10317 | 14d 19h 55min | 20s | G |
| Helsinki 5 | 16591 | 12d 3h 40min | 20s | H |
| **TOTAL:** | **36496** | **≈ 49d 20h** | | |

Table 6.1: Summary of the datasets that were used in the evaluation.

was sampled at 20 second intervals. The remaining seven datasets were collected from different locations and over different periods of time. The sampling rate for these datasets was 60 seconds. The location measurements were processed before analysis using the the operations described in Sec. 5.1.1. Table 6.1 presents summary statistics for the datasets. Note that the summaries have been calculated after the data preparation phase and the original number of measurements is significantly higher. In the following we give a brief characterization of each dataset:

- **Buenos Aires:** Tourism-oriented traces collected from three short visits to the city of Buenos Aires, Argentina during February 2009. The places contain, e.g., hotels, the ferry and bus stations and main tourist sights in Buenos Aires.
- **Canberra:** Dataset containing everyday and tourism-oriented traces collected intermittently over a period of three months from December 2007 to February 2008. The places in the dataset contain, e.g., home, workplace and the main museums in Canberra.
- **Calcutta:** Dataset containing everyday activities collected during two weeks in December 2008. The places in the dataset contain, e.g., home, two restaurants and homes of several relatives and friends.
- **Gran Canaria:** Tourism-oriented dataset collected during a single day in February 2008. The places correspond to the main sights on

the island.

- **Innsbruck:** Dataset containing a mixture of tourism and business-related location measurements collected during Ubicomp 2007 in Innsbruck, Austria. The places contain, e.g., a hotel, the location of Ubicomp banquet and tourist sights within the city. As no data was collected during the conference sessions, the conference venue is not part of the places.

- **Petra:** Tourism-oriented traces collected from the archaeological site of Petra, Jordan, in October 2007. The places contain, e.g., a hotel and the main sights within the Petra area.

- **Tokyo:** Dataset collected in May 2009 during a business trip to Tokyo, Japan. The places in the dataset contain the hotel where the person was staying and different universities that the person visited.

- **Helsinki 1:** Dataset containing traces collected by a visitor to our university during February 2009. The places contain, e.g., the flat where the person was staying, the university and restaurants.

- **Helsinki 2-5:** Datasets containing everyday activities collected over a three month period from July 2008 to September 2008. The places in these datasets contain, e.g., homes, workplaces, shops, homes of friends and relatives, libraries and kindergartens. Each Helsinki dataset was collected by a different individual.

### 6.1.2 Evaluation Procedure

In the evaluation phase we applied the place identification algorithms on the datasets and compared the resulting clusters against information about the actual locations of places. We represent places as individual points and clusters as ellipsoids. As most place identification algorithms merely estimate the cluster indicators, i.e., which points correspond to a place, but not the cluster parameters, we needed to estimate the ellipsoid parameters from the points that are assigned to each cluster. The error ellipses that are estimated from the data also cover all the points. To this end, the ellipsoids of the DPCluster algorithm are centered at the estimated cluster means, whereas the ellipsoids of the other algorithms are centered at the geographic mean of the data points assigned to the cluster. The principal axes of the ellipsoids are determined from cluster covariances so that the resulting ellipsoids correspond to 95% error ellipses; see, e.g., [103]. With

the DPCluster algorithm we use the estimated cluster covariances to calculate the error ellipses, and we use the sample covariance of the points assigned to a cluster with the other algorithms.

To obtain information about the actual locations of places, we visualized the datasets using Google Earth and asked the person who had collected the data to mark the places in the data. We also showed the resulting clusters to the participant who was allowed to augment the place annotations by creating spurious places that correspond to (i) clusters that form around a meaningful place and could be merged with the place; (ii) places that the participant did not initially mark but that she recognizes (e.g., places that are visited infrequently or for a short period of time). Finally, we compare the clustering results against the place labels and classify the detected clusters into the following categories:

- **Correct (C)**: The cluster corresponds to an actual place.
- **Spurious (S)**: The cluster corresponds to a place that the participant has labeled as spurious.
- **Failed (F)**: The cluster is a mistake and does not correspond to any location where the participant has intentionally spent time.
- **Missing (M)**: Places that were not detected even though the participant expected the algorithm to be able to detect them from data. A place is considered missing if it was not detected and it is not spurious.

In the comparison phase we match a cluster with a place if it satisfies one of the following criteria: (i) the mean of the cluster is within 150 meters of the place; (ii) the closest cluster point is within 150 meters of the place and the cluster mean is within 300 meters of the place; (iii) the place is inside the cluster and within 300 meters of the cluster mean. As we represent places as simple dots, the heuristics are used to ensure that small discrepancies between place locations and clusters do not influence the results. In reality places seldom correspond to simple dots, but tend to correspond to more complex geometries [123]. The heuristics also ensure that large clusters are not considered places unless they are actually centered near the actual place.

### 6.1.3 Metrics

We evaluate the algorithms based on accuracy and completeness of the place identification. We use precision, i.e., the proportion of correctly identified

| | C | S | F | M | Precision | Recall | F1-Score |
|---|---|---|---|---|---|---|---|
| **DPCluster** | 57 | 22 | 28 | 27 | 0.74 | 0.68 | **0.71** |
| **DJCluster** | 51 | 24 | 19 | 33 | **0.80** | 0.61 | 0.69 |
| **Kang et al.**[a] | 64 | 40 | 80 | 20 | 0.57 | 0.76 | 0.65 |
| **A & S**[b] | 62 | 31 | 68 | 22 | 0.58 | 0.74 | 0.65 |
| **DBScan** | 67 | 40 | 92 | 17 | 0.54 | **0.80** | 0.64 |
| **AGC**[c] | 53 | 20 | 44 | 31 | 0.62 | 0.63 | 0.63 |

[a]Iterative radius-based clustering of Kang et al.
[b]Algorithm of Ashbrook and Starner
[c]Agglomerative Gaussian clustering

Table 6.2: The results of the study sorted by F1-score.

places, as a measure of clustering accuracy, and we use recall, i.e., the fraction of places that the algorithm is able to identify, as a measure of clustering completeness [98]. The precision and recall values are typically closely related so that an increase in precision often causes a decrease in recall and vice versa. For this reason we consider the $F1$-score, a combined measure (harmonic mean) of precision and recall, as our main evaluation criterion. The precision, recall and F1-score were calculated as follows:

$$\text{Precision} = \frac{\text{Correct} + \text{Spurious}}{\text{Correct} + \text{Failed} + \text{Spurious}} \tag{6.1}$$

$$\text{Recall} = \frac{\text{Correct}}{\text{Correct} + \text{Missing}} \tag{6.2}$$

$$\text{F1-score} = \frac{2(\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}} \tag{6.3}$$

The spurious clusters are by definition correctly identified places. For this reason we consider the spurious clusters while calculating precision values. However, since the participant does not expect the algorithms to detect spurious clusters, we do not consider them while calculating recall values.

## 6.2 Results

Table 6.2 summarizes the results of the evaluation. The DPCluster algorithm has the highest F1-score and it is the best performing algorithm on five of the twelve datasets. In total, DPCluster is also among the top three algorithms for nine of the datasets. The second best performing algorithm, DJCluster, has the highest precision but it suffers from poor recall.

DJCluster is best on three datasets and it is among the three best algorithms for a total of seven datasets. The remaining algorithms, i.e., the iterative radius-based clustering, the algorithm of Ashbrook and Starner, the DBScan algorithm and the agglomerative Gaussian clustering perform practically equally.

The results indicate that most algorithms have a significant trade-off between precision and recall. For example, DJCluster has the highest precision but second worst recall. Another example is the DBScan algorithm, which has the highest recall but poorest precision. Moreover, DBScan identifies more failed clusters than any other algorithm. Another observation from the results is that the optimal parameter values seem to depend on the dataset. Temporal thresholds are more resistant to variations in the datasets than the spatial thresholds. The best performing algorithms, DPCluster and DJCluster, learn the parameter values during the clustering phase, whereas most of the other algorithms rely on fixed parameter thresholds, or they would require manual tuning of parameters for each dataset. As our goal was to evaluate the generalization performance of the algorithms, we used fixed parameter values for all datasets.

We also examined whether algorithms that rely on fixed parameter values would perform better on certain types of datasets (e.g., everyday life or tourism oriented), but this was not the case. For example, the DBScan algorithm was the best performing algorithm on the Helsinki 5 dataset, but it had the worst performance on the Helsinki 4 dataset. Both of these datasets have been collected in the same spatial area and from everyday life situations.

We also separately evaluated the influence of spurious clusters. Considering spurious clusters as mistakes, or ignoring them while calculating precision values, slightly influences the precision values, but the differences between the algorithms remain practically the same.

## 6.3 Discussion

The evaluation also indicated various shortcomings in existing place identification algorithms. This section discusses some of these shortcomings and suggests possible ways to improve the performance of place identification algorithms.

### 6.3.1 Commuting Stops, Traffic Lights, Traffic Jams etc.

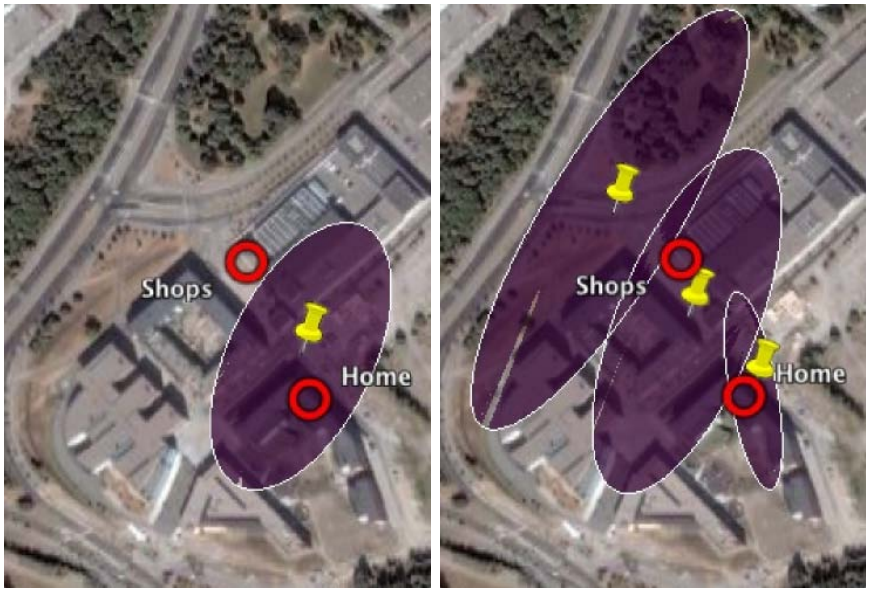
Most place identification algorithms rely exclusively on temporal criteria to determine whether detected clusters are meaningful or not. The temporal


(a) (b)

Figure 6.1: Many place identification algorithms erroneously recognize non-meaningful stops such as commuting stops or traffic lights as places. The pins represent cluster means, the ellipses correspond to error ellipses and the toruses represent places.

criteria can be encoded as a minimum threshold on the number of visits to a place or as a minimum threshold on the time the participant has stayed at a location. Relying only on temporal information can cause the algorithm to detect non-meaningful clusters that correspond, e.g., to tram stops, traffic jams or traffic lights. For example, Fig. 6.1(a) illustrates how the algorithm of Ashbrook and Starner detects non-meaningful places along a tram route from the Helsinki 2 dataset. The two non-meaningful places correspond to traffic lights that are near a tram stop. Velocity pruning can further magnify this problem as it removes information about the density of points around a cluster. This is illustrated in Fig. 6.1(b), which shows how the DBScan algorithm detects practically all tram stops along the route as well as a traffic light.

The DPCluster algorithm is relatively robust against this effect as most of the non-meaningful stops are pruned out in the spatial pruning phase. During the clustering phase the algorithm creates a new cluster for each




(a) (b)

Figure 6.2: Incorrect parameter values may lead to situations where places that are nearby are merged together or where the algorithm creates a large number of non-meaningful clusters around the actual places. The pins represent cluster means, the ellipses correspond to error ellipses and the toruses represent places.

non-meaningful stop. However, since the density around the intermediate points is not high enough for creating another new cluster, these points will be assigned to one of the clusters corresponding to a stop. This increases the cluster variance and makes it possible to prune out the cluster in the post-processing phase. Thus our results suggest that utilizing information about the spatial density of points around the place can help detecting and removing non-meaningful clusters. Note that this is in contrast with density-based clustering, which looks at the density of points within a cluster, not around it.

### 6.3.2 Place Granularity

When multiple places are near each other, place identification algorithms may fail to distinguish the individual places. This problem is illustrated in Fig. 6.2(a), which shows how the agglomerative Gaussian clustering algorithm merges home and shops into the same place in the Helsinki 2 dataset.

Similar effects can be observed from the results of the heuristic graph clustering algorithm in Article III. Varying place granularity influences especially place identification algorithms that are based on sequential clustering (i.e., radius-based clustering and the agglomerative Gaussian clustering) as they merge multiple visits to the same place over time. Problems with merging clusters are further illustrated in Fig. 6.2(b), which shows the corresponding results for the iterative radius-based clustering of Kang et al.; see Sec. 5.2.1. This algorithm uses a small merge threshold which causes the algorithm to fail to merge the appropriate clusters. This problem could be alleviated by adapting the clustering thresholds based on the local density of points and considering information about visiting patterns to different places in the merge step.

The granularity of places can cause problems also for density-based algorithms. For example, both the DBScan and DJCluster algorithms created only one cluster which was centered around the shop area. Only two algorithms, the algorithm of Ashbrook and Starner and the DPCluster, were able to correctly identify the two different places in the example. Generally speaking the granularity of places does not influence the accuracy of the DPCluster algorithm, but it can slow down the mixing of cluster indicators, which means the Gibbs sampler would require more iterations to converge.

### 6.3.3 Altitude variations

Practically all place identification algorithms are based on (Euclidean) distances that are calculated from two-dimensional position information. These distances are accurate only when there are no altitude variations between measurements. Ignoring altitude variations can thus skew the distance calculations and result in inaccuracies during the clustering phase. This problem is illustrated in Fig. 6.3, which shows how the DPCluster algorithm creates a large cluster around two places (one spurious, the lake, and one actual place, the banquet) in the Innsbruck dataset. In the example, altitude variations cause distance measurements to be underestimated and, as a consequence, the algorithm is unable to split the points into two places. The underestimation can also be evidenced from the differences in the cluster variances along the latitude and longitude axes.

All place identification algorithms are vulnerable to altitude variations, though density-based algorithms and the DPCluster algorithm are more vulnerable than the other approaches as they rely on information about the density of points. As the figure illustrates, altitude variations can also skew variance estimates in the DPCluster algorithm.

Altitude information is typically ignored because existing location sys-

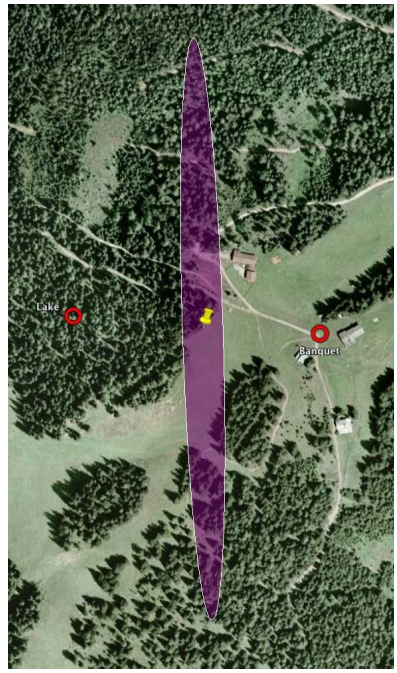



Figure 6.3: Altitude variations can skew distance calculations and cause overly large cluster variances. The pin represents the mean of a cluster, the ellipse represents the error ellipse and the toruses represent places.

tems do not support high quality altitude measurements. Fingerprinting systems are unable to estimate the client's altitude and GPS altitude estimates tend to be of lower quality than the latitude and longitude measurements [70]. GPS errors tend to be systematic across a specific period of time, which suggests that relative altitude information, i.e., differences between successive measurements, could be used in place identification. This is one of the issues we plan to tackle as part of our future work.

## 6.4 Summary of Place Identification Algorithms

Table 6.3 summarizes the place identification algorithms and their strengths as well as weaknesses; see Table 6.2 for the abbreviations that are used in the table.

| Algorithm | Strengths | Weaknesses |
|---|---|---|
| DPCluster | Good precision and recall<br>Detects places at different granularities<br>Robust against non-meaningful stops | Gibbs sampler can be slow<br>Vulnerable to altitude variations<br>Poor at detecting infrequently visited indoor places |
| DJCluster | Good precision<br>Detects compact clusters | Poor recall<br>Slow on large datasets<br>Poor at detecting infrequently visited indoor places |
| A & S | Relatively robust against non-meaningful stops<br>Good recall | Suffers from place granularity<br>Poor precision |
| Kang et al. | Fast performance<br>Able to detect infrequent places | Poor accuracy<br>Highly sensitive to parameter values<br>Vulnerable to non-meaningful stops |
| DBScan | Detects places at different granularities<br>Fast performance | Vulnerable to non-meaningful stops<br>Vulnerable to altitude variations |
| AGC | No precision-recall tradeoff<br>Principled way to handle uncertainty | Sensitive to parameter values<br>Non-intuitive parameters<br>Slow on large datasets |

Table 6.3: Summary of the place identification algorithms considered in the study.

# Chapter 7

# Conclusions

In this thesis we have investigated the process of moving from coordinate information to information that corresponds to places that are meaningful to the user. Information about meaningful places can be used, e.g., to provide awareness cues in applications that support social interactions, to provide personalized and location-sensitive information to the user, and to support mobile user studies by providing cues about the situations the user has been in. Enabling the use of place information for these purposes requires both system level support and algorithmic solutions. On a system level, there is a need for platforms that facilitate the interactions between location systems, place identification algorithms and applications, whereas on the algorithmic level there is a need for techniques that are accurately, and without offline tuning, able to identify place information from data collected by the user. The contributions of this thesis address these needs by providing an open source mobile platform that provides the desired system level support, and by providing a novel place identification technique that does not require tuning for different datasets and that performs better than existing approaches.

The research towards this thesis has also opened up new questions that have not been addressed in this thesis, many of which we are currently addressing as part of our ongoing activities. On a system level, the performance evaluation of BeTelGeuse in Article II indicated that high battery consumption of Internet connectivity is currently a major obstacle for location-aware applications and services. To this end, we are currently designing intelligent data uploading policies that aim to reduce the need for Internet connectivity. More specifically, we are focusing on policies that determine when to send location updates to a remote server in a way that balances battery consumption and freshness of location information.

In terms of place identification algorithms, the evaluation of different

approaches in Chapter 6 identified limitations with all approaches. For example, the best algorithms were ones that search the parameter space for optimal values and, as a consequence, are slower than algorithms that fix the parameters beforehand. A potential direction for future investigations is to examine locally adaptive algorithms that search for the optimal parameter values within a small neighborhood of points. Other areas for potential improvements include development of algorithms that efficiently discard, e.g., non-meaningful traffic stops, and that consider altitude information in the place identification process.

Existing work, including this thesis, has mainly focused on place identification from GPS measurements. Although the algorithms discussed in this thesis can be used on any coordinate data, a major limitation of all of the algorithms is that they do not consider errors in location measurements. When GPS measurements are not available, GSM positioning can be used to estimate the user's location. However, since GSM positioning errors are typically larger than GPS measurement errors, place identification algorithms that operate on a combination of GPS and GSM data should consider the differing errors in the location estimates. In order to extend the DPCluster algorithm, presented in this thesis and Article IV, to take into account uncertainty in location estimates, error models that characterize the measurement error of the underlying technology are needed. While GPS errors are relatively well understood, no generic models that characterize GSM positioning errors are currently available.

In the thesis we focused on offline evaluation of place identification algorithms. While offline evaluations provide insights into the algorithms' capabilities of detecting places after they have been visited, they fail to provide insights into how well the algorithms can support applications and services. In real world systems, detecting when the user revisits a place is equally important as discovering the places from data. Moreover, the recognition should adhere to the users' mental models about places, e.g., recognize that the user is at home only when she actually is inside the home, not near it. As discussed in Sec. 5.3.2, recognizing visits to detected places is an inherent part of the DPCluster algorithm, however, not all algorithms have inherent capabilities for recognizing visits to places. Evaluating and comparing the recognition capabilities of different algorithms remains an important piece of future work.

Deploying place identification algorithms into real world systems poses challenges to system design and data management. First of all, place identification algorithms can operate directly on the mobile device or data can be sent to a server for analysis. As places are personal and as location in-

formation can be potentially sensitive, the former approach is better from a privacy perspective. The latter approach, on the other hand, provides better scalability as place identification algorithms do not need to be deployed on the mobile handsets. The latter approach also provides better support for social interactions as a server-based approach makes it easier to share place information across users. The second deployment aspect we consider is whether the place identification algorithms operate in a batch mode or sequentially as the data arrives. As the evaluation in Chapter 6 indicated, algorithms that operate in a batch mode have better accuracy and are less vulnerable, e.g., to non-meaningful stops. On the other hand, batch algorithms suffer from latency as applications and services can only access place information after the suitable batch of data has been collected and analyzed[1]. In our current work we are investigating the possibility to use user-provided semantics to provide rough estimates of places and to use batch algorithms to refine the place estimates afterwards. The final deployment issue we consider is how to maintain place information when the system is used for a long period of time. The places that are relevant to a user can change over time and the system should be able, not only to learn new places, but also to forget places that are no longer relevant to a user. Understanding and modeling the relevance of places over time is an important and interesting direction for future research.

As discussed in Chapter 3, the meanings people attribute to places are not necessarily personal, but they can relate to social situations. However, current algorithmic solutions are typically unable to detect these places as the users are not necessarily stationary or because the physical locations that are linked with the social situations change (e.g., different coffee shops). Places that are related to meaningful social interactions provide a novel, largely unexplored perspective to the topics addressed in this thesis. For example, how can these situations be detected; how can sensor data measuring social interactions be combined with location data; how can people use this information; what interaction possibilities does this open; how can place information be shared across users; what privacy implications does this have for the users?

[1]Storage space requirements are negligible for contemporary devices. Collecting data for a single day requires approximately 45 kilobytes of storage space when the GPS receiver is sampled once every minute. When the receiver is sampled every 10 seconds, approximately 270 kilobytes of storage space is required.

# References


[1] B. Adams, D. Phung, and S. Venkatesh. Extraction of social context and application to personal multimedia exploration. In *Proceedings of the ACM Conference on Multimedia (MM)*, pages 987 – 996. ACM, 2006.

[2] B. Adams, D. Phung, and S. Venkatesh. Sensing and using social context. *ACM Transactions on Multimedia Computing*, 5(2):1 – 27, 2008.

[3] R. Aipperspach, T. Rattenbury, A. Woodruff, and J. Canny. A quantitative method for revealing and comparing places in the home. In *Proceedings of the 8th International Conference on Ubiquitous Computing (UbiComp)*, volume 4206 of *LNCS*, pages 1–18. Springer, 2006.

[4] C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

[5] I. Arminen. Social functions of location in mobile telephony. *Personal and Ubiquitous Computing*, 10(5):319–323, 2006.

[6] D. Ashbrook and T. Starner. Learning significant locations and predicting user movement with GPS. In *Proceedings of the 6th International Symposium on Wearable Computers (ISWC)*, pages 101– 108. IEEE, 2002.

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

[8] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proceedings of the 19th Conference on Computer Communications (INFOCOM)*, volume 2, pages 775–784. IEEE Computer Society, 2000.

[9] D. Bannach, K. Kunze, P. Lukowicz, and O. Amft. Distributed modular toolbox for multi-modal context recognition. In *Proceedings of the 19th International Conference on Architecture of Computing Systems (ARCS'06)*, volume 3894 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2006.

[10] D. Bannach, P. Lukowicz, and O. Amft. Rapid prototyping of activity recognition applications. *IEEE Pervasive Computing*, 7(2):22 – 31, 2008.

[11] J. Baus, K. Cheverst, and C. Kray. A survey of map-based mobile guides. In L. Meng, A. Zipf, and T. Reichenbacher, editors, *Map-based Mobile Services Theories, Methods and Implementations*, pages 197–216. Springer-Verlag, 2005.

[12] J. Baus, A. Krüger, and W. Wahlster. A resource-adaptive mobile navigation system. In *Proceedings of the 7th international conference on Intelligent user interfaces (IUI)*, pages 15–22. ACM, 2002.

[13] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 322 – 331. ACM, 1990.

[14] S. Benford, R. Anastasi, M. Flintham, A. Drozd, A. Crabtree, C. Greenhalg, N. Tandavanitj, M. Adams, and J. Row-Farr. Coping with uncertainty in a location-based game. *IEEE Pervasive Computing*, 2(3):34–41, 2003.

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

[16] P. A. Bernstein. Middleware: a model for distributed system services. *Communications of the ACM*, 39(2):86–98, 1996.

[17] S. Bhattacharya, P. Nurmi, J. Kukkonen, and P. Floréen. Serpens - a tool for sematically enriched location information on personal devices. In *Proceedings of the 3rd International Conference on Body Area Networks*, 2008.

[18] H. H. Bock. Probabilistic models in cluster analysis. *Computational Statistics & Data Analysis*, 23(1):5–28, 1996.

[19] M. S. Braasch and A. J. van Dierendonck. GPS receiver architectures and measurements. *Proceedings of the IEEE*, 87(1):48–64, 1999.

[20] R. B. Cattell. The scree test for number of factors. *Multivariate Behavioral Research*, 1(2):245–276, 1966.

[21] M. Y. Chen, T. Sohn, D. Chmelev, D. Hähnel, J. Hightower, J. Hughes, A. LaMarca, F. Potter, I. E. Smith, and A. Varshavsky. Practical metropolitan-scale positioning for GSM phones. In *Proceedings of the 8th International Conference on Ubiquitous Computing (UbiComp)*, volume 4206 of *Lecture Notes in Computer Science*, pages 225–242. Springer, 2006.

[22] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, P. P. Klasnja, K. Koscher, A. LaMarca, J. A. Landay, L. LeGrand, J. Lester, A. Rahimi, A. Rea, and D. Wyatt. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 7(2):32–41, 2008.

[23] K. Church and B. Smyth. Understanding the intent behind mobile information needs. In *Proceedings of the 13th international conference on Intelligent user interfaces (IUI)*, pages 247–256. ACM, 2009.

[24] S. Consolvo, I. E. Smith, T. Matthews, A. LaMarca, J. Tabert, and P. Powledge. Location disclosure to social relations: Why, when & what people want to share. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI'05)*, pages 81–90. ACM, 2005.

[25] S. Consolvo and M. Walker. Using the experience sampling method to evaluate ubicomp applications. *IEEE Pervasive Computing*, 2(2):24–31, April-June 2003.

[26] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley & Sons, New York, 1991.

[27] T. Cresswell. *Place: A Short Introduction*. Blackwell Publishing, 2004.

[28] M. Csikszentmihalyi and R. Larson. Validity and reliability of the experience-sampling method. *Journal of Nervous and Mental Diseases*, 175(9):526–536, 1987.

[29] H. Daumé III. Fast search for Dirichlet process mixture models. In M. Meila and X. Shen, editors, *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 83–90, 2007.

[30] A. Dix, J. Finlay, G. D. Abowd, and R. Beale. *Human-Computer Interaction*. Prentice Hall, 3rd edition, 2004.

[31] T. H. Dixon. An introduction to the global positioning system and some geological applications. *Reviews of Geophysics*, 29(2):249 – 276, 1991.

[32] P. Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004.

[33] P. Dourish and G. Button. On technomethodology: foundational relationships between ethnomethodology and system design. *Human-Computer Interaction*, 13:395 – 432, 1998.

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

[35] P. Enge and P. Misra. Special issue on global positioning system. *Proceedings of the IEEE*, 87(1):3 – 15, 1999.

[36] P. K. Enge. The global positioning system: Signals, measurements, and performance. *International Journal of Wireless Information Networks*, 1(2):83 – 105, 1994.

[37] F. Espinoza, P. Persson, A. Sandin, H. Nyström, E. Caccitore, and M. Bylund. GeoNotes : Social and navigational aspects of location-based information systems. In *Proceedings of the 3rd International Conference on Ubiquitous Computing (UbiComp)*, volume 2201 of *Lecture Notes in Computer Science*, pages 2–17. Springer, 2001.

[38] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB)*, pages 323–333. Morgan Kaufmann Publishers Inc., 1998.

[39] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226 – 231. AAAI, 1996.

[40] A. Famili, W.-M. Shen, R. Weber, and E. Simoudis. Data preprocessing and intelligent data analysis. *Intelligent Data Analysis*, 1(1-4):3 – 23, 1997.

[41] L. Feldman Barrett and D. J. Barrett. An introduction to computerized experience sampling in psychology. *Social Science Computer Review*, 19(2):175 – 185, 2001.

[42] T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

[43] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay. MyExperience: A system for *in situ* tracing and capturing of user feedback on mobile phones. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys)*, pages 57 – 70. ACM, 2007.

[44] J. Froehlich and J. Krumm. Route prediction from trip observations. In *Proceedings of the SAE World Congress and Exhibition*, 2008.

[45] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall / CRC, 2004.

[46] I. A. Getting. Perspective/navigation - the global positioning system. *IEEE Spectrum*, 30(12):43 – 47, 1993.

[47] W. Gilks, D. Spiegelhalter, and S. Richardson. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC, 1996.

[48] M. C. González, C. A. Hidalgo, and A.-L. Barabási. Understanding individual human mobility patterns. *Nature*, 453:779–782, 2008.

[49] P. Gustafson. Meanings of place: Everyday experience and theoretical conceptualizations. *Journal of Environmental Psychology*, 21:5–16, 2001.

[50] S. Harrison and P. Dourish. Re-place-ing space: the roles of place and space in collaborative systems. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work (CSCW)*, pages 67–76. ACM, 1996.

[51] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[52] J. Hightower, S. Consolvo, A. LaMarca, I. Smith, and J. Hughes. Learning and recognizing the places we go. In *Proceedings of the 7th International Conference on Ubiquitous Computing (UBICOMP)*, volume 3660 of *Lecture Notes in Computer Science*, pages 159–176. Springer-Verlag, 2005.

[53] J. Hightower, A. LaMarca, and I. E. Smith. Practical lessons from Place Lab. *IEEE Pervasive Computing*, 5(3):32–39, 2006.

[54] G. H. T. Hofte. What's that Hot Things in my Pocket? SocioXensor, a smartphone data collector. In *Proceedings of the 3rd International Conference on e-Social Science*, 2007.

[55] S. S. Intille, J. Rondoni, C. Kukla, I. Ancona, and L. Bao. A context-aware experience sampling tool. In *CHI'03 extended abstracts on Human Factors in Computing Systems*, pages 972–973. ACM, 2003.

[56] S. S. Intille, E. M. Tapia, J. Rondoni, J. Beaudin, C. Kukla, S. Agarwal, L. Bao, and K. Larson. Tools for studying behavior and technology in natural settings. In *Proceedings of the 5th International Conference on Ubiquitous Computing (Ubicomp)*, pages 157 – 174, 2003.

[57] S. Jain and R. Neal. Splitting and merging components of a nonconjugate Dirichlet process mixture model. *Bayesian Analysis*, 2(3):445 – 472, 2007.

[58] Q. Jones, S. A. Grandhi, S. Karam, S. Whittaker, C. Zhou, and L. Terveen. Geographic 'place' and 'community information' preferences. *Computer Supported Cooperative Work*, 17(2-3):137–167, 2008.

[59] E. Kaasinen. User needs for location-aware mobile services. *Personal and Ubiquitous Computing*, 7(1):70–79, 2003.

[60] J. Kang, W. Welbourne, B. Stewart, and G. Borriello. Extracting places from traces of locations. In *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots (WMASH)*, pages 110 – 118. ACM Press, 2004.

[61] M. B. Kjærgaard and C. V. Munk. Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength. In *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 110–116, 2008.

[62] J. Koolwaaij, A. Tarlano, M. Luther, P. Nurmi, B. Mrohs, A. Battestini, and R. Vaidya. ContextWatcher - sharing context information in everyday life. In *Proceedings of the IASTED conference on Web Technologies, Applications and Services*. IASTED, 2006.

[63] B. Krämer. Classifications of generic places: Explorations with implications for evaluation. *Journal of Environmental Psychology*, 15:3–22, 1995.

[64] A. Krüger, J. Baus, D. Heckmann, M. Kruppa, and R. Wasinger. Adaptive mobile guides. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321, pages 521 – 549. Springer, 2007.

[65] J. Krumm, G. Cermak, and E. Horvitz. RightSPOT: A novel sense of location for a smart personal object. In *Proceedings of the 5th International Conference on Ubiquitous Computing (Ubicomp)*, pages 36–43. Springer, 2003.

[66] K. Laasonen, M. Raento, and H. Toivonen. Adaptive on-device location recognition. In *Proceedings of the Second International Conference on Pervasive Computing (PERVASIVE)*, LNCS 3001, pages 287–304. Springer, 2004.

[67] K. V. Laerhoven and H.-W. Gellersen. Spine versus porcupine: A study in distributed wearable activity recognition. In *Proceedings of the 8th International Symposium on Wearable Computers (ISWC)*, pages 142–149. IEEE Computer Society, 2004.

[68] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[69] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. E. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. N. Schilit. Place Lab: Device positioning using radio beacons in the wild. In *Proceedings of 3rd International Conference on Pervasive Computing (PERVASIVE)*, volume 3468, pages 116–133. Springer, 2005.

[70] R. B. Langley. Dilution of precision. *GPS World*, 10(5):52 – 59, 1999.

[71] E. Laurier. Why people say where they are during mobile phone calls. *Environment and Planning D: Society and Space*, 19:485–504, 2001.

[72] J. T. Lehikoinen and A. Kaikkonen. PePe field study: constructing meanings for locations in the context of mobile presence. In *Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'06)*, pages 53–60. ACM, 2006.

[73] W. Li and M. Jonsson. Providing user self-contained location information using mobile phones. In *Adjunct Proceedings of Pervasive*, 2006.

[74] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from GPS traces using hierarchical conditional random fields. *International Journal of Robotics Research*, 26(1):119–134, 2007.

[75] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171:311–331, 2007.

[76] N. Marmasse and C. Schmandt. A user-centered location model. *Personal and Ubiquitous Computing*, 6(5 - 6):318 – 321, 2002.

[77] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE Computer Society, 2006.

[78] G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, 2000.

[79] J. G. McNeff. The global positioning system. *IEEE Transactions on Microwave Theory and Techniques*, 50:645 – 652, 2002.

[80] F. Meneses and A. Moreira. Using GSM CellID positioning for place discovering. In *Proceedings of the 1st Workshop on Location Based Services for Health Care (Locare)*, pages 34–42, 2006.

[81] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys)*, pages 337–350, New York, NY, USA, 2008. ACM.

[82] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[83] R. Natau. Efficient, rotation-independent activity recognition on mobile phones. Master's thesis, ETH Zürich, 2009.

[84] R. Neal. Markov chain methods for Dirichlet process mixture models. Technical Report 9815, University of Toronto, Department of Statistics, 1998.

[85] V. Otsason. Accurate indoor localization using GSM fingerprinting. Master's thesis, University of Tartu, 2005.

[86] V. Otsason, A. Varshavsky, A. LaMarca, and E. de Lara. Accurate GSM indoor localization. In *Proceedings of the 7th International Conference on Ubiquitous Computing (UbiComp)*, volume 3660 of *Lecture Notes in Computer Science*, pages 141–158. Springer, 2005.

[87] A. Oulasvirta, R. Petit, M. Raento, and S. Tiitta. Interpreting and acting on mobile awareness cues. *Human-Computer Interaction*, 22(1&2):97–135, 2007.

[88] A. Oulasvirta, M. Raento, and S. Tiitta. ContextContacts: redesigning SmartPhone's contact book to support mobile awareness and collaboration. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services (MobileHCI)*, pages 167–174. ACM, 2005.

[89] L. Palen and M. Salzman. Voice-mail diary studies for naturalistic data capture under mobile conditions. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work (CSCW)*, pages 87 – 95. ACM, 2002.

[90] D. Patterson, L. Liao, D. Fox, and H. Kautz. Inferring high-level behavior from low-level sensors. In *Proceedings of the 5th International Conference on Ubiquitous Computing (UBICOMP)*, 2003.

[91] D. J. Patterson, L. Liao, K. Gajos, M. Collier, N. Livic, K. Olson, S. Wang, D. Fox, and H. A. Kautz. Opportunity knocks: A system to provide cognitive assistance with transportation services. In *Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp)*, volume 3205 of *Lecture Notes in Computer Science*, pages 433–450. Springer-Verlag, 2004.

[92] M. Raento, A. Oulasvirta, and N. Eagle. Smartphones: an emerging tool for social scientists. *Sociological Methods and Research*, 37(2):426–454, 2009.

[93] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. ContextPhone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4(2):51 – 59, 2005.

[94] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *Bulleting of the Technical Committee on Data Engineering*, 23(4):3 – 13, 2000.

[95] B. Rao and L. Minakakis. Evolution of mobile location-based services. *Communications of the ACM*, 46(12):61 – 65, December 2003.

[96] C. E. Rasmussen. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12, pages 554 – 560. MIT Press, 2000.

[97] E. Relph. *Place and Placelessness*. Pion Books, London, 1976.

[98] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, 1968.

[99] C. Schmandt, N. Marmasse, S. Marti, N. Sawhney, and S. Wheeler. Everywhere messaging. *IBM Systems Journal*, 39:660 – 677, 2000.

[100] M. I. Silventoinen and T. Rantalainen. Mobile station emergency locating in GSM. In *Proceedings of the IEEE International Conference on Personal Wireless Communications*, pages 232–238. IEEE, 1996.

[101] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *ACM SIGIR Forum*, 33(1):6–12, 1999.

[102] R. Simon and P. Fröhlich. GeoPointing: Evaluating the performance of an orientation aware location based service under real-world conditions. *Journal of Location Based Services*, 2(1):24–40, 2008.

[103] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

[104] T. Sohn, W. G. Griswold, J. Scott, A. LaMarca, Y. Chawathe, I. Smith, and M. Chen. Experiences with Place Lab: an open source

toolkit for location-aware computing. In *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, pages 462–471. ACM, 2006.

[105] T. Sohn, K. A. Li, W. G. Griswold, and J. D. Hollan. A diary study of mobile information needs. In *Proceedings of the 26th SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*. ACM, 2008.

[106] S. Tamminen, A. Oulasvirta, K. Toiskallio, and A. Kankainen. Understanding mobile contexts. *Personal and Ubiquitous Computing*, 8(2):135–143, 2004.

[107] G. H. H. ter Hofte. Xensible interruptions from your mobile phone. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services (MobileHCI)*, pages 178–181. ACM, 2007.

[108] N. Toyama, T. Ota, F. Kato, Y. Toyota, T. Hattori, and T. Hagino. Exploiting multiple radii to learn significant locations. In *Proceedings of the 1st International Workshop on Location- and Context-Awareness (LoCA)*, volume 3479 of *Lecture Notes in Computer Science*, pages 157 – 168, Berlin Heidelberg, 2005. Springer-Verlag.

[109] E. Trevisani and A. Vitaletti. Cell-ID location technique, limits and benefits: an experimental study. In *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 51–60. IEEE Computer Society, 2004.

[110] Y.-F. Tuan. Humanistic geography. *Annals of the Association of American Geographers*, 66(2):266–276, 1976.

[111] Y.-F. Tuan. *Space and Place: The Perspective of Experience*. University of Minnesota Press, 2001.

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

[113] J. Wang and J. Canny. End-user place annotation on mobile devices: A comparative study. In *Extended Abstracts of the Conference on Human factors in computing systems (CHI)*, pages 1493 – 1498, 2006.

[114] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys'09)*, pages 179–192, 2009.

[115] R. Want. The active badge location system. *ACM Transactions on Information Systems*, pages 91 – 102, 1992.

[116] A. H. Weilenmann and P. Leuchovius. "I'm waiting where we met last time": Exploring everyday positioning practices to inform design. In *Proceedings of the 3rd Nordic Conference on Human-Computer Interaction (NordiCHI)*, pages 33–42. ACM, 2004.

[117] T. Winograd. Architectures for context. *Human-Computer Interaction*, 16:401–419, 2001.

[118] T. Yamabe, A. Takagi, and T. Nakajima. Citron: A context information acquisition framework for personal devices. In *Proceedings of the 11th IEEE International Conference on Embedded and Re*, pages 489– 495. IEEE, 2005.

[119] O. R. Zaïane, A. Fos, C.-H. Lee, and W. Wang. On data clustering analysis: Scalability, constraints, and validation. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 28–39. Springer, 2002.

[120] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen. Discovering personal gazetteers: an interactive clustering approach. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems (GIS)*, pages 266 – 273, New York, NY, 2004. ACM Press.

[121] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen. Discovering personally meaningful places: An interactive clustering approach. *ACM Transactions on Information Systems*, 25(3):12, 2007.

[122] C. Zhou, P. Ludford, D. Frankowski, and L. Terveen. Talking about place: An experiment in how people describe places. In *Adjunct Proceedings of the Third International Conference on Pervasive Computing (PERVASIVE)*, 2005.

[123] C. Zhou, P. J. Ludford, D. Frankowski, and L. G. Terveen. How do people's concepts of place relate to physical locations? In *Proceedings of the International Conference on Human-Computer Interaction*

*(INTERACT)*, volume 3585 of *Lecture Notes in Computer Science*, pages 886–898, 2005.