Model-Driven Optimizations for Public Sensing Systems

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der Naturwissenschaften
(Dr. rer. nat.) genehmigte Abhandlung

vorgelegt von

Damian Cassius Philipp

aus Bergisch-Gladbach

Hauptberichter:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Mitberichter:	Prof. Dr. rer. nat. Jörg Hähner
Tag der mündlichen Prüfung:	28. Oktober 2015

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart
2015

Acknowledgements

First and foremost, I would like to thank Prof. Dr. Kurt Rothermel for giving me the opportunity to perform my research as a member of his group. As my advisor, he provided feedback and guidance that have been invaluable in successfully performing the work documented in this thesis. My thanks also go to Prof. Dr. Jörg Hähner, who kindly agreed to act as my co-advisor.

Next, I would like to thank my current and former colleagues in the Distributed Systems group. Inspiring discussions, constructive feedback and countless matches at the soccer table made the Distributed Systems group a friendly and productive working environment. Special thanks go to Dr. Frank Dürr, as this work would not have been possible without both his technical and personal input. Furthermore, special thanks go to my collaborators in various projects, namely Daniel Fischer, Dr. Harald Weinschrott, Gerald Georg Koch, Patrick Baier, Christoph Dibak, Naresh Nayak, Dr. Susanne Becker, and Michael Peter.

During my research, I had the pleasure of working with a number of exceptionally bright students that contributed to this work through their thesis or as student research assistants. I would like to thank them for their effort and contributions that have greatly advanced—and partially inspired—this work.

I would also like to express my gratitude towards the German Research Foundation and the Baden-Württemberg Stiftung GmbH for funding my research under the "ComNSense" and "SpoVNet" projects, respectively.

Last but not least, I thank my friends and family who shared with me the joy of good times and supported me through hard times. Special thanks go to my parents Margot and Johannes for their continuous support that helped me to complete this work.

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

The proliferation of modern smartphones such as the Apple iPhone or Google Android Phones has given rise to *Public Sensing*, a new paradigm for sensor data acquisition using spare resources of commodity smartphones. As smartphones are already deployed wherever there are people present, data collection is enabled at an urban scale. Having access to such a wealth of data facilitates the creation of applications depending on real-world information in a way that may have a lasting impact on our everyday life.

However, creating large-scale Public Sensing systems is not without its challenges. On the data requesting side, an interface is required that allows to specify arbitrary sensing tasks independent of the mobility of participants and thus facilitates user acceptance of Public Sensing. On the data gathering side, as many people as possible must participate in the system and thus provide a sufficient amount of data. To this end, the system must conserve the resources shared by participants as much as possible, with the main concern being energy. Participants will withdraw from the system, when participating significantly impacts the battery life of their smartphone. We address the aforementioned issues in the context of two applications: Indoor map generation and large-scale environmental data acquisition.

In the area of indoor map generation, we first address the problem of building an indoor map directly from odometry traces. In contrast to existing approaches, our focus is to extract the maximum amount of information from trace data without relying on additional features such as WiFi fingerprints. Furthermore, we present an approach to improve indoor maps derived from traces using a formal grammar encoding structural information about similar indoor areas. Using this grammar allows us to extend an incomplete trace-based map to a plausible layout for the entire floor while simultaneously improving the accuracy of floor plan objects observed by odometry traces. Our evaluations show that the accuracy of

grammar-based maps in the worst-case is similar to the accuracy of trace-based maps in the best-case, thus proving the benefit of the grammar-based approach. To improve the energy efficiency of the mapping process, we furthermore present a generic quality model for trace-based indoor maps. This quality metric is used by a scheduling algorithm, instructing a participating device to disable its energy-intensive sensors while it travels in an area that has been mapped with high quality already, enabling energy savings of up to 15 %.

In the area of large-scale environmental data acquisition, we first present the concept of virtual sensors as a mobility-independent abstraction layer. Applications configure virtual sensors to report a set of readings at a given sampling rate at a fixed position. The Public Sensing system then selects smartphones near the position of a virtual sensor to provide the actual data readings. Furthermore, we present several optimization approaches geared towards improving the energy efficiency of Public Sensing. In a local optimization, smartphones near each individual virtual sensor coordinate to determine which device should take a reading and thus avoid oversampling the virtual sensor. The local optimization can achieve a 99 % increase in efficiency with the most efficient approaches and exhibits only about 10 % decrease in result quality under worst conditions. Furthermore, we present a global optimization, where a data-driven model is used to identify the subset of most interesting virtual sensors. Data is obtained from this subset only, while readings for other virtual sensors are inferred from the model. To this end, we present a set of online learning and control algorithms that can create a model in just hours or even minutes and that continuously validate its accuracy. Evaluations show that the global optimization can save up to 80 % of energy while providing inferred temperature readings matching an error-bound of 1° C up to 100 % of the time.

Zusammenfassung

Die wachsende Verbreitung moderner Smartphones wie des Apple iPhone oder den Android-Geräten von Google hat die Idee des *Public Sensing* inspiriert. *Public Sensing* ist ein neues Paradigma der Datenerfassung unter Verwendung ungenutzter Ressourcen von Smartphones. Da Smartphones überall dort verfügbar sind, wo sich Menschen aufhalten, ermöglicht Public Sensing eine Datenerfassung auf urbanem Maßstab. Die Verfügbarkeit dieser Datenmenge ermöglicht die Entwicklung neuer, Realwelt-bezogener Anwendungen, die unser alltägliches Leben nachhaltigen verändern können.

Die Entwicklung von Public-Sensing-Systemen in großem Maßstab birgt jedoch einige Herausforderungen. Systeme müssen die Anfrage von unterschiedlichsten Daten unabhängig von der Mobilität der Menschen ermöglichen, um die Verwendung und Akzeptanz von Public Sensing zu fördern. Weiterhin müssen so viele Personen wie möglich am System teilnehmen, damit die notwendigen Daten erfasst werden können. Dazu muss das Public-Sensing-System die Ressourcen der Teilnehmer, insbesondere den Energievorrat der Smartphones, so weit wie möglich schonen. Teilnehmer werden sich aus dem System zurückziehen, wenn die Teilnahme die Akkulaufzeit ihres Smartphones signifikant beeinträchtigt. In dieser Arbeit betrachten wir die zuvor genannten Herausforderungen im Kontext zweier Anwendungen: Der Erfassung von Innenraumkarten und der Erfassung von Umweltdaten in großem Maßstab.

Im Bereich der Erfassung von Innenraumkarten präsentieren wir zunächst ein System zur Ableitung von Innenraumkarten aus Bewegungsspuren der Teilnehmer. Im Gegensatz zu bestehenden Ansätzen konzentrieren wir uns darauf, ohne Verwendung zusätzlicher Informationen, wie z.B. WLAN-Fingerabdrücken, ein Maximum an Informationen aus den Bewegungsspuren zu extrahieren. Weiterhin präsentieren wir einen Ansatz zur Verbesserung von Innenraumkarten aus Bewegungsspuren mit Hilfe einer formalen Grammatik. Die Grammatik codiert dabei

strukturelle Informationen über ähnliche Innenraumbereiche anderer Gebäude. Die Verwendung der Grammatik erlaubt das plausible Vervollständigen einer unvollständigen Innenraumkarte, wobei zusätzlich die Genauigkeit fehlerbehafteter Beobachtungen verbessert wird. Diese Verbesserung wird von unseren Experimenten belegt, die zeigen, dass die Genauigkeit der besten Innenraumkarten aus Bewegungsspuren gerade der Genauigkeit der schlechtesten grammatikgenerierten Karten entspricht. Um die Energieeffizienz des Kartierungsprozesses zu verbessern präsentieren wir weiterhin ein generisches Qualitätsmodell für Innenraumkarten. Dieses Qualitätsmodell wird von einem Planungsalgorithmus verwendet, der die energieintensive Erfassungssensorik eines Smartphones deaktiviert, während es sich in einem Bereich aufhält, der bereits mit hoher Qualität erfasst wurde. Dadurch kann bis zu 15 % der Energie für die Datenerfassung eingespart werden.

Im Bereich der Umweltdatenerfassung in großem Maßstab stellen wir zunächst das Konzept des virtuellen Sensors als mobilitätsunabhängige Abstraktion vor. Anwendungen konfigurieren virtuelle Sensoren, um Informationen an einer festen Position mit einer vorgegebenen Datenrate aufzunehmen. Das Public-Sensing-System wählt dann Smartphones in der Nähe jedes virtuellen Sensors aus, um die tatsächliche Datenerfassung auszuführen. Weiterhin präsentieren wir mehrere Optimierungsanätze um die Energieeffizienz von Public Sensing zu verbessern. Bei der lokalen Optimierung koordinieren die Smartphones in der Nähe jedes einzelnen virtuellen Sensors selbstständig, welches Gerät Daten erfassen soll. Somit wird vermieden, Daten mehrfach zu erfassen. Die lokale Optimierung kann die Effizienz der Datenerfassung um bis zu 99 % verbessern, während die Datenqualität im schlechtesten Fall um höchstens etwa 10 % sinkt. Weiterhin präsentieren wir eine globale Optimierung, die ein datengetriebenes Modell nutzt, um eine Untermenge der wichtigsten virtuellen Sensoren zu bestimmen. Daten werden nur für diese Untermenge erfasst, während die Messwerte für alle anderen virtuellen Sensoren aus dem Modell abgeleitet werden. Dazu stellen wir eine Reihe von Online-Lern- und Kontrollalgorithmen vor, die die Erstellung eines Modells in Stunden oder sogar nur Minuten ermöglichen, und die kontinuierlich die Gültigkeit des Modells überprüfen. Unsere Untersuchungen zeigen, dass die globale Optimierung bis zu 80 % Energieeinsparung bietet. Dabei bleibt der Fehler in abgeleiteten Temperaturmesswerten bis zu 100 % der Zeit unter einer Fehlerschranke von 1° C.

Part I.

Introduction

1. Introduction

1.1. Motivation

The proliferation of modern smartphones such as the Apple iPhone or Google Android Phones has given rise to *Public Sensing*, a new paradigm for sensor data acquisition [CEL+08]. In Public Sensing, data is gathered using spare resources of commodity smartphones. These smartphones contain a wealth of sensors and are carried and maintained by participants, i.e., ordinary people willing to share their resources with the public. As these sensors are already deployed wherever there are people present, data collection is enabled at an urban scale. Having access to such a wealth of data facilitates the creation of applications depending on real-world data in a way that may have a lasting impact on our everyday life [HD09]. Common examples of such applications include finding social hotspots ("Vibe of the City") [MPL+11], traffic monitoring [TRL+09] or large-scale environmental monitoring [KBRL09, MSN+09, WAK+10].

Traditionally, real-world measurement data is either collected manually by experts or acquired automatically using static sensor networks specially crafted for each sensing task. Both approaches are labor-intensive and expensive and are thus rarely ever used beyond individual applications. Furthermore, static sensor networks are managed by a single authority, which is problematic when operating on an urban scale, where areas of interest are governed by different authorities and thus individual agreements on the placement of sensors have to be negotiated.

Public Sensing represents a move away from this centralized paradigm, as data collection is fundamentally distributed in the hands of the people. As smartphones are already deployed, there is no monetary up-front cost to execute a sensing task, enabling both experts and "Citizen Scientists" [CHK08] to use real-world data on a large scale. In addition, this solves the problem of negotiating sensor deployment locations, as participants carry sensors to any accessible area.

This opportunity to set up data collection campaigns enables new ways to use the resulting data. Previously, availability of and access to data was controlled by the funding entity of the corresponding measurement campaign as a way to compensate for the cost of running the campaign. In such cases, access to data carries a significant upfront monetary cost or comes with restrictive licensing issues. One example of this problem is the availability of data for geo information services (GIS data, commonly referred to as maps). GIS data can be bought from for-profit companies, e.g., NAVTEQ. If such a (substantial) financial investment is not practical or desired, Google offers access to its Maps service free of charge, but reserves the right to restrict data access on its own terms or to add advertising information into the available data [goo14]. In contrast, the OpenStreepMap Project uses data collected by volunteers in a Crowd Sensing system to build a GIS database which can be copied and used free-of-charge by anyone [HW08, opeb]. Thus, Public Sensing as a data provider can accelerate the creation of new applications and services, based on publicly available data, as application developers can shift their focus from finding input data for their application to creating novel applications that make use of the wealth of available data.

Another major benefit of "free data" is an increased environmental awareness of individuals [WAK+10]. Environmental influences such as noise pollution or air pollution (CO or fine particulates, for instance) are becoming an increasing problem [Com96, Com13]. In many countries, legislation exists that dictates measures to reduce such influences, e.g., erecting noise-reflecting walls or imposing speed limits on road traffic. However, to begin the deployment of such measures, a problem such as excessive pollution must first be detected. Currently, this often involves lengthy lobbying campaigns fueled by suspicions that a problem might exist, before even initial measurements are taken to verify (or disprove) the suspicion, let alone to ascertain the extent of the problem. Furthermore, the actual deployment of such measures is based on extrapolations from a small number of measurements. In some cases, these extrapolation models are known to be inaccurate. Yet, they continue to be used for lack of a better method. With large-scale fine-grained data acquisition as provided by Public Sensing, problems can be detected as soon as they arise and extrapolation models can be continuously verified and improved—i.e., be grounded in reality at the location in question—or possibly be abandoned altogether in favor of real-time measurement data.

However, creating large-scale Public Sensing systems is not without its challenges. On the data gathering side, as many people as possible must participate in the system and thus provide a sufficient amount of data. To this end, the execution of sensing tasks in the Public Sensing system must be unobtrusive, i.e., participants must not be required to interact with the Public Sensing system or even to go out of their way to facilitate sensing task execution. Furthermore, the system must conserve the resources shared by participants as much as possible, with the main concern being energy. Participants will withdraw from the system, when participating significantly impacts the battery life of their smartphone. On the data requesting side, an interface is required that allows to specify arbitrary sensing tasks independent of the mobility and population of participants and thus facilitates user acceptance of Public Sensing.

We address the aforementioned issues in this thesis in the context of two applications: Indoor map generation and large-scale environmental data acquisition. Our main focus is the development of optimization algorithms that reduce the data acquisition workload subject to a user-specified quality bound. Rather than blindly getting all the data there is to get, we show how to select a reduced set of effective data readings out of the wealth of data potentially provided by the Public Sensing system at any given time, gathering only information that is actually relevant to the task at hand. Furthermore, we show how to apply model-driven approaches in the context of Public Sensing to infer the missing information from effective readings. In combination, optimization approaches and model-driven inference provide data that is sufficient for the application at hand (i.e., satisfy the user-specified quality bound) at a much reduced energy cost, thereby making large-scale Public Sensing viable.

The next subsections give a more detailed view on the problems we face in deploying large-scale Public Sensing systems and on our approaches to tackle them for each of the application contexts. In the following, Section 1.2 summarizes our contributions and Section 1.3 presents the structure of this thesis.

1.1.1. Indoor Map Generation

A very basic requirement for Location Based Services, e.g., navigation applications, is the availability of map data. Maps provide a mapping of real-world

objects, e.g., rooms within a building such as "Room 2.164", to positions, e.g., coordinates such as $(48.745205, 9.106515)$. As presented before, OpenStreetMap was one of the earliest Public Sensing projects with the goal of creating a free map of the world [HW08]. Volunteers record odometry traces using the GPS device in their smartphones. The resulting traces are combined to remove measurement errors and to determine the type of path (road, freeway, etc.) a user traveled.

While this approach works very well for outdoor settings, creating indoor maps is a far more complex task, mainly due to the unavailability of absolute positioning systems like GPS or Galileo. In the past, indoor maps had to be created in a manual effort by converting building blueprints (often not available in digital form) or by tediously mapping a building using 3D laser scanners. Fueled by the research on SLAM (Simultaneous Location and Mapping) in the robotics community, several approaches for indoor mapping from odometry traces recorded by participants using their smartphones have emerged, e.g., [AR12, AY12, SCC12], [PBD+14]. In these approaches, odometry traces are recorded as relative measurements from an Inertial Measurement Unit (IMU) such as the built-in acceleration sensors of a smartphone or an external foot-mounted sensor.

All of these approaches face the problem that vast amounts of data are required to fully and accurately map a given building, for a number of reasons:

1. IMU readings are inaccurate, typically due to sensor drift. Step lengths and turning angles are frequently recorded incorrectly, so that a large number of odometry traces is required to reliably identify the building layout.
2. Odometry traces only carry information about exactly those positions a user passed through and no information about the surroundings. For instance, when a participant moves through a hallway several meters wide, a single odometry trace does not provide information on the width of the hallway. Thus, multiple observations of each part of the interior are required to find the full geometric extent of rooms and hallways. Furthermore, each and every room must be visited to obtain a complete building layout.

As argued before, gathering large amounts of data also consumes large amounts of energy. We therefore leverage observations on architectural principles to facilitate the creation of indoor maps from a much smaller amount of information.

To this end, we first address the problem of building an indoor map directly

from odometry traces. Existing approaches on the one hand rely on a large number of observations to average out individual errors. On the other hand, they are either computationally complex or require the acquisition of additional features, e.g., WiFi fingerprints, to combine observations into a single map. Thus, we look at how we can extract the maximum amount of information from observations, i.e., how to improve the overall process of indoor map construction from odometry traces. We present a system that can smooth out most data recording errors of each trace, even when only few traces are available. Furthermore, we present a novel approach to find the indoor room layout that, in contrast to existing approaches, does not require additional features to identify individual rooms.

Second, we present an interpolation approach for indoor maps. If the amount of trace data that is input to the indoor map construction algorithm is reduced, the amount of features represented in the indoor map will decrease. Even when all available information is used, some parts of the building may not be mapped, as they have never been visited. To alleviate this problem, Becker et al. presented an *indoor grammar* that encodes knowledge about architectural principles as well as actual geometric and topological information taken from similar floors (e.g., other floors from the same building) [BPF+13], [PBD+14]. Applying this grammar to the problem of indoor map construction from a small amount of input data, we show an algorithm that, given a trace-based indoor map, will interpolate a grammar-based indoor map that extends the trace-based map to a plausible layout for the entire floor while explaining the observations from odometry traces.

Finally, to optimize the process of data acquisition, we present a generic quality model for trace-based indoor maps that quantifies the state of the map for various areas of the indoor space. Areas of low quality represent parts of the floor where additional information is required, whereas in areas of high quality, energy can be saved by not acquiring any more data without sacrificing map accuracy. This quality model is used by a scheduling algorithm, instructing a smartphone to disable its energy-intensive sensors when it enters a well mapped area.

1.1.2. Large-Scale Environmental Data Acquisition

Another common application for Public Sensing systems is the observation of environmental phenomena, thus serving as a flexible and efficient replacement for

traditional static sensing systems. Several existing approaches, e.g., [KBRL09, MSN+09, SM08, WAK+10, MPLM11], show the effectiveness of Public Sensing for this task. However, these approaches do not address the aforementioned specific issues for building viable Public Sensing systems.

To address these issues, we present the DrOPS system. DrOPS solves the problem of how to specify sensing tasks through the concept of *virtual sensors* (*v-sensors* for short). V-sensors provide a mobility-transparent abstraction of the Public Sensing system. They are configured to report a set of readings at a client-defined sampling rate at a given position, thus presenting a view on a (virtual) static sensor network. The Public Sensing system then selects nearby smartphones to provide readings for a v-sensor. Thus, DrOPS can be used as a drop-in replacement for a static sensor network. This, in turn, enables *Request-Driven Execution*, i.e., the Public Sensing system performs work only when data consumers requested data by specifying a set of v-sensors they are interested in. While the Request-Driven Execution is effective with respect to providing requested data, its efficiency can be improved further. The Request-Driven Execution collects readings from all participating smartphones near any specified v-sensor. When only a single reading is desired, or, more generally, k readings were requested (k-coverage), all but the first k readings are discarded. Thus, the energy for recording and transmitting these readings is wasted. We tackle this problem in our *Local Optimization* where we reduce the effort for data acquisition by adjusting the number of readings taken for each individual v-sensor.

While the v-sensors are a powerful abstraction, DrOPS faces another problem: A v-sensor is temporarily unavailable if no participating smartphone is in its vicinity. Clearly, the probability for a v-sensor being unavailable increases with an increasing v-sensor sampling rate and a decreasing density of participating smartphones. We alleviate this problem by a model-driven data acquisition approach, which uses a model to infer readings of unavailable v-sensors from other, available data. Moreover, we save energy for sensing and communicating data on mobile devices by inferring readings of v-sensors even if they are available when inferred readings are sufficiently accurate, thus enabling a *Global Optimization*.

For model-driven approaches to provide accurate inferred readings, a minimum set of input data from available v-sensors is required. Thus, the basic global optimization performs well in dense networks, where most v-sensors are

well covered (and thus available), e.g., in a busy city center or a business area at lunchtime. Collecting the minimum set of data is a problem in (partially or completely) sparsely populated areas, e.g., business areas during off-hours or housing areas during business hours, where the density of smartphones is overall low, or when the most interesting v-sensors are unavailable while many less interesting v-sensors are available. We address this challenge by providing an adaptive extension to our approach that enables operation in both dense and sparse networks.

1.2. Contributions

In the area of indoor map generation, we present *MapGenie*, a system for indoor map generation from odometry traces recorded using inertial sensors. MapGenie improves over existing systems by taking into account knowledge about architectural principles and the building exterior. In detail, our contributions are as follows.

1. We detail a novel approach for detecting rooms from odometry traces in the indoor environment. In contrast to existing algorithms, our approach can fuse observations from several traces solely based on features extracted from these traces, i.e., does not rely on observations of features external to the traces, such as WiFi fingerprints.

2. We show how to use architectural principles encoded in a formal grammar to extend an incomplete observed indoor map to a plausible full indoor map using a grammar-based map layout derivation algorithm.

3. We present a generic quality model to quantify the available information in the indoor maps for parts of a building. The quality model can be used to determine which parts of the building are insufficiently mapped and need additional observations. We demonstrate how our quality model can be used in a scheduling algorithm for improving the energy efficiency of the mapping process. The scheduling algorithm disables the sensors of a smartphone when the device is located in an area with sufficient quality, i.e., reduces the energy spent on observing unnecessary information.

Parts of this work have been previously published in [PBD+14], [BPDR14]. These works were a collaborative effort, where others have contributed as follows.

In [PBD+14], Peter et al. contributed the concept of recording odometry traces using Zero Velocity Updates and the concept of aligning odometry traces to exterior walls (cf. [PHF11]). The trace-based modeling component was developed by Christoph Dibak in a thesis supervised by the author [Dib13]. Based on his work, the author developed an improved trace-based modeling component as presented in this thesis. The Android App is the work of Naresh Nayak under the supervision of Patrick Baier, while the grammar for the construction of indoor layouts is original work by Becker et al. [BPF+13]. The algorithm for incorporating observations of floor plan objects derived from odometry traces is original work of the author.

In [BPDR14], the quality model and the mobility model are original work of the author. The energy model and the energy-efficient mapping approach and were developed by Patrick Baier. Furhermore, the student project of Bonet et al. was invaluable help in developing the evaluations for this publication [BRW14].

In both publications, Fank Dürr provided guidance for the project and helped to improve the presentation of concepts and results.

In the area of large-scale environmental monitoring, we present *DrOPS*, a system for model-*Dr*iven *O*ptimization of *P*ublic *S*ensing systems. DrOPS provides a range of optimizations to reduce the data acquisition workload, and thus the energy consumption, for environmental monitoring. In detail, our contributions are as follows.

4. A Sensor Network Abstraction to build a simple application interface to Public Sensing systems and to enable the Request-Driven Execution of sensing tasks. The Sensor Network Abstraction introduces the concept of the v-sensor that can be placed like a sensor in a static sensor network. The role of the v-sensor is then filled by nearby participating devices.

5. A series of distributed coordination algorithms for k-coverage, i.e., to select k devices for each v-sensor to fill its role. These algorithms aim to minimize the effort of data acquisition by reducing the number of redundant readings taken. Selecting one of these algorithms allows to trade off decreased energy consumption for increased robustness of the data collection process.

6. An approach to infer readings for spatially distributed phenomena from a reduced set of effective readings using multivariate Gaussian models. Thus, DrOPS can reduce the required effort for data acquisition while still providing data matching a client-defined quality threshold. In contrast to existing approaches, it does not require a prototypical deployment. DrOPS can obtain training data from a running sensing task and create a model suitable for model-driven sensing in a matter of minutes. To this end, it uses an online learning algorithm (OLA) for the quick construction of models and an online model validity check algorithm (MOCHA) to verify whether a model still accurately reflects the underlying phenomenon.

7. An improved multi-round execution model that extends DrOPS to compensate for unavailable v-sensors. The availability of v-sensors is learned online during query execution and the data acquisition effort is adapted so that—if at all possible—sufficient effective data from v-sensors is recorded, even when an a-priori unknown number of these sensors are unavailable.

Parts of this work have been previously published in [PDR11, PSA+13, PSDR13]. These works were a collaborative effort, where others have contributed as follows.

In [PDR11], the clustering algorithm used for multi-hop communication was developed by Daniel Fischer [HFP12]. All other contributions are original work of the author.

In [PSA+13], the model driven approach for sensor data acquisition is original work of the author. Implementation and experiments in the real-world testbed were carried out by a group of students under the supervision of the author [TMA12, Alt12]. The implementation of the simulated environment was advanced by a number of theses supervised by the author [Fro11, Sta11].

The concepts presented in [PSDR13] are original work of the author. This work was inspired by a number of theses supervised by the author that helped to advance the understanding of the topic [Til12, Neu13].

In all publications, Fank Dürr provided guidance for the project and helped to improve the presentation of concepts and results.

1.3. Structure

This work is structured in three parts. The remainder of the first part presents background information on Public Sensing and its enabling technologies in Chapter 2 and an overview of our system model and architecture in Chapter 3.

The second part is concerned with the challenge of constructing an indoor map using Public Sensing. In Chapter 4 we present an overview of the MapGenie system. Chapter 5 details how to extract an indoor map from observations. We show how to improve this map using a grammar-based map generation algorithm in Chapter 6 and present an approach to improve the energy efficiency of the mapping process using a quality model in Chapter 7. Chapter 8 presents the evaluation results of a real-world experiment before we take a look at related work in Chapter 9.

In the third part, we discuss the use of Public Sensing for environmental monitoring. Chapter 10 presents our Sensor Network Abstraction and gives an overview of the operation of DrOPS. We detail the local optimization for DrOPS in Chapter 11. Our model-driven approach for global optimization is discussed in Chapter 12. Chapter 13 presents the evaluation results for the performance of DrOPS from a small-scale real-world testbed and a large-scale simulated setting, followed by a discussion of related work in Chapter 14.

Finally, we conclude this work in Chapter 15 with a summary and outlook onto future work.

2. Background

In this chapter we present background information on Public Sensing and on the technologies Public Sensing systems build on. We begin with a more in-depth discussion of the characteristics and paradigms of Public Sensing. In subsequent sections, we discuss the properties of positioning systems and communication systems available in todays smartphones.

2.1. Public Sensing

The idea of Public Sensing (also known as Urban Sensing) is to use the near-omnipresent commodity smartphones of ordinary people as a large-scale mobile sensor platform that can—in populated areas—reach almost all places of interest. Public Sensing is one of several focal points in the research area of "people-centric sensing" [CHK08]. While Public Sensing deals with data acquisition for public use, *social sensing* is concerned with gathering data for the benefit of a group of (social) peers, e.g., finding social hotspots ("Vibe of the City") [MPL+11], and *personal sensing* deals with providing data to (and usually about) a single person, e.g., personal health monitoring [CEL+08].

The foundation for the idea of Public Sensing was laid by Liu et al. [LBD+05]. They compared a system of static sensors to a system of the same number of mobile devices. A formal analysis regarding the coverage area and object detection speed showed the benefit of using mobile devices. Later, Campbell et al. proposed to use smartphones as mobile sensors to fill the gaps left in the deployment of static sensor networks [CEL+06]. When smartphones came into wide-spread use amongst consumers with the release of the Apple iPhone, focus shifted to stand-alone Public Sensing systems, e.g., [HW08, CEL+08, AAB+07].

Using such mobile sensors provides several benefits compared to the use of stationary sensor networks. When there is an application that prompts the use

of a sensor network, the necessary devices are so expensive that only the bare minimum of devices is deployed. This leads to a sensor network that is specifically crafted for this one purpose and cannot be reused for other tasks. Thus, stationary sensor networks are both application-specific as well as expensive to deploy and maintain. Public Sensing, in contrast, does not require the purchase of devices or even the deployment thereof. Smartphones are carried by people regardless of participation wherever they go, without the need for the operator of a Public Sensing system to buy them beforehand or to actively deploy them – a major advantage, when monitoring a large area, as there is no need to negotiate places for deployment with a multitude of authorities.

People may choose to participate in the system for several reasons. Participation can yield an immediate personal benefit, e.g., when the participant is directly interested in the obtained data [HW08] or when the Public Sensing system operator offers a monetary compensation [YKG10]. Participation in the system may also be motivated by a delayed personal benefit, e.g., a tit-for-tat: If a person shares her resources with the system now to make the system viable, she can later use the existing system for her own data acquisition tasks. Furthermore, people may also participate for purely altruistic reasons, i.e., share their resources for the greater good without expecting any direct personal benefit from participation. Under any motivation, participation is likely to decline when participating in the system impacts user experience, i.e., prevents people from pursuing tasks as they would when not participating. This may take the form of the system interrupting the persons current task by constantly prompting for input, or by impacting the user experience of the respective participants smartphone by draining the battery.

To this end, Public Sensing systems can be classified by their interaction pattern [LEM+08]. *Participatory Systems* follow a "human-in-the-loop" paradigm, i.e., they require the participant to manually execute a sensing task. For example, this might involve moving to a certain location ("Go to the pedestrian precinct to take a noise measurement.") or answering a question about an observation ("How many cars are there waiting at the traffic light?"). *Opportunistic Systems*, on the other hand, completely exclude the participant from sensing task execution. They execute tasks only when a device is in the correct state for task execution (location, sensor exposure,...), i.e., running in the background with no user interaction and without requiring the participant to go out of her way.

While the wide-spread presence of smartphones is one factor that makes Public Sensing feasible, their capabilities are another. Even though smartphones are resource-constrained devices, their resources in terms of storage capacity, computation power, energy supply and communication capabilities as well as available bandwidth far surpass that of commercially available stationary sensor nodes. Furthermore, they offer a wealth of communication capabilities such as WiFi, Mobile Internet Connections, or Bluetooth, to communicate to internet services, other smartphones, or additional hardware augmentations. Typically, Smartphones come with a number of sensors built-in, e.g., ambient light and sound, gyroscope, accelerometer, compass, and an absolute positioning system such as GPS. When a sensor is not built-in, the smartphone may be augmented using USB or low-power wireless communication technologies. An example for this is the Broadcom WICED Sense [Bro14], which provides an external accelerometer, gyroscope and compass along with barometer, humidity sensor and thermometer to a smartphone via a Bluetooth LE connection.

2.2. Positioning Systems

To be able to interpret a data reading received from a smartphone, a client requires information about the context of the smartphone at the time the reading was taken. Given the unknown mobility of participants, the position at which the readings was taken is among the most important pieces of context information. Smartphones have access to a range of positioning systems, which we will review in the following. In the following section, we consider absolute positioning systems that provide absolute coordinates relative to earth. In subsequent sections, we focus on relative positioning systems, providing a position in relation to a known initial position. Note that we only consider positioning systems from a users' point of view. An in-depth review of the mathematical foundation of many of the systems presented here is provided by, e.g., Roth [Rot02].

2.2.1. Absolute Positioning Systems

Absolute positioning systems provide a device with a specific position on earth. To determine the position, these systems rely on a set of radio beacons in one

form or another, located at well-known coordinates. On the one hand, using an absolute positioning system requires either knowledge about all beacon positions or access to an on-line database of beacon positions. On the other hand, it also requires radio contact, typically line-of-sight, to a minimum number of beacons. While absolute positioning systems have a wide range of applications, their most common application is in wide-area outdoor environments. Commercial systems for indoor use are currently emerging.

In the following, we will first discuss three types of absolute positioning systems designed for outdoor settings: Satellite-based, terrestrial infrastructure based and terrestrial opportunistic systems, before moving on to a review of absolute positioning systems for indoor settings.

Satellite-based Systems

Satellite-based absolute positioning systems rely on a set of radio transmitters placed in earth orbit. The best-known systems in operation today are GPS, Glonass and Galileo. Given a sufficient number of satellites in operation, they provide a high accuracy location service with world-wide coverage. The satellites broadcast their identifier together with time information from an on-board atomic clock. A device on the ground or in the atmosphere receives the information from multiple satellites and uses the time information to estimate its position relative to the satellites. Given information about the satellite orbits, the device can compute the current position of each satellite relative to earth and thus derive its own absolute position on earth. Information about satellite orbits can be received from the satellites themselves at an extremely low data rate or, as an optimization, downloaded via an Internet connection ("Assisted GPS", AGPS).

Satellite-based systems offer very good accuracy and coverage, but they are not perfect. While most position fixes are exact, GPS only provides a position with a horizontal error of less than 22 m in 95 % of cases, even under perfect conditions. This error is frequently modeled as a normal distribution [Gur91, LWG+09]. When the number of satellites with a direct line-of-sight to the device is insufficient, the system instantly becomes unavailable. This can occur, for example, when traveling in woods, where the foliage shields the satellite signals, or in cities, where only a small patch of sky is visible between buildings. Furthermore, due

to the low data rate for downloading orbit information, it may take up to 12.5 Minutes after starting the positioning system on the device to obtain a first position fix when AGPS is not available. Even when AGPS is available, the first position fix is frequently delayed by several seconds until the device has received timing information from a sufficient number of satellites. Additionally, the energy consumption of running a receiver for satellite-based positioning systems is a major challenge in Public Sensing scenarios. Running a GPS positioning system on a smartphone for longer periods of time will quickly drain its battery. Finally, as satellite-based positioning systems require the concurrent operation of a large number of satellites for global coverage (24 in the case of GPS), operating such a system is beyond the capabilities of any single private entity. As a result, all current systems are operated by one or more nations.

Terrestrial Infrastructure-based Systems

As satellite-based systems are expensive to maintain, have a high energy cost on the receiver-side and have a high startup-time, other options for implementing positioning systems were investigated. Terrestrial infrastructure-based systems make use of radio transmitters located at fixed positions. The most notable of these technologies is triangulation using cellphone towers, e.g., in the Ericsson Mobile Positioning System [EMP]. In this case, existing cellphone towers are used as radio beacons, amortizing the maintenance cost. They are located at fixed, known positions and are placed densely enough that in most cases a sufficient number of cellphone towers is in range to perform a triangulation. In the majority of cases, a device can sample its position instantly at next to no extra energy cost, as the cellphone radio is almost constantly enabled.

The major drawback of terrestrial infrastructure-based systems is their varying accuracy. The accuracy of a position fix depends on the correct measurement of the distance to the cell towers in range. However, due to reflections or moving obstacles, these measurements are frequently incorrect. While obtaining a precise position is possible, most position fixes are just accurate enough to determine which neighborhood a device is located in.

Terrestrial Opportunistic Systems

Terrestrial opportunistic positioning systems rely on the presence of WiFi Access Points as radio beacons, which are found in abundance in most households and businesses. Their exact position is unknown, making triangulation infeasible. However, in urban areas, they provide an extremely dense network of short-range radio beacons, allowing for a fingerprinting approach: In a training phase, the set of visible WiFi access points is recorded for known positions. To take a position fix, a mobile device performs a WiFi scan and determines the most likely location by matching the set of WiFi access points to the training database. Such a system is, for example, operated by Skyhook Wireless [Sky].

While Terrestrial Opportunistic positioning systems do not provide completely accurate positions, their accuracy is fairly constant and, in urban areas, sufficient to determine the closest building and the side of the street. However, as the access points are not subject to a central management entity, they may be moved or disappear, or new access points may be added at any time, requiring constant updates to the underlying database.

Indoor Positioning Systems

The principles from terrestrial opportunistic positioning using WiFi can, to a limited extent, also be applied to indoor settings. In buildings with several access points, fingerprints containing the set of visible access points along with their received signal strength are recorded at known positions in a training phase [BP99]. In the operation phase, to obtain a position fix, a device measures the current set of visible access points and their received signal strength. It then selects the position from the database that has the closest match to observed values, e.g., by picking the position that minimizes the mean square difference of recorded and observed signal strength values. Such systems provide an accuracy of a few meters and are thus sufficient to determine the current room (for rooms larger than the accuracy of the system) or the current hallway. However, indoor WiFi positioning is still a subject of research to date. Systems like Active Bat [WJH97] use triangulation via ultrasonic sound to locate mobile devices with an accuracy in the range of a few centimeters. Their drawback is that the infrastructure part needs to be set up with extreme care to achieve this level of accuracy.

Currently commercially emerging approaches for absolute indoor positioning follow the infrastructure paradigm, where beacons are deployed by the building management. These beacons may be Bluetooth LE transmitters (e.g., Apple iBeacon [ibe15]) or specialized overhead lights (e.g., Philips Connected Light, [N.V14]). The position may be determined by simple proximity to a beacon, through fingerprinting, or through triangulation.

2.2.2. Relative Positioning Systems

Relative Positioning Systems provide the current position relative to some previous position ("initial position") based on observations or estimations of movement direction and movement speed of a device. The current position can only be given in absolute coordinates (relative to earth) when the absolute coordinates of the initial position or some intermediate position are known. Before the availability of absolute positioning systems with global coverage, *dead reckoning* was a common method for relative positioning in naval and aerial navigation. The speed of a ship and all changes in heading are constantly recorded. Integrating these values over time gives the relative position of the ship from the last known position.

The benefit of relative positioning systems is that they do not require communication with the outside world, i.e., the position can be tracked using sensors local to the device only. Thus, they are applicable when there is either no network connectivity or when signals from beacons cannot be received with sufficient quality, such as GPS signals inside buildings [KBG+10]. The challenge in using relative positioning systems is to provide accurate estimates for the movement vectors. In naval or aerial applications, the heading can be measured as an absolute value (compass), but the speed is measured relative to a medium that is itself moving, i.e., relative to water or air. This, along with ever-present measurement errors, leads to an inaccurate estimated position, even when the initial position is perfectly known. Furthermore, the error of the position estimate increases over time.

Inertial Navigation in Smartphones

In the context of positioning systems for smartphones, movement can be tracked using built-in sensors, i.e., accelerometer and gyroscope, as well as the compass

[Har13, XSF10]. The accelerometer detects changes in speed, e.g., whether a user speeds up or slows down. The integral of the acceleration provides the momentary speed at any point in time. Similarly, the gyroscope detects rotations and thus changes in heading. Speed and heading can then be integrated a second time to perform dead reckoning.

However, this approach suffers from two major problems. On the one hand, readings due to actual user movement are mixed with other movement, e.g., the smartphone being placed in a flapping jacket or the up-and-down movement occurring during the gait cycle. On the other hand, sensors are inexact and exhibit drift errors. Compared to aerial and naval applications, the forces observed for changes in heading are very small. Thus, the magnitude of the error becomes a sizable fraction of the magnitude of the observed value. The position error grows cubically in time over the runtime of the system since the last known position [Woo07, Fox05]. Thus, the accumulated error after dead reckoning is so massive that after just a few steps, the position is essentially unknown. While the compass may be used to correct the recorded heading in outdoor settings, this approach does not transfer to indoor settings as measurements are easily disturbed by metallic parts of the building construction or live electrical wiring.

As we will show in the following sections, several approaches exist to improve the accuracy of estimated positions.

Zero Velocity Update Protocol

One approach to improve the accuracy of inertial navigation in smartphones is to reduce the growth of the error over time. To this end, Foxlin presented the *Zero Velocity Update Protocol* (ZUPT) [Fox05]. Based on a foot-mounted inertial measurement unit, it recognizes the gait cycle of a walking user. The gait cycle consists of a stationary phase and a stride phase. The stationary phase, where the foot rests on the ground, is detected. From measurements taken during the stride phase, i.e., the time since the last stationary phase, the length and orientation of the last step are computed as a movement vector using dead reckoning. Furthermore, as during the stationary phase acceleration, rotation, and velocity of the sensor are known to be zero, the difference of readings during the stationary phase and these known values can be used to compensate for the

drift error during the last step. Thus, the integration of acceleration and rotation measurements is performed for each step individually instead of the full runtime of the system. With ZUPT, the error is improved to growing linearly in the number of steps since the last known position.

Landmarks & Map-Matching

Another approach for better position estimates is to use a map of landmarks to periodically reset the estimated position to a known position. A landmark is a position with easily detectable features and known coordinates, such as an intersection or a location with a distinct signature of sensor values. For example, if movement is restricted to a known network of paths, map-matching helps to reduce the noise in movement vectors [CCR10]. When a distinct turn is detected, it can be assumed that the participant is currently turning at an intersection. In case the position estimate after the turn is located outside the path network, it can be reset to the closest intersection. Furthermore, when a movement vector ends outside of any known path but intersects the boundary of the path at a very small angle, this observation is assumed to be caused by a drift error. Thus, the current position estimate is shifted back onto the last path.

If no a-priori map is available or the movement is not restricted, more general landmarking techniques can be employed. In Pazl [RKM13], devices track their (estimated) position using dead reckoning while recording WiFi fingerprints at the same time. WiFi fingerprints and corresponding position estimates from multiple trips and devices are combined to derive a set of common anchor points, i.e., places with distinct WiFi fingerprints that were visited multiple times. For each such anchor, coordinates are computed as an average of the corresponding individual position estimates. The resulting knowledge can then be used to improve the position estimates along the remainder of each trip passing an anchor point by resetting the estimated position to the position of the anchor at the time of passing.

Particle Filters

Another approach to computationally reduce the noise in observed movement vectors without knowledge of the environment is the application of particle filters

[KR08, WH08, WKB08]. Particle filters combine current observed movement with a given mobility model, e.g., computed from earlier movement vectors with added white noise. In general, they work as follows. First, the observed movement is split into a set of movement vectors, e.g., resembling individual steps as provided by ZUPT. The particle filter then iterates over this sequence of movement vectors. Initially, a set of particles is created at the initial position. Each particle is then displaced according to a mobility model, representing the expected movement of a person. Based on the similarity of particle movement and the current movement vector, particles are assigned weights. Finally, the set of particles is modified to represent the distribution of weights, i.e., particles with bigger weights are replicated in the set. The filter then continues with the next movement vector.

While it has been shown that particle filters can find the original trace of a user from noisy observations with very little error, this performance comes at a price: The complexity in terms of computation and storage is prohibitive for many real-world applications. Thus, they are often not applicable without the addition of other techniques such as the use of landmarks [Har13].

2.3. Wireless Communication Systems

Wireless Communication Systems are a fundamental requirement for Public Sensing. Wide-area wireless communication is used to communicate tasks to smartphones and resulting readings to clients. Local-area communication allows smartphones to coordinate their sensing effort with other devices in the vicinity. Personal-area communication can be used to augment the capabilities of a single smartphone by, e.g., attaching additional sensing hardware via a wireless connection. In the following, we present one communication technology for each of these scenarios.

2.3.1. Cellular Networks

Cellular networks, originally developed for voice communication, today provide wide-area network access with virtually universal coverage in any moderately densely inhabited area. As they are designed for connecting mobile devices running on a limited energy supply, they exhibit a distinct energy usage characteris-

tic, making them suitable for only a subset of communication patterns. Balasubramanian et al. [BBV09] analyzed the energy usage characteristics of smartphones on a 3G network. They found that cellular network transceivers have two distinct power modes. While the connection is idle, the transceiver enters a low-power mode. When data is to be exchanged, the transceiver switches to a high-power mode. However, switching power modes is not instantaneous. Especially when in high-power mode, a transceiver delays switching to low-power mode for over 10 s to wait for the possible arrival of new data. This tail time with high energy usage makes 3G communication better suitable for either bulk transfer at medium data rates (tail energy is amortized over a lot of data) or burst transfer at low duty cycles (tail time is small compared to overall operation time).

2.3.2. Wireless LAN

Wireless LAN networks using the IEEE 802.11 family of standards, commonly called WiFi, have become the dominant technology to extend local area networks to mobile devices. Their traditional application is in infrastructure mode, where an Access Point, connected to the local LAN, advertises its wireless network to all devices in radio range (10 to 50 m, depending on the presence of obstacles). When a device joins the wireless network, the Access Point relays messages between devices on the wired and wireless networks.

Another application of Wireless LAN technology is Ad-Hoc Networking. WiFi Direct [CMGSS13] is proposed as a new standard for ad-hoc high-speed data exchange between a group of devices without the need for the presence of an Access Point. Furthermore, this mode can be used to create mobile Ad-Hoc networks (MANETs). These employ a routing protocol to relay messages across multiple hops between devices that are not in direct communication range, e.g., Geographical Routing [KK00] or Ring-based Routing [HFP12].

Wireless LANs provide a higher data rate than cellular networks. However, this comes at the cost of a higher energy consumption. While the cost of actually transferring data is lower than in cellular networks, the energy for basic system maintenance is higher [BBV09, XSK+10]. Thus, Wireless LANs are more suitable when higher duty cycles are required.

2.3.3. Bluetooth

Bluetooth is a personal-area networking technology present in virtually every modern mobile device (smartphones, laptops, etc.) today. It is commonly used to wirelessly attach peripherals to a device. In its original form, Bluetooth provides a "virtual wire" abstraction. This requires device owners to complete a pairing process to authenticate devices and peripherals to each other. Furthermore, it requires devices to constantly maintain their connection, causing a significant energy drain even when no data is transmitted.

To open Bluetooth communication for new applications, Low-Energy Bluetooth (BLE) was specified. On the one hand, BLE allows communication without prior pairing, thus enabling the use of BLE devices as beacons for positioning systems [ibe15]. On the other hand, BLE also loosenes the requirements on connection maintenance, thus allowing communication at extremely low duty cycles. This mode is frequently used to connect a smartphone to additional sensor devices for personal fitness tracking or portable environmental sensors [Bro14]. The ultra-low duty cycle allows operation of suitable peripherals on a single coin cell for up to a decade.

3. System Overview

In this chapter we will give an overview of our Public Sensing system (PS system) in terms of the architecture of the software system and our assumptions on the underlying technology. We begin by identifying the requirements for a PS system in the following section before detailing our system model and the software architecture in subsequent sections. Note that our system model follows the general design of PS systems, e.g., [Eis08, KBP+08]. Therefore, our algorithms can easily be integrated into other systems.

3.1. Requirements

From our previous discussions, we identify the following key requirements for creating viable PS systems.

Quality-Aware Data Acquisition The system must aim to provide data of a sufficient (client-requested) quality. As the ability for data acquisition is upheld by volunteers, the system cannot guarantee the constant availability of the respective data. If data cannot be provided at sufficient quality, the client must be informed so that he can account for low-quality data in his application.

Opportunistic Operation To maximize data availability, the system must operate in an unobtrusive fashion so that data can be gathered wherever there are participating devices present, rather than where there are participating people present and willing to interact with the system.

Energy Efficiency The system must minimize the energy drain on participating devices to avoid a negative impact on a participants' user experience with her device. Namely, it must ensure that the battery lifetime is not reduced

to less than the recharge cycle duration, which is typically one day. To this end, it must exploit the optimization potential presented by the best-effort data acquisition requirement and thus limit its data acquisition efforts to provide the requested quality only.

Mobility-Independent Operation The system must operate without tracking the mobility of participating devices. Devices participating in the system move in a way that cannot be anticipated by a client or the system itself. Continuously tracking the position of participating devices is in conflict with the requirement on energy efficiency. Thus, the design of the system must take this unforeseeable mobility into account. On the one hand, the interface to the system must abstract from device mobility, so that clients can request data without knowledge of device mobility. On the other hand, the system must take into account that there may not be any participating device present at a point of interest and thus find suitable substitute data to fulfill the promise of best-effort data acquisition with sufficient quality.

Scalability Gathering data using PS is possible wherever there are people carrying smartphones. With an increasing population and both increasing population density as well as the sprawling of populated areas, the system must be scalable both in the number of participating devices and the size of the service area, managing a multitude of queries executed by millions of participants.

3.2. System Model

Our PS system consists of two major components, as depicted in Figure 3.1: The *gateway service* and the *participating mobile devices*. These work together to provide a data acquisition service for a predefined *service area*, i.e., a set of locations where data readings may be obtained. The size and shape of the service area is defined by the operator of the PS system and may cover anything in the range of a few square meters to the entire world. In the following subsections, we define the capabilities and responsibilities of the gateway and the mobile devices. Furthermore, we show how the system implements the scalability requirement, before we detail the failure model assumed in our system.

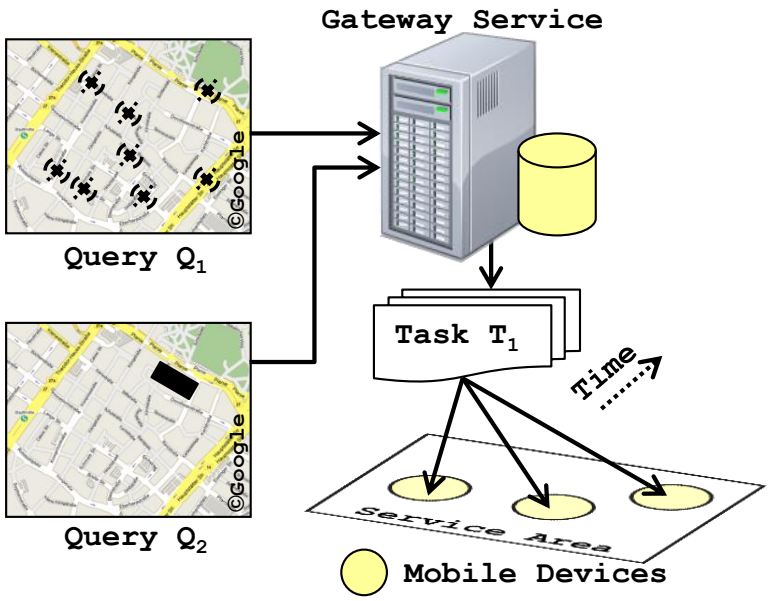


Figure 3.1.: Overview of the System Model

3.2.1. Gateway Service

The *gateway service* is located on the Internet and serves as a coordinator and central meeting point for data consumers and mobile devices. For data consumers, i.e., clients and applications, it provides an interface to submit queries for data and to receive the results of current and past data acquisition campaigns. The content of a query depends on the type of data acquisition to be performed. We will detail the *query model* for each of the applications considered in this work in corresponding Chapters 4 and 10.

For the mobile devices, it provides an interface to register as participating device and to submit sensor readings. The gateway generates *sensing tasks* for participating devices from currently active queries, defining the current data acquisition work to be performed, and stores all data readings subsequently taken by the mobile devices. Furthermore, the gateway computes additional information from stored data whenever required by one of our optimizations presented in later chapters.

3.2.2. Mobile Devices

The set of *participating mobile devices* is the core component of any PS system. These devices are commodity smartphones, carried by *participants*, i.e., ordinary

smartphone owners, who choose to share the resources of their smartphones with the system. We assume the mobile devices (through their owners) to exhibit pedestrian mobility. For indoor settings, we assume that each trip of a participant has a fixed destination. While the actual destination is unknown to us, we assume that participants travel on the shortest path from their current location to the destination. For outdoor settings, we make no further assumptions on the mobility of participants, i.e., movement is essentially random with a limited top speed.

Mobile devices are equipped with a number of sensors that can be either built-in or may be connected wirelessly, e.g., via Bluetooth. They can use environmental sensors (e.g., light, sound, temperature, air pollution) to observe their environment, inertial sensors (Accelerometer, Gyroscope, etc.) to observe their situation and, in outdoor settings, absolute positioning sensors (e.g., GPS, Cell-Tower triangulation) of varying accuracy and varying energy consumption to determine their location. We assume readings from environmental sensors to be exact. Readings from inertial sensors are assumed to be noisy, where the noise is represented by an unknown probability distribution added to the real value. The positioning sensors also provides noisy readings, modeled as a 2D Gaussian distribution with standard deviation σ_{vis}, centered at the true position of the device. To distinguish true positions and noisy positions, we define the following terminology. A position vector is denoted as $\overrightarrow{a}$. $\delta(\overrightarrow{a}, \overrightarrow{b})$ denotes the euclidean distance between the positions $\overrightarrow{a}$ and $\overrightarrow{b}$. The *true position* of an object a is indicated as $\overrightarrow{a}_{true}$. The *sensed position* $\overrightarrow{a}_{sens}$ is a noisy position, such as given by the positioning sensors. It holds that $\delta(\overrightarrow{a}_{true}, \overrightarrow{a}_{sens}) \geq 0$. In cases where we do not distinguish between true and sensed positions, the index is omitted.

Furthermore, mobile devices can communicate via a number of interfaces. Bluetooth and WiFi allow ad-hoc communication between mobile devices up to a certain maximum distance, denoted as the *ad-hoc communication range*. Communication to services on the Internet, e.g., registering with the gateway service or submitting data readings, is possible using either the aforementioned interfaces when a nearby access point (e.g., WiFi-Hotspot) is available, or using the (always available) mobile data connection, e.g., 3G or LTE. Every mobile device has a built-in battery that provides a limited energy supply for its operation. This battery is frequently recharged by the device's owner, e.g., every night.

3.2.3. Extensions for Scalability

To ensure scalability of the system, the PS system may be partitioned. To this end, the service area of the system is partitioned into smaller *partition service areas* by the operator, e.g., per city district or per street block. Furthermore, the gateway service is implemented as a distributed system consisting of multiple *gateway servers*, e.g., physical machines or virtual machines rented from a cloud service provider. Each partition service area is assigned to a gateway server. Mobile devices are informed of the partitioning by the gateway service when joining the system or when partitions are modified. Upon moving to a different partition service area, they register with the gateway server responsible for the new partition. Thus, each partition forms a self-contained PS system capable of independent operation, where the gateway service can geographically address all nodes in a partition. However, the ability for independent operation is limited when a query requires data from multiple partitions. In this case, the query is managed by one of the gateway servers responsible for any of the partitions the query is interested in, while other gateway servers only relay communication between the managing gateway server and the mobile nodes in their partition.

Defining the partitioning of the system requires defining the size of the partition service areas. While smaller partition service areas lead to a better distribution of the workload amongst gateway servers, this also leads to more frequent registrations for new partitions from mobile devices, thus violating our requirements on energy efficiency and mobility-independent operation. However, finding the best size of partition service areas is also a well-known problem in the design of cellular networks and can be solved using existing techniques (see the work of Akyildiz et al. for a survey of possible approaches [AH96]). We thus consider this problem to be out of scope for this work.

3.2.4. Failure Model

In this work, we assume the following failure model. All Ad-Hoc channels used for communication to nearby devices, i.e., Bluetooth and WiFi, are subject to omission failures, e.g., due to message collisions or temporary interference by other communication technologies, and crash failures, e.g., due to permanent interference.

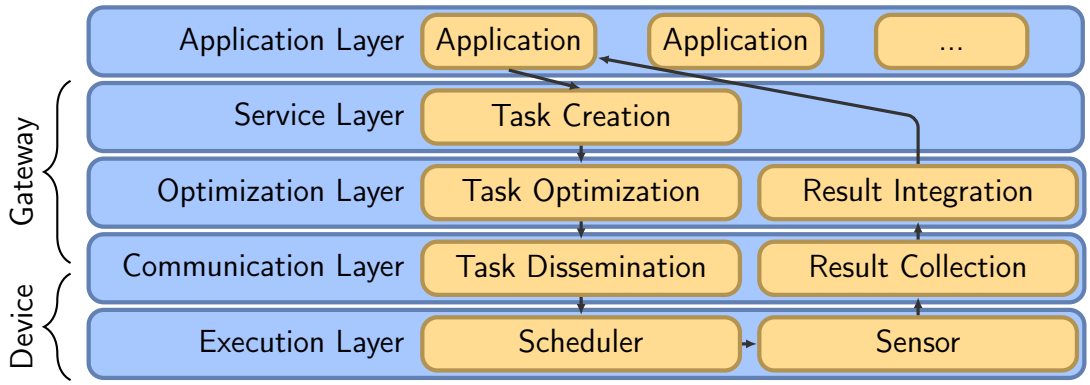


Figure 3.2.: Overview of the canonical system architecture

All other parts of the system are assumed to be failure-free. Omission failures by any mobile device or the infrastructure-based communication channel (WiFi-Hotspots and mobile data connection) would degrade the performance of our system to that of a failure-free system with fewer participating devices. The same holds true for omission failures of the gateway service or crash failures of mobile devices. Thus, we do not consider such failures in the design of our system.

Crash failures of the gateway service or infrastructure-based communication channels cannot be handled by the system. They would, in turn, cause the PS system to crash and must be handled by an operator outside of the system. Therefore, we do not consider them in the design of our system.

3.3. System Architecture

The architecture of our PS system is split into five layers as shown in Figure 3.2. In the following, we give an overview of the purpose of each layer and its components. The concrete operation of each component is dependent on the type of query to be executed and will thus be detailed in the corresponding sections.

The *Application Layer* represents the client side of a PS system, where observations are requested and resulting data is returned. It contains the applications that consume data provided by the system. To request such data, applications formulate queries that are passed to the gateway.

On the system side, the *Service Layer* represents the application interface to the PS system. It receives queries and converts them into tasks suitable for distribution to the mobile devices. These tasks are then passed on to the *Opti-*

mization Layer. This layer consists of two components. The *Task Optimization* component uses knowledge gathered from a-priori information or historic data to identify parts of the task (or entire tasks) that no longer need execution, e.g., where the result can be inferred from existing knowledge with sufficient quality, and modifies the task accordingly. The *Result Integration* component integrates results inferred from available information with the data that has been collected as a result of the task execution, e.g., it performs the inference and outputs a single result to the application layer.

The task, as modified by the Task Optimization component, is then passed to the *Communication Layer* for dissemination to mobile devices by the *Task Dissemination* component. Furthermore, the *Result Collection* component collects the data readings transmitted by mobile devices.

Finally, the *Scheduler* component of the *Execution Layer*, located on each mobile device, receives the task to be executed and instructs the device hardware to act accordingly, e.g., determine whether the device is in the correct location for taking a data reading, sample the appropriate sensing hardware, and transmit the reading to the Result Collection component.

Note that the focus of this work is on the operation of the gateway and the mobile devices, not on the communication network. Therefore, we assume the Task Dissemination component to provide a best-effort multicast service for communication from the gateway to the mobile devices and assume best-effort unicast communication service for the Result Collection component. The implementation of these semantics is subject to the specific networking technology used. We will note in our evaluations which implementation of these semantics has been chosen for the respective experiments.

Part II.

Indoor Map Construction

4. The MapGenie System for Indoor Map Construction

The availability of indoor maps is a key enabler for the deployment of indoor location based services. To aid in the developement of such applications we present MapGenie, a system for the efficient generation of indoor maps. MapGenie derives maps from mobility traces recorded using Public Sensing, thus enabling the creation of indoor maps on the fly without relying on specialized devices or expert knowledge. We also show how MapGenie leverages knowledge about architectural principles to provide a complete map of a building floor even when only a part of this floor has been observed, and how to increase the energy efficiency of the mapping process.

We begin with an overview of the MapGenie system, detailing its components in relation to the overall system architecture (cf. Section 3.3) in the remainder of this chapter. Chapter 5 details how to record and process odometry traces to derive an indoor map purely from observations. To extend the floor plan in unobserved areas, we describe our algorithm for the grammar-based generation of indoor layouts in Chapter 6. In Chapter 7 we then introduce our Quality Model for indoor maps, which is used to improve the energy efficiency of the mapping process by reducing the number of repeated observations for accurately mapped areas. Chapter 8 presents the evaluation results of our system in a real-world setup, before we conclude this part with a discussion of related work in Chapter 9.

4.1. MapGenie System Overview

Figure 4.1 presents an overview of the components of the MapGenie system and their place in the overall system architecture (cf. Figure 3.2).

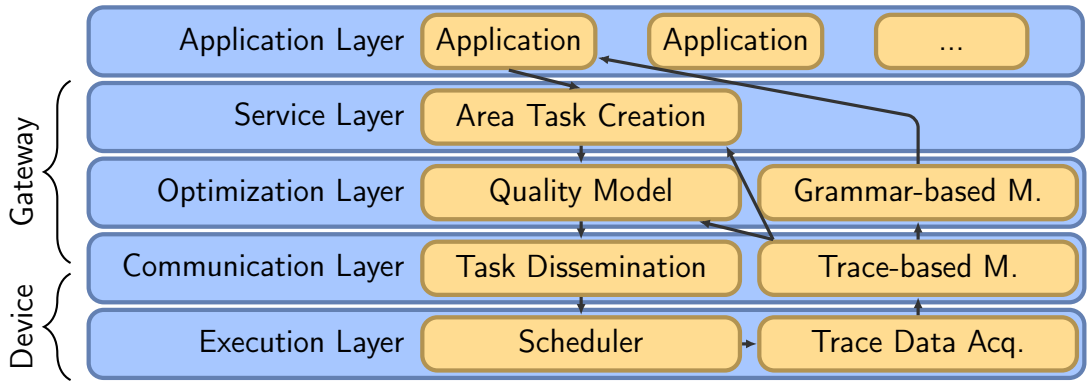


Figure 4.1.: Overview of the MapGenie Architecture

Applications specify a street address to query the MapGenie system for the indoor map of the respective building. The address is received by the *Area Task Creation* component. This component first converts the given street address to GPS coordinates of the building outline using existing (outdoor) map services, such as OpenStreetMap. Next, it generates a set of sensing tasks by partitioning the building into a set of *areas*, where an individual task is defined for each area. The definition of areas is subject to requirements of the mapping algorithm and will be discussed in Section 4.3 in more detail. The tasks are then handed off to the *Quality Model*, which provides an indication of how well an area has been mapped, i.e., classifying the area of each task as accurately mapped (AM-area) or inaccurately mapped (IM-area), based on the currently available knowledge about the building layout in the corresponding area. Note that this binary notation is sufficient, as we only require information on the set of areas for which additional information is required, rather than on the amount of information required for each area. The *Task Dissemination* component then distributes all tasks to all participating devices. Furthermore, as the definition of tasks as well as the quality values for individual tasks may change when additional information about the building layout becomes available, the Task Dissemination component also ensures that the set of tasks is up-to-date on all participating devices.

Participating devices run a *Scheduler* component. The Scheduler instructs the device to enable its Inertial Measurement Unit (IMU) to execute a sensing task or to otherwise disable it to conserve battery power. While the device is outside of a building, i.e., a GPS signal is available, it monitors the device position to determine when the user enters a building for which there are tasks available.

While the device is inside a building, it uses information about current and past mobility from the IMU as well as WiFi signal measurements to track its position inside the building and to determine which task to execute, if any.

When the IMU is enabled, the *Trace Data Acquisition* component records odometry traces. These are transmitted back to the infrastructure, where the *Trace-based Mapping* component derives the current trace-based floor plan, reflecting the observed building layout. The Trace-based Mapping component informs the Area Task Creation component and the Quality Model whenever the trace-based floor plan changes, so that tasks can be updated immediately. Furthermore, the trace-based floor plan is the input to the *Grammar-based Mapping* component, which uses knowledge about architectural principles and indoor maps of similar buildings to enhance the floor-plan, resulting in the grammar-based floor plan. To this end, the Gammar-based Mapping component performs a procedural reconstruction of the building layout to provide information in unobserved areas and to reduce observation errors in observed areas.

Our research focuses on the Trace-based Mapping component, the Grammar-based Mapping component and the Quality Model, which will be discussed in detail in subsequent chapters. For the remaining components, we briefly sketch their operation in the remainder of this chapter, after introducing our floor plan data model.

4.2. Floor Plans

MapGenie considers a *floor plan* to consist of spatial objects ("floor plan objects"), namely hallways and rooms (cf. Figure 4.2). Each room is accessible from at least one hallway. We assume individual hallway objects to be sections of the building's hallway network that are free of intersections. Hence, intersections of the hallway network are formed when the ends of two or more hallway objects meet. As every hallway network can be projected into this definition, this does not impact the generality of our approach. Furthermore, we define a fixed assignment of rooms to hallways based on accessibility. A room that is accessible from exactly one hallway is assigned to that hallway. A room that is accessible from multiple hallways is assigned to one of these hallways. Note that while the choice of hallway is arbitrary, the room must always be assigned to the same hallway.

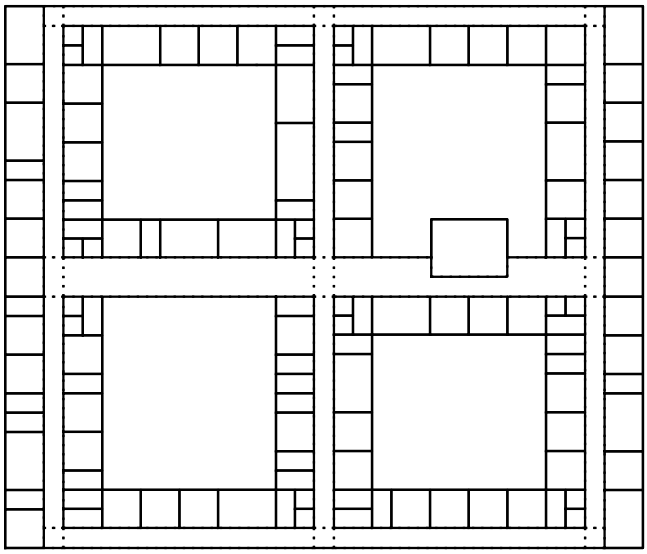


Figure 4.2.: Indoor Map consisting of Hallways and Rooms.

4.3. Area Task Creation

In order to control the process of energy-efficient mapping, the floor plan is partitioned into *areas* by the *Area Task Creation* component, generating a task for each area. Areas represent the granularity at which mapping can be controlled and subsequently optimized. Therefore, their definition can be provided by, e.g., an operator, depending on requirements of the concrete mapping algorithm used. The only restriction is that areas must be defined such that they only contain whole floor plan objects, e.g., do not cut a room in half. The most basic partitioning is to consider the whole building as a single area, thus providing binary control over the mapping process.

In line with the requirements of our energy-efficient mapping approach, we define an area to comprise a single hallway and all rooms assigned to (accessible from) that hallway, i.e., every hallway forms its own area. Note that full knowledge about the floor plan layout is required to find areas according to this definition. If there is no prior knowledge about the interior of a building, the required set of areas can be found through an iterative refinement. Initially, the entire building is covered by a single area. Whenever information about hallways is gathered, this single area can be split down into new subareas, allowing for more fine-grained control of the mapping process. This is feasible, as our quality

model places emphasis on finding hallways first before considering an area to be accurately mapped (see Chapter 7.1), thus allowing for the timely refinement of areas.

4.4. Trace Data Acquisition

The goal of the *trace data acquisition* component is to provide *odometry traces* from mobile users. These traces are periodically uploaded to the gateway service and later used by the Trace-based Mapping component to derive the trace-based indoor map. In our evaluation, odometry traces are collected using a foot-mounted IMU that detects steps by using a Zero-Velocity-Update protocol (cf. Section 2.2.2).

Formally, we define an *odometry trace* $t = (\overrightarrow{\pi}, S)$ where $\overrightarrow{\pi}$ is the initial position and $S = \{s_1, ...s_n\}$ is an ordered set of steps. Each s_i is given as a (relative) 2D vector (x_i, y_i), annotated with a timestamp $s_i.time$ denoting the time the step began. The position $\overrightarrow{p_k}$ after the k-th step is calculated as $\overrightarrow{p_k} = \overrightarrow{\pi} + \sum_{i=1}^{k} s_i$, annotated with a timestamp $\overrightarrow{p_k}.time = s_k.time$. To record an indoor trace we rely on inertial positioning, as absolute positioning systems (such as GPS) are not available with sufficient accuracy in indoor settings [KBG+10]. Their infrastructure-based indoor counterparts (using, e.g., WiFi beacons) are not usable for the reconstruction task at hand, as their set-up itself involves mapping, such as fingerprints of the received signal strength or the locations of the beacons. Therefore, we can only use GPS to determine the initial position $\overrightarrow{\pi} = \overrightarrow{\pi}_{sens}$ before entering a building [AY12]. Note that due to the possible inaccuracy in GPS (cf. Section 2.2.1), the sensed initial position $\overrightarrow{\pi}_{sens}$ may be different from the true initial position $\overrightarrow{\pi}_{true}$, i.e., translated from its true location.

Although we assume the use of a foot-mounted IMU, the relative positions of traces can also be derived from a variety of other sensors. Every modern smartphone has a built-in accelerometer, which can be used to track user movement. Since these built-in sensors are subject to high sensor noise [Woo07], a large set of traces is necessary to cancel out the resulting drift errors [AY12]. The use of an external IMU using a foot-mounted strap-down system with drift correction (cf. Figure 8.1b) can reduce drift errors and, thus, perform well even with a small set of traces. However, this requires the deployment of additional hardware that is

rather expensive [Fox05]. Our system is agnostic of the actual inertial positioning system that is used for recording traces.

Even with the most accurate IMU, traces are still inaccurate. To deal with the remaining errors, we include trace correction steps as detailed in the next chapter. An initial trace correction step automatically preprocesses traces to deal with remaining drift errors, whereas a trace translation recovery step reduces the impact of translation errors of the initial position. Implementing and evaluating the system with more noisy sensors is part of future work.

5. Generating Indoor Maps from Odometry Trace Data

In this chapter we describe the *Trace-based Mapping* component. Its goal is to derive a floor plan for an observed floor of a building. Given an exterior outline of the building and a set T of (noisy) odometry traces collected by the trace data acquisition component, we compute a set of hallways (the *hallway skeleton*) and a set of rooms (the *room layout*) that together form the *trace-based indoor map*.

As depicted in Figure 5.1, detecting the hallway skeleton and detecting the room layout both are multi-step processes. As a preprocessing step to both, we first reduce the noise of direction changes in the traces in the *initial trace correction* step. For the detection of the hallway skeleton, in the *hallway detection step*, we extract *hallway segments* from all traces and group these segments per hallway. These segments define the walkable area of each hallway, which is converted into a 2D rectangle representation in the *hallway geometry estimation* step, forming the hallway skeleton. Using the hallway skeleton, we improve the translation error of traces in the *trace translation recovery* step.

To subsequently detect the room layout, we first reduce the noise of odometry traces further using the hallway skeleton in the *secondary trace correction* step. Next, we extract individual observations of rooms from traces in the *initial room detection* step by analyzing all steps that were taken inside the building interior but outside of a hallway. Finally, we find the actual room layout in the *room geometry estimation* step by clustering initial rooms. We present these steps in detail in the following subsections.

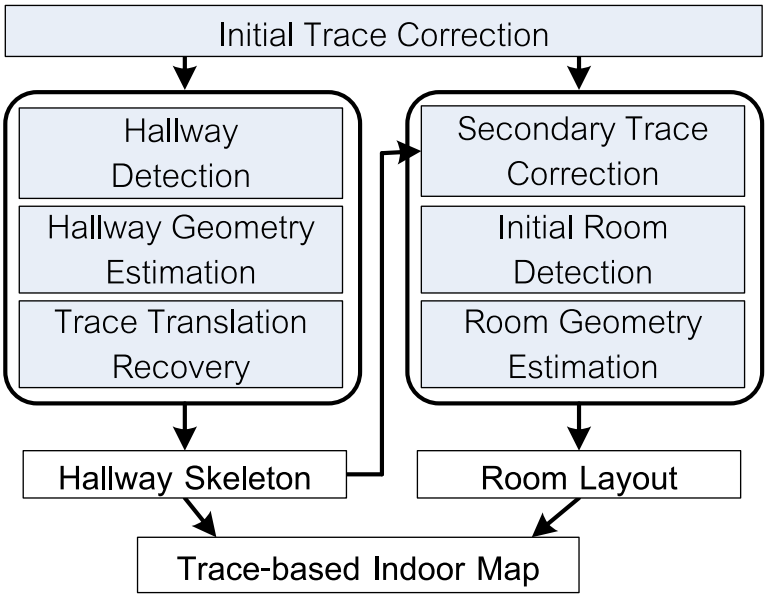


Figure 5.1.: Overview of the Trace-based Mapping component.

Algorithm 5.1 Initial Trace Correction

Require: Trace t, Exterior Outline E, *minLength*, τ
  $HS_t \leftarrow$ FINDHALLWAYSEGMENTS$(t, minLength, \tau)$
  for all $hs \in HS_t$ **do**
    $hs.chord \leftarrow (hs.start, hs.end)$
    $minAngle \leftarrow \min(\{\text{angle}(hs.chord, e) |$ Exterior Wall $e \in E\})$
    rotate$(t[p_s, p_{last}], minAngle)$
  end for

5.1. Initial Trace Correction

As detailed in Section 4.4, odometry traces recorded using an IMU contain errors. As an example, consider the trace depicted in Figure 5.3, taken using a foot-mounted IMU: Turning angles were recorded incorrectly, and due to sensor drift, the trace is slightly bent even though the user walked in a straight line. To reduce sensor noise, we leverage two observations: First, users commonly walk along the long axis of a hallway, and, secondly, hallways are commonly built parallel to an exterior wall of the building. Thus, given the exterior outline of the building—as can be readily obtained from, e.g., OpenStreetMap—we rotate straight segments of a trace parallel to an exterior wall according to Algorithm 5.1: We first extract a set of hallway segments HS_t from each trace t. Hallway segments are maximum-length sequences of steps with a minimum length of *minLength*, where the angle of

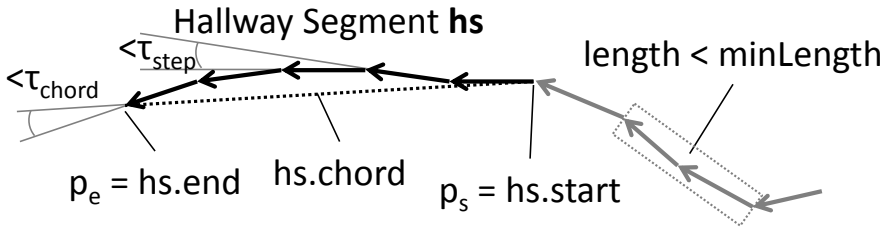


Figure 5.2.: Definition of Hallway Segments

any two subsequent steps is less than τ_{step} and the angle of any step to the chord of the segment is less than τ_{chord} (cf. Figure 5.2). The value of *minLength* must be chosen longer than the longest room that is presumably present in the building, so that no hallway is detected when walking along the length of such a room. Starting from the first hallway segment, for each hallway segment $hs \in HS_t$, we rotate the remainder of the trace including hs around $hs.start$ by the smallest possible angle so that $hs.chord$ is parallel to an exterior wall. Note that if a trace is extremely noisy, it may be rotated towards the wrong exterior wall. In this case, we say that a trace has been *corrupted*. Corrupted traces are detected and handled while deriving the hallway skeleton in the next steps.

5.2. Deriving the Hallway Skeleton

To derive the hallway skeleton, we begin by determining the set of hallways in the building and the topology of these hallways, i.e., finding the connections between hallways, by grouping the observed hallway segments in the *hallway detection* step. Intuitively, we would say that a hallway has been found wherever a hallway segment is detected on a trace. However, there are two problems with this simple criterion: First, if multiple users traveled through a hallway, that hallway is detected several times. Secondly, non-existing hallways are detected from corrupted traces. We solve these problems by detecting clusters of steps forming a hallway using the DBSCAN clustering algorithm [SEKX98]. We then apply a heuristic solution to identify which detected hallways are actual hallways and which are misdetections from corrupted traces. After finding the topology of hallways, we next compute their geometry in the *hallway geometry estimation* step from the walkable area observed in the hallway segments and the topological information found during hallway detection. Last, we use the *trace translation recovery* to

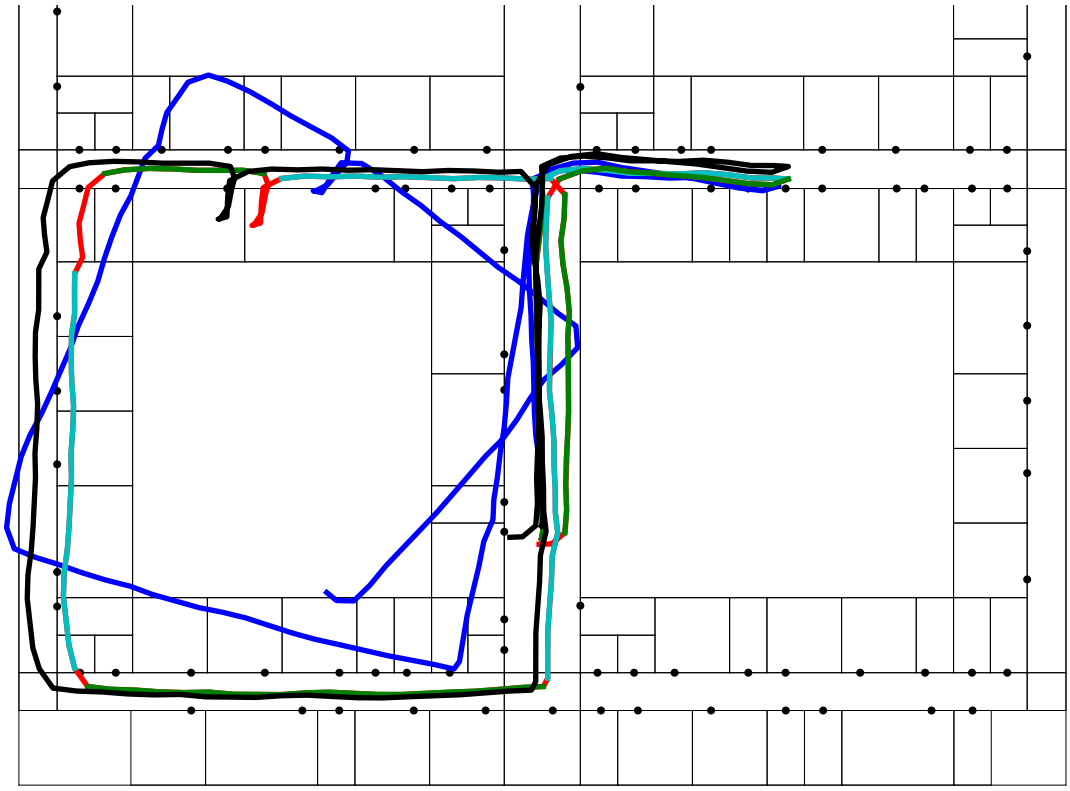


Figure 5.3.: Raw and corrected trace shown on the actual building layout. Original trace marked in blue. Trace after initial correction shown in red, with Hallway Segments marked in Cyan and Green. Trace after secondary trace correction shown in black.

attempt to recover information from traces that may have been erroneously classified as corrupted. In detail, the process works as follows.

Hallway Detection We first partition the set of observed hallway segments HS into subsets HS_{wall} according to which exterior wall they were rotated towards. For each HS_{wall} we extract all positions $\overrightarrow{p_k}$ from all segments in HS_{dir}. The coordinates of each position are then used as features for the DBSCAN clustering algorithm. DBSCAN assigns the steps to clusters based on their distance. Intuitively, a cluster is formed from positions $\overrightarrow{p_0}$ that have at least $minPts$ other positions $\overrightarrow{p_1}, ..., \overrightarrow{p_{minPts}}$ in their ϵ-Neighborhood, i.e., $\delta(\overrightarrow{p_0}, \overrightarrow{p_i}) \leq \epsilon, i \in \{1, \ldots, minPts\}$. This definition is transitive: if $\overrightarrow{p_0}$ forms a cluster P_0, $\overrightarrow{p_1}$ forms a cluster P_1 and $\overrightarrow{p_1} \in P_0$, then $P_1 \subseteq P_0$ and vice versa, i.e. Clusters P_0 and P_1 are joined. Thus, segments whose positions were assigned to the same cluster by DBSCAN are then considered to be observations of the same hallway.

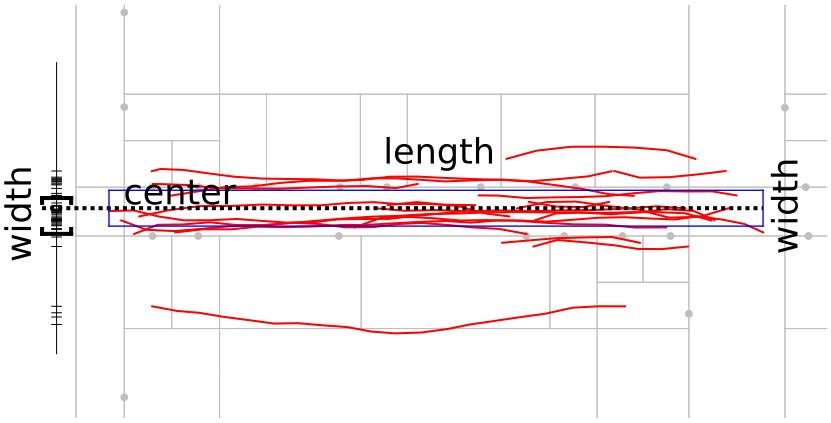
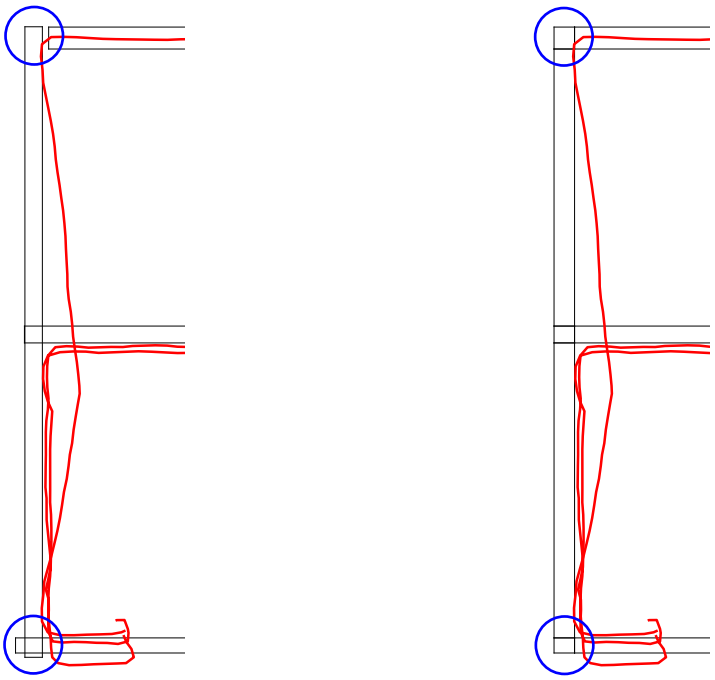


Figure 5.4.: Hallway geometry estimated from the bounding box, interquartile range and centerIndex. Actual building layout shown in grey.

To exclude false positives caused by corrupted traces, we require a minimum number of *minHallwaySegments* segments to contribute positions $\overrightarrow{p_j}$ to a cluster in order to accept the existence of a hallway. If there are less than *minHallwaySegments* segments, the traces contributing segments to this hallway are considered to be corrupted. Note that *minHallwaySegments* is different from the *minPts* parameter of DBSCAN. *minPts* regards positions only, i.e., multiple positions from a corrupted trace might still form a cluster. The additional use of *minHallwaySegments* requires the observations leading to the acceptance of a hallway to be independent.

Hallway Geometry Estimation To find the geometrical outline of each hallway, we begin by computing the *initial geometry* of each hallway. When the initial geometry is known, we extend it to the *actual geometry* by performing a topological extension and a topological contraction of all hallways (connecting topological neighbors), ensuring that the geometry accurately reflects the topological relationship of the hallways.

The *initial geometry* of each hallway is computed from three values, as depicted in Figure 5.4. The *length* (the dimension along the direction) is taken from the bounding box around all steps of all hallway segments. The *width* (the dimension orthogonal to the direction) is computed as the interquartile range of step positions, i.e., 25 % of steps are not included on either side of the hallway. This is motivated by the observation that traces exhibit a normal-distributed error orthogonal to the direction of a hallway. Furthermore, we observed that the position



(a) Initial geometry of hallways

(b) Actual geometry after topological expansion and contraction

Figure 5.5.: Topological expansion and contraction of hallways.

of hallway segments with respect to the true position of a hallway trends towards the side of entry. When users can enter a hallway from only one side, e.g., at the edge of a building, the center of the hallway is found too close to the center of the building. Thus, we compute the ratio of entrances into a hallway from either side and push the center (location of the hallway) towards the side with fewer entries. To this end, we set $centerIndex = (\frac{enter_{left} - enter_{right}}{2(enter_{left} + enter_{right})} + 0.5) \cdot number\ of\ steps$ and choose the *center line of the hallway* so that it passes through the *centerIndex*'th step from the left.

After finding the initial geometry of all hallways, we compute the *actual geometry* by performing the topological extension and topological contraction (see Figure 5.5). In the topological extension, we first consider the space left between hallways and exterior walls: The size of each hallway is increased so that no space of depth less than *minRoomDepth* is left between a hallway and an exterior wall, i.e., the space to place rooms in must have sufficient size to at least open the door. Next, we consider traces that pass from one hallway h_i to another hallway h_j. The rationale behind the topological extension is that when a trace t_k passes

from h_i to h_j, h_i and h_j must be connected. Therefore, we extend the length of h_i and h_j so that they intersect. This effect is illustrated in the topmost hallway depicted in Figure 5.5.

In the topological contraction, we again consider traces passing from h_i to h_j. The rationale behind this is that when a trace t_k passes from h_i to h_j, t_k should not cause h_i to be extended past the intersection with h_j. Due to the fact that the width of h_j was limited to the interquartile range of step positions, t_k may have overshot the derived intersection of h_i and h_j. Therefore, we recompute the length of h_i and h_j, considering the respective segments from t_k up to the point of intersection only. This effect is illustrated in the bottommost hallway depicted in Figure 5.5.

The set of all hallways with their actual geometry computed in this step forms the *hallway skeleton*.

Trace Translation Recovery In a last step we attempt to recover traces that were classified as corrupted during hallway detection. There are two reasons why traces are classified as corrupted: If they were rotated towards the wrong exterior wall during the initial trace correction or if they use an erroneous (translated) initial position $\overrightarrow{\pi}_{sens} \neq \overrightarrow{\pi}_{true}$. While in the first case the trace is lost, in the latter case, traces can easily be recovered given an existing hallway skeleton as follows. For each hallway segment hs_i of a trace classified as corrupted, we find the closest accepted hallway with the same orientation from the hallway skeleton. We then compute a translation vector $\overrightarrow{v_i}$ orthogonal to $hs_i.chord$ that would place hs_i onto the center line of the hallway. The trace is then translated by the average of all $\overrightarrow{v_i}$. We then recompute the hallway skeleton using the hallway detection and hallway geometry estimation as detailed before.

5.3. Finding the Room Layout

Using the hallway skeleton computed in the previous section, we now determine the room layout. As explained in the hallway geometry estimation step, in each hallway, traces exhibit a normally distributed error orthogonal to the direction of the hallway. Therefore, we add a *secondary trace correction* step, where we move the hallway segments of each trace onto the center line of their corresponding hallway (see Figure 5.3), analogous to the initial trace correction process shown

in Algorithm 5.1. Rooms then become visible as distinct protrusions of steps from hallways. Next, as an intermediate step, we perform an *initial room detection*, where we extract these observations of initial rooms from traces. In the subsequent *room geometry estimation* step, these initial rooms are merged using a Clustering algorithm based on a mixture of Gaussians to derive the *actual rooms* that were visited by participants.

Initial Room Detection We detect initial rooms by extracting *room segments* from traces. A room segment is a maximum-length sequence of contiguous steps in a trace that do not overlap with the hallway skeleton and are not part of a hallway segment which is, in turn, part of an accepted hallway.

However, there are other causes for observing such segments besides the observation of a room. First, when a participant takes a wide turn around a corner passing from one hallway into another, the apex of the turn may appear outside of the hallway skeleton even after the secondary trace correction. Second, when movement along a hallway was too erratic or the sensor drift was too high, some steps may not be detected as belonging to a hallway segment. In this case, parts of a trace may be "leaking" out of the side of the hallway. Fortunately, both of these cases are easy to detect. For every room segment, we find an *entrance door* and an *exit door*, placed at the location where the first and last step of the segment, respectively, cross the boundary of a hallway. In addition, we also identify the *entrance hallway* and *exit hallway* to which the room is connected by each respective door. We accept only room segments where the entrance hallway and exit hallway are identical (thus excluding wide turns) and where the entrance door and exit door are no more than 2 m apart (excluding leaking hallway segments).

The remaining room segments form the set of *initial rooms* (cf. Figure 5.7a). Each room segment is converted into an initial room by determining its outline and its door. Note that we limit our discussion to rectangular rooms, which are most common, and therefore simply use a bounding box parallel to the hallway for the outline. Support for rooms of a more complex shape may be added using, e.g., alpha shapes [AY12], if required. The door is placed at the location of the entrance door. The rationale behind this is that the accuracy of the trace degrades while the participant moves about in a room where our trace correction mechanisms cannot limit the sensor drift. Thus, the position of the entrance door is likely to be more accurate than that of the exit door.

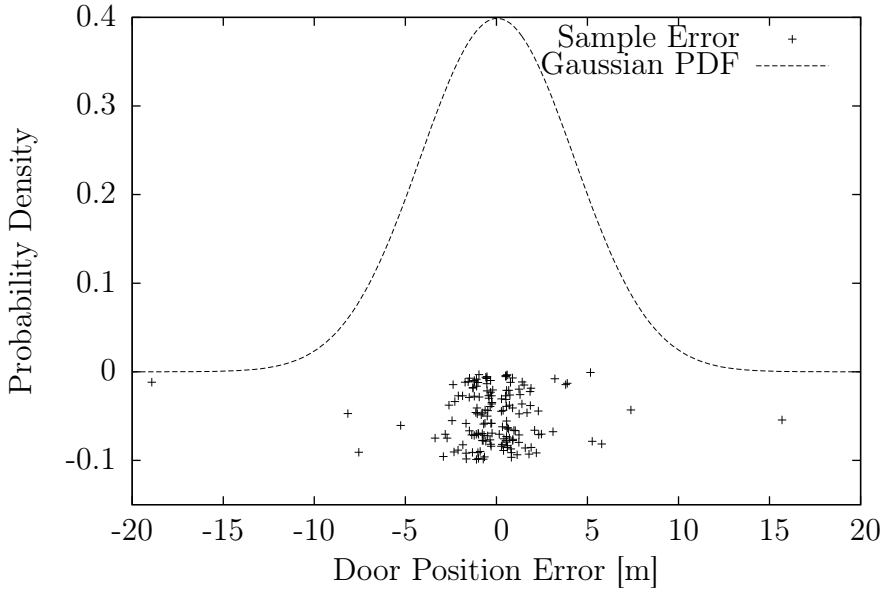


Figure 5.6.: Door Position Error for detected initial rooms.

Room Geometry Estimation Next, we determine the set of *actual rooms* visited by participants along with their geometry (outline and position) from the set of initial rooms. To this end, we first determine clusters of initial rooms, where each cluster corresponds to an actual room. We then compute the geometry of each actual room from the geometry of the initial rooms in its cluster.

In order to find a suitable clustering algorithm as well as a method to combine the geometries of clustered initial rooms, we conducted a preliminary analysis. Figure 5.6 shows the error in door position ("Sample Error"), i.e., the difference in position between an initial door (the door of an initial room) and the true position of the corresponding door. We found that the error in door position can be approximated by a Gaussian distribution with mean 0.04 m and standard deviation 4.22 m ("Gaussian PDF"). Thus, doors of initial rooms observed along a hallway can be viewed as samples from a Gaussian Mixture Model (GMM), where each mixture component corresponds to an actual room. Furthermore, the true position of a door can be found as the mean of all door positions of initial rooms observing this actual room.

Note that our GMM-based clustering algorithm makes use of the functionality for finding the geometry of an actual room from the geometry of a set of initial rooms. Therefore, we first solve the problem of finding the geometry of an actual

Algorithm 5.2 estimateRoomGeometry function to combine multiple initial rooms into a clustered room.

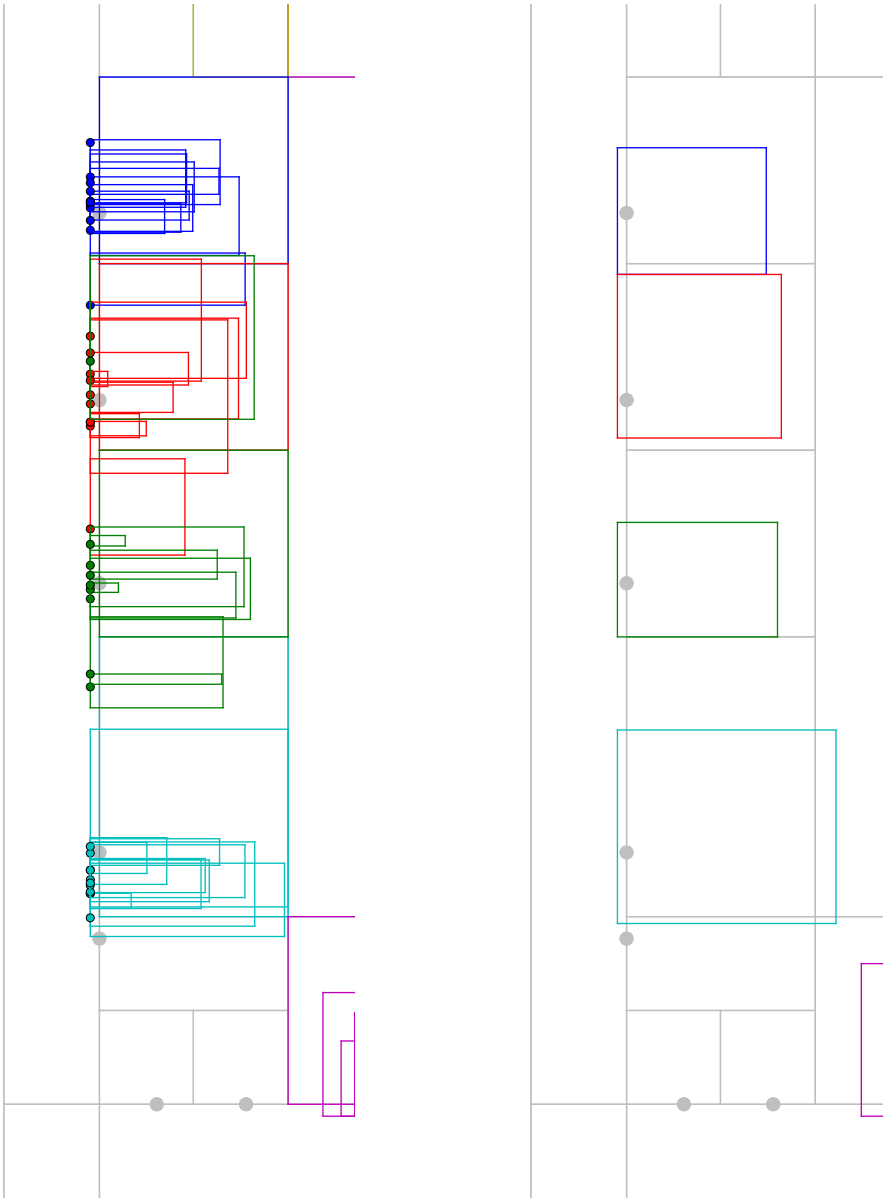
Require: Cluster c, initial rooms $r_i \in c.initialRooms$ with doors $r_i.door$

1: $c.door \leftarrow \text{mean}(\{r.door | r \in c.initialRooms\})$
2: $translatedRooms \leftarrow \{r + (c.door - r.door) | r \in c.initialRooms\}$
3: $c.outline \leftarrow \text{boundingBox}(translatedRooms)$
4: **return** c

room as follows. Given a cluster of initial rooms, we compute the geometry of the actual room using function estimateRoomGeometry (see Algorithm 5.2). estimateRoomGeometry sets the door of the resulting room *c.door* to the mean position of all doors in the cluster (line 1). To find the outline of the clustered room, a (temporary) set of translated rooms is computed (line 2). Translated rooms are the initial rooms in the cluster, each translated so that its door is located at *c.door*. The outline of the clustered room is set of the bounding box of all translated rooms (line 3). The rationale behind this is that first, as stated before, the mean of all initial door positions is a good estimate of the true door position. Second, all traces must have passed through the same door. Thus, the mean door position can be used to improve the accuracy of the observed initial rooms.

Given estimateRoomGeometry, we can now tackle the problem of finding the correct cluster of initial rooms for each actual room. As stated before, we view the doors of initial rooms as samples from a GMM. Determining the parameters of the underlying GMM requires two steps: First finding the number of mixture components in the model, equivalent to finding the number of rooms, and second finding the parameters (mean, variance) for each component. Initial rooms can then be clustered based on their membership to a mixture component to obtain the actual rooms.

When the number of components in the mixture model is known, the second problem can easily be solved through a maximum-likelihood estimation using a readily available EM-Algorithm [PVG+11]. The first problem, finding the correct number of components, is more challenging. A common solution is to use an information criterion, for example, by finding the number of components that minimizes the Bayesian Information Criterion (BIC) of the resulting mixture model. In practice, this requires fixing a plausible range for the number of components



(a) Detected initial rooms. Colors indicate the corresponding actual room.

(b) Actual rooms derived according to the Expand-and-Merge algorithm.

Figure 5.7.: Example of initial rooms, clustered according to the Expand-and-Merge algorithm. Actual building layout shown in grey. Dots represent doors.

to test, finding the parameters for each number of components in this range, and computing the BIC for each resulting mixture model. In the following, we denote this approach as function BICClustering*(initial rooms, minClusters)*, where *minClusters* denotes the lower bound on the number of components to test.

In our experiments we found that although BICClustering computes the correct set of rooms in some cases, most often the results are incorrect. On the one hand, BICClustering does not take into account that doors are usually observed close to their actual position, i.e., the variance of the initial doors for the same actual room is bounded. On the contrary, the use of the BIC leads the algorithm to use the smallest number of components that can fit the data. Thus, it often underestimates the true number of rooms, with the extreme case of merging all rooms along a hallway into one. On the other hand, when the lower bound for the number of components matches the actual number of components, i.e., underestimation is not possible, BICClustering tends to separate observations of the same room, i.e., finds more than the number of rooms that were actually observed. As with the position of hallway segments, the position of initial doors trends towards the side of entry (cf. Section 5.2). While the set of initial doors for a single room follows a Gaussian distribution, we often observe multiple distinct subsets of doors for each room, typically one to the left and one to the right of the true door position, while almost no observations are made at the true door position, leading to an overestimation of the number of rooms.

To address these issues, we developed our *Expand-and-Merge* algorithm, shown in Algorithm 5.3. Expand-and-Merge augments BICClustering using domain knowledge as identified above. In the *expansion phase*, we find a set of clusters that overestimates the true number of actual rooms, but where all initial rooms in a cluster are observations of the same actual room. In the *merge phase*, the resulting clusters are combined so that all observations for an actual room belong to the same cluster. In detail, Expand-and-Merge works as follows.

In the *expansion phase*, we iteratively increase the lower limit on the number of mixture components for BICClustering until it returns a set of clusters where the variance of the initial doors in each cluster is less than σ_{max}^{expand} (line 1–5). Once we have found a suitable lower limit, BICClustering will typically return too many clusters, e.g., identify observations for a room located to the left and a room to the right of an actual room.

Algorithm 5.3 Expand-And-Merge algorithm to find the set of actual rooms.

Require: Initial Rooms I, Variance Limits $\sigma_{max}^{expand}, \sigma_{max}^{merge}$
$minClusters \leftarrow 1$ ▷ Expansion Phase
2: **repeat**
$C \leftarrow$ BICClustering$(I, minClusters)$ ▷ Find BIC-minimal clusters
4: $minClusters \leftarrow minClusters + 1$
until $\forall c \in C$: variance$(c.initialDoors) \leq \sigma_{max}^{expand}$
6: $C \leftarrow \{$estimateRoomGeometry$(c) | c \in C\}$ ▷ Prepare Clustered Rooms
$C \leftarrow$ sort$(C, key = c.door)$ ▷ Merge Phase
8: **for** $i \in \{1, ..., |C|\}, C[i] \neq \perp$ **do**
for $j \in \{i + 1, ..., |C|\}$ **do**
10: $merged \leftarrow C[i] \cup C[j]$ ▷ Find Merge Candidates
$merged \leftarrow$ estimateRoomGeometry$(merged)$
12: **if** variance$(merged.initialDoors) \leq \min(\sigma_{max}^{merge}, merged.outline)$ **then**
$C[i] \leftarrow merged$
14: $C[j] \leftarrow \perp$
else
16: break
end if
18: **end for**
$i \leftarrow j$
20: **end for**
return $\{$estimateRoomGeometry$(c) | c \in C\}$

In the *merge phase*, we compensate for this by merging a cluster of initial rooms $C[i]$ with its neighbors. To this end, the clusters are first ordered by their mean door position along the hallway (line 7). Starting at one side of the hallway (line 8), we merge $C[i]$ with its neighbor $C[j]$ iff the variance of the door positions in the combined set of initial doors is less than σ_{max}^{merge}, limiting the overall distance of initial rooms being merged, and less than the width of the merged room, ensuring that several neighboring small rooms are not accidentally merged (line 12). If $C[i]$ and $C[j]$ have been merged, we continue to try and merge the next neighboring room $C[j + 1]$ into $C[i]$. Otherwise, $C[i]$ is complete and we restart the merging process with $C[j]$ and $C[j + 1]$ (line 19). Figure 5.7b shows an example of actual rooms derived by our algorithm.

Finally, the hallway skeleton together with the detected rooms forms the *trace-based indoor map*.

5.4. Limitations

Compared to previous approaches, MapGenie offers a number of improvements. However, its performance is limited in several areas, mainly the detection of short hallways and the clustering of very small rooms, as we will discuss in the following.

One limitation of MapGenie is that extremely short hallway segments are not detected, such as when a user turns into a hallway and then immediately enters the first room on this hallway. These (undetected) hallway segments can neither be used in any of our correction steps to improve the quality of the trace, nor do they contribute independent observations of a hallway. Furthermore, these undetected hallway segments may later be considered parts of a room, leading to the detection of initial rooms that are assigned to the wrong hallway and can subsequently not be used in finding the room layout. While the resulting erroneous initial rooms are eliminated by our criteria, we lose the information about the respective actual room contributed by these observations. In particular, we obtain only few usable observations for rooms at the end of a hallway.

An overall problem in deriving indoor floor plans from multiple independent observations is how to merge these observations. For rooms, a common approach is to rely on external features such as WiFi fingerprints to assign an observation of a room to an actual room [AY12] . The accuracy of WiFi fingerprints heavily depends on the environment, e.g., the number of access points and construction materials used. Thus, the resolution of such an approach can greatly vary. In our experiments, we found the resolution of WiFi fingerprinting to be insufficient to distinguish all but the largest rooms. Thus, we introduced our Expand-and-Merge algorithm, to enable correct clustering of smaller rooms. However, while Expand-and-Merge improves over the Fingerprinting approach, its resolution is still insufficient in some cases, i.e., initial rooms observed from a large number of very small actual rooms in close proximity may be incorrectly merged.

5.5. Summary

In this chapter we have adressed the problem of extracting an indoor floor plan for a building from a set of noisy odometry traces. First, our *initial trace correction* uses observations on architectural principles and typical user behavior to

improve the quality of the odometry traces. By rotating and translating each individual trace so that all of its hallway segments are aligned with an exterior wall, we obtain a set of odometry traces that is much less noisy and where hallway segments lie close to their actual position.

Next, we presented an approach for deriving the hallway skeleton. The DBSCAN clustering algorithm is used to form clusters of hallway segments that represent the same hallway. We then showed how to extract the geometry of each hallway from the hallway segments in its cluster by taking statistical properties of the observed hallway segments into account.

Finally, we detailed our *expand-and-merge algorithm* to retrieve the room layout of the building from the odometry traces without relying on external features such as WiFi fingerprints. Augmenting a standard algorithm for computing a Gaussian Mixture Model with domain knowledge allows expand-and-merge to improve the resolution of the room geometry estimation over similar approaches, i.e., the minimum size of rooms that can accurately be recovered is reduced.

6. Grammar-based Improvement of Indoor Maps

The information in the trace-based indoor map can be incomplete for areas where we do not have traces, or the map can be inaccurate due to the inaccuracies in the trace data (see Figure 6.3). We try to improve the quality of the trace-based indoor map in the *Grammar-based Mapping* component by filling white spots in the map and by trying to correct these inaccuracies by imposing constraints on valid indoor maps. For instance, in the trace-based map, rooms of the same type have slightly different geometries, and some walls might have been missed due to missing traces in a room. The Grammar-based Mapping will try to align walls such that these errors are corrected. To this end, we use architectural knowledge, encoded in a formal *room grammar* as developed by Becker et al. [BPF+13], [PBD+14].

In the following, we first present the formal definition of the grammar that encodes structural knowledge of the building. Subsequently, we define the *Layout Error*, a measure used to identify how well a grammar-based map fits the observations from the trace-based indoor map. Finally, we present our algorithm to derive a room layout with the help of the grammar. Note that for simplicity, we limit our discussion to rectangular rooms. The full grammar that supports arbitrarily shaped rooms is presented in [BPF+13].

	R_1^{room}	R_5^{unit}
Rule	$Space \rightarrow r_1\ Space$	$Space \rightarrow r_3 r_2 r_3\ Space$
Width	2.4 m	19.2 m
A-Priori	0.06	0.04
Type	Small Office	Two executives with assistant's office

Figure 6.1.: Example for room rules and room unit rules

6.1. Room Grammar

In general, the structure of a building can be separated in two different areas: Hallway spaces and non-hallway spaces. While hallway spaces are traversed by users to reach different rooms, non-hallway spaces contain the rooms which are ordered in a certain sequence along each non-hallway space. However, the size and relative ordering of these rooms is not created by random composition of walls, but follows architectural principles and semantic relationships. For instance, public buildings often feature a very limited set of room sizes. Furthermore, individual rooms may be grouped into superior room units by their semantic relationship, e.g., the office of an assistant is very likely next to the office of an executive. Thus, once an executive office has been detected from traces, the neighboring assistant's office is implicitly detected along with it.

Structural information for non-hallway spaces is encoded by a formal grammar for a regular language of the form $G = (N, T, P, S)$, where $N = \{Space\}$ is the set of non-terminal symbols, $T = \{\epsilon, r_a, r_b, r_c, ...\}$ is the set of terminal symbols, P is the set of production rules and $S = Space$ is the axiom. Each terminal $r_i \in T$ represents a class i of rooms, identified by their geometric extent (see Figure 6.1, 6.4). For instance, assistant's office and executive office are two different classes of rooms that can occur more than once in one floor plan. Furthermore, knowledge about room units is encoded as fixed sequences of rooms that can be produced. Therefore, the production rules P are defined as follows (cf. Figure 6.1):

- $R_i^{room} : Space \rightarrow r_i\, Space$ for $i \in$ room classes found in the building
- $R_n^{unit} : Space \rightarrow r_j ... r_k\, Space$ for $n \in$ room units found in the building
- $R^{\epsilon} : Space \rightarrow \epsilon$

For instance, one possible sequence of rules to fill a non-hallway space with this grammar is: $Space \rightarrow r_3\, Space \rightarrow r_3 r_1 r_5$.

While this grammar encodes knowledge about existing room classes and room units, it does not contain knowledge about the typical neighborhood of these elements. For example, a combination of assistant's office and executive office may typically occur in combination with other office rooms, but hardly ever in combination with maintenance access rooms. Therefore, two probability models are defined for the grammar: (1) The *a priori probabilities* of rules and (2)

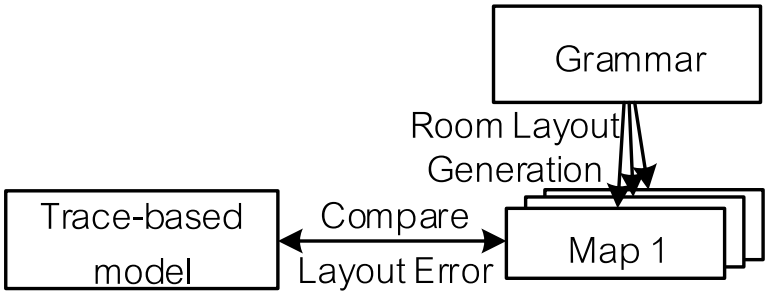


Figure 6.2.: Overview of the grammar-based room layout generation

the *relationship probabilities* between rules. The a priori probability $P_a(R_i)$ encodes the relative frequency of occurrence of a room or room unit defined by rule $R_i^{\{room,unit\}}$. The relationship probability $P_{rel}(R_i|R_j)$ is a conditional probability which models neighborhood relationships between rooms or room units. For instance, $P_{rel}(R_j^{room}|R_i^{room}) = 0.5$ states that with a probability of 50%, room $r_{next} = r_j$ in any sequence $...r_i r_{next}$.

To use these probabilities to generate a room layout, the grammar is translated into a Markov chain according to [BPF+13]. Room rules R_m^{room} and unit rules R_n^{unit} form the nodes of the Markov chain. The probability for a transition from R_i to R_j is defined as $\frac{P_a(R_j)}{P_a(R_i)} \cdot \frac{P_{rel}(R_i|R_j)}{P_{rel}(R_j|R_i)}$. We can then generate a room layout according to the grammar by performing a random walk on the resulting Markov chain.

6.2. Layout Error

The grammar presented in the previous section can create a multitude of different grammar-based indoor maps. However, we are interested only in the map that matches the observations from the trace-based indoor map. Intuitively, a grammar-based map matches a trace-based map when the following constraints are satisfied:

Free Constraint No wall may be placed inside a room observed from traces, i.e., observed rooms should not be cut in half.

Wall Constraint At least one wall must be placed in between any pair of neighboring rooms, i.e., observed rooms should not be merged.

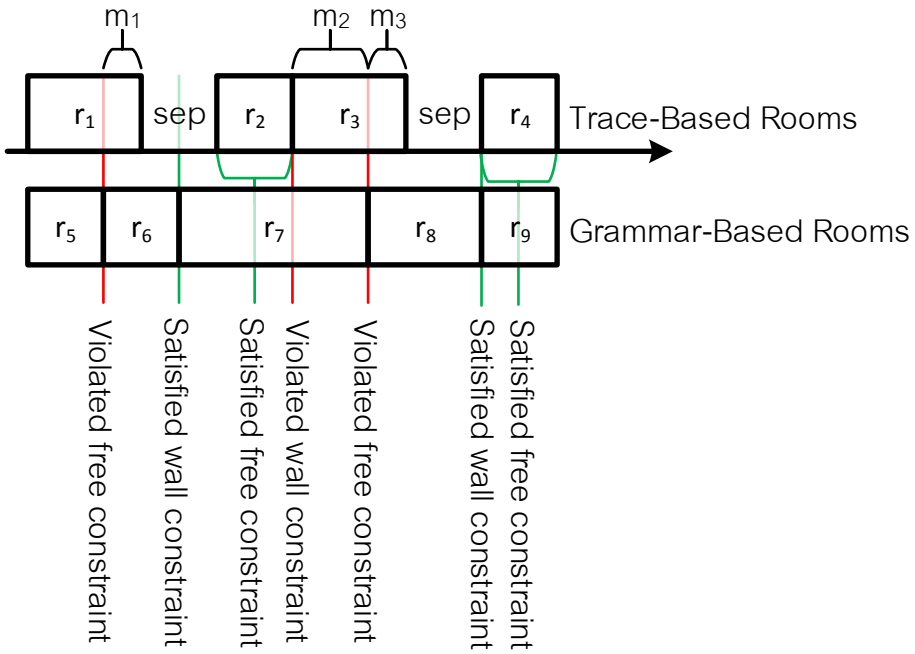


Figure 6.3.: Example identification of constraint satisfactions and constraint violations. r_1 to r_4 denote rooms observed from traces, r_5 to r_9 denote rooms generated by the grammar. m_i denote magnitudes of violations. *sep* denote separating spaces.

As observations may be inaccurate, a grammar-based indoor map that fully satisfies these constraints may not exist. Thus, we are more interested in identifying the grammar-based map that *best* matches the observations from the trace-based indoor map, i.e., violates these constraints the least (cf. Figure 6.2). To compare grammar-based indoor maps in terms of how well they satisfy the above constraints, we introduce the *layout error (LE)* of a grammar-based map:

$$LE = \frac{\left(\sum_{v \in violations} v^2\right)}{|violations| + satisfactions} \tag{6.1}$$

Here, *satisfactions* denotes the number of instances where a constraint was satisfied by the grammar-based map. *violations* is the set of *magnitudes of violation*. The magnitude of a violation is defined as the distance from the position where a wall was placed by the grammar to the position where it was expected in the trace-based map. In detail, these values are computed as follows (cf. Algorithm 6.1).

First, we iterate over individual rooms in the trace-based map. For each wall in the grammar-based map that intersects the current trace-based room, the

Algorithm 6.1 Algorithm for computing the layout error of a rule sequence

Require: Rule Sequence w, Trace-based Map TM

$satisfactions \leftarrow 0$
2: $violations \leftarrow \emptyset$
for all $room \in TM$ **do**
4: **if** w generates any wall in $room$ **then**
for all $wall$ generated by w, $wall$ intersects $room$ **do**
6: $magnitude \leftarrow \text{MIN}\left(\delta(wall, room.leftWall), \delta(wall, room.rightWall)\right)$
$violations \leftarrow violations \cup \{magnitude\}$
8: **end for**
else
10: $satisfactions \leftarrow satisfactions + 1$
end if
12: **end for**
for all $room_a, room_b \in TM$, $room_a$ is neighbor of $room_b$ **do**
14: $sep \leftarrow$ area in between $room_a$ and $room_b$
if w generates $wall$ in $sepSpace$ **then**
16: $satisfactions \leftarrow satisfactions + 1$
else
18: $magnitude \leftarrow \text{MIN}(\{\delta(w, sep) | wall \text{ generated by } w\})$
$violations \leftarrow violations \cup \{magnitude\}$
20: **end if**
end for
22: **return** $\left(\sum_{v \in violations} v^2\right) / \left(|violations| + satisfactions\right)$

free constraint is violated. We record a magnitude of violation as the minimum distance of the generated wall to either wall of the room (line 6). If no such wall exists, we count a constraint satisfaction for this trace-based room (line 10).

Next, we iterate over all pairs of neighboring rooms in the trace-based map. For each of these pairs we identify the *separating space*, i.e., the space in between these rooms (line 14). If a wall exists in the grammar-based map that is located inside the separating space, this wall satisfies the wall constraint and we count a constraint satisfaction for this pair of rooms (line 16). Otherwise, we record the minimum distance of any wall in the grammar-based map to the separating space as a magnitude of violation (line 18).

Finally, we compute and return the LE as defined in equation 6.1 (line 22).

An example of this process is depicted in Figure 6.3. For rooms r_1 and r_3 the free constraint is violated with magnitude m_1 and m_3, respectively. For rooms

Algorithm 6.2 Constraint-Augmented Random Walk

Require: Current Rule Sequence w, Set of Rules P, Trace-based Indoor Map TM

$P \leftarrow \{R \in P | wR$ fits in the available space$\}$ ▷ Annotate
for all $R \in P$ **do**
  $RE_R \leftarrow$ getError(wR, TM)
  $Q \leftarrow$ last rule in w
  $R.prob \leftarrow \frac{P(R)}{P(Q)} \cdot \frac{P(Q|R)}{P(R|Q)}$
end for
$P_{error} \leftarrow \{R \in P | RE_R \neq \perp\}$ ▷ Filter
$P_{prob} \leftarrow \{R \in P | RE_R = \perp\}$
$minRE \leftarrow$ min$(\{RE_R | R \in P_{error}\})$
$P_{error} \leftarrow \{R \in P_{error} | RE_R \leq minRE\}$
$P \leftarrow \{R \in P_{error} \cup P_{prob} | R.prob > 0\}$
if $P = \emptyset$ is empty **then** ▷ Recover
  $P \leftarrow P_{error}$
  for all $R \in P$ **do**
    $R.prob \leftarrow 1/|P|$
  end for
end if
normalize all $R.prob$ to 1.0 ▷ Select
if $P = \emptyset$ **then**
  return R^{ϵ}
else
  return $R \in P$ randomly selected according to $R.prob$
end if

r_2 and r_4 the free constraint is satisfied. For the pair of rooms (r_2, r_3) the wall constraint is violated with magnitude m_2. For all other pairs of neighboring rooms, the wall constraint is satisfied.

Note that the grammar can produce room layouts that do not completely fill (or may overflow) the available area. Avoiding such layouts is the responsibility of the room layout generation algorithm presented in the next section.

6.3. Room Layout Generation

The goal of the room layout generation algorithm is twofold: (1) Find a room layout with minimal layout error and (2) provide a probable room layout for unobserved areas that completely fills the non-hallway space. Finding the minimum-

error room layout from the grammar is an NP-hard problem that can be reduced to the Knapsack problem. Therefore, we employ a heuristic solution: We perform a *constraint-augmented random walk* on the Markov chain to find different room layouts (see Algorithm 6.2). The random walk is repeated *numRandomWalk* times and the room layout that minimizes the LE is chosen. Note that achieving a LE of 0 may not be possible: due to noisy observations, e.g., rooms detected at a short distance left or right of their actual position, it may not be possible to perfectly recreate the trace-based room layout from the grammar. To make our algorithm less sensitive to low noise, we define a tolerance threshold $\hat{T}_{LE}$ for all violations. Any magnitude of violation below the threshold is rounded to zero when computing the layout error. In our experiments, $\hat{T}_{LE} = 0.8\,\text{m}$ achieved the best results.

The algorithm iteratively performs the four steps of *Annotate*, *Filter*, *Recover* and *Select* as follows (cf. Algorithm 6.2).

Annotate: Initially, rules are annotated with their transition probability and the error they will cause. Given an (initially empty) sequence of rooms and the rules w that were used to produce these rooms, we first determine which rules can be applied without the resulting rooms overflowing the current non-hallway space (line 1). For each remaining rule R we compute a rule error (RE_R, line 3) and set its transition probability from the Markov chain (line 5). The rule error is computed analogously to the LE, limited to the area of the non-hallway-space that is affected by R. If no trace-based rooms are encountered in the affected area, e.g., when computing a layout in a part of the building that has not been observed at all, no rule error is set for R ($RE_R = \perp$, cf. Figure 6.1).

Filter: Next, rules are filtered for their applicability, i.e., whether they have a non-zero transition probability and minimize the resulting error. To this end, we split rules into two disjoint subsets: Error-Rules P_{error} (rules that have RE_R set, line 7) and Probability-Rules P_{prob} (all remaining rules, line 8). Error-Rules are filtered, keeping only the rules with the minimal rule error value (line 10). Note that we cannot limit the selection to either Error- or Probability-Rules, as illustrated by the example depicted in Figure 6.4. $r_1 \in P_{prob}$ as r_1 does not violate the free constraint for the trace-based room, whereas $r_3 \in P_{error}$. For instance, if we were to focus only on P_{error}, we would lose the chance to select the room sequence $r_1 r_3$ which would improve the LE over selecting r_3. Furthermore, the

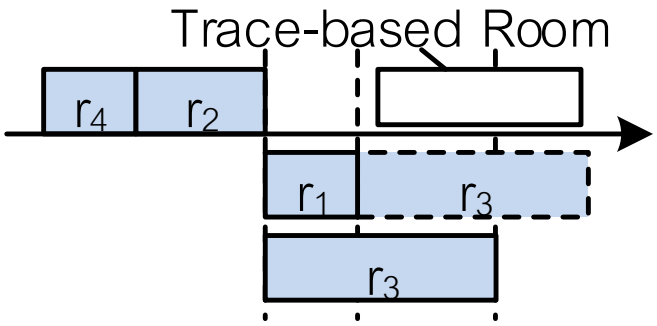


Figure 6.4.: Example application of rules. Combining Error- and Probability-Rules avoids unnecessary increases of the layout error.

grammar would be biased towards choosing rules that place more and larger rooms, as any rule in P_{error} generates rooms that occupy more space than those generated by any rule in P_{prob}. The reverse would also be true if we were to focus only on P_{prob}. Therefore, both sets are rejoined, removing all rules with a transition probability of 0 in the Markov chain, i.e., rules that cannot occur in this neighborhood (line 11).

Recover: The grammar can be semi-accurate for the current floor, e.g., productions may be missing, the grammar may contain additional room classes, or transition probabilities in the Markov Chain may be different. Semi-accurate grammars can occur any time a grammar is constructed from an indoor layout that is similar but non-identical to the current floor, e.g., partial observations of the same floor that have not yet encountered all room classes, or observations of another floor of the same building featuring the same room classes but different room units. When building a room layout using a semi-accurate grammar, we may encounter a situation where none of the remaining rules in either set has a transition probability > 0 in the Markov chain, effectively deadlocking our algorithm. In this case, we override the Markov chain by using the set of Error-Rules with minimal rule error value, temporarily adjusting their transition probabilities to a uniform value (lines 12–17).

Select: Finally, we choose a rule out of the remaining ones at random according to the (adjusted) transition probabilities (line 22). When the available space is insufficient to accommodate the rooms generated by any of the remaining rules, R^{ϵ} is applied and the resulting sequence of rooms is returned as the room layout for this non-hallway-space.

The hallway skeleton from the trace-based indoor map together with the set of grammar-based room layouts for all non-hallway-spaces forms the *grammar-based indoor map*, which is the final output of our system.

6.4. Summary

In this chapter, we addressed the problem of improving trace-based indoor maps using structural knowledge about the building. Structural knowledge can be obtained from indoor maps of floors similar to the current floor to be mapped, e.g., other floors of the same building. This information is then used in our *grammar-based mapping* approach to fill white spots and to correct inaccuracies in the trace-based map.

To this end, we first presented a formal grammar that encodes the structural knowledge, e.g., information about the types of rooms and about common groups of rooms. This grammar can be used to create a multitude of different grammar-based indoor maps that are similar in structure to the observed floor.

To identify which of these many grammar-based maps best resembles the current floor, we introduced the *layout error*. The layout error is a measure of how well a given grammar-based map matches the observations from the trace-based map. By selecting the grammar-based map with the minimum layout error, we find the map that best explains the current observations.

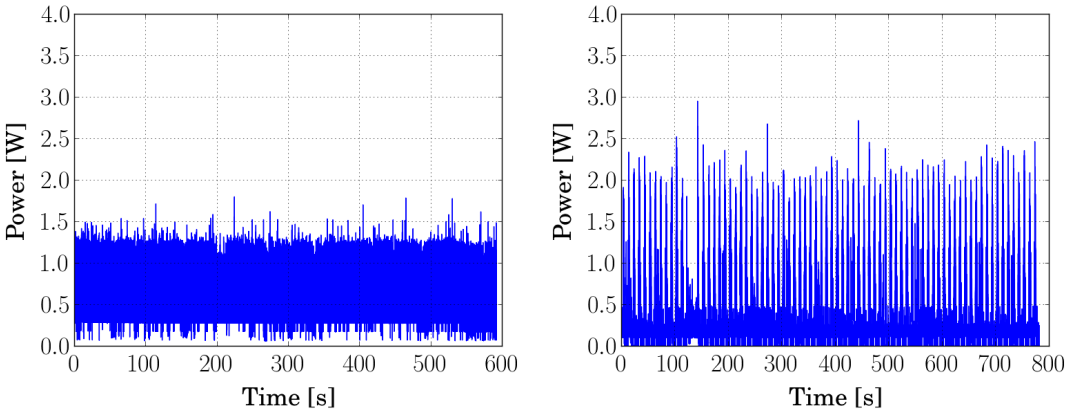
However, deriving all possible maps from the grammar is an NP-hard problem. Thus, the above approach is not feasible for practical applications. To alleviate this problem, we presented the *constraint-augmented random walk*, a heuristic solution to find the grammar-based map that best matches the observations.

7. Energy-Efficient Mapping

As argued previously, a major challenge in developing Public Sensing systems is limiting the energy drain on participating devices. The analysis of Baier et al. has shown that the biggest energy drain in MapGenie—as presented up to now—is recording a mobility trace via the Inertial Measurement Unit (IMU) [BPDR14]. Moreover, as participants will observe most objects several times and the amount of information gained decreases with each additional observation, the utility of running the IMU (and thus the energy spent) decreases over time. Therefore, we aim to focus the mapping effort on parts of the building where the currently available information is insufficient and to disable the IMU while traveling elsewhere.

In developing an approach for a more focused mapping effort, we face two challenges. The first challenge is to create a quality model that labels areas of the building as inaccurately mapped (IM) when we still need to record additional information, or accurately mapped (AM) when the available information is sufficient. The second challenge is to efficiently track the devices' position when the IMU is disabled, and to provide a new initial position for the IMU upon restarting the recording of a mobility trace. If we simply deactivate the IMU upon entering an AM-area, we sacrifice all information that a device may contribute in the future when it moves back to an IM-area.

To solve the second problem, we use a WiFi-based approach for position tracking. On the one hand, using WiFi for position tracking is more efficient than continuously using the IMU. Figure 7.1 compares the energy consumption of recording a mobility trace via the IMU and tracking the position through WiFi scans. We see that while the energy for running the IMU is less than the peak power consumption during WiFi scans, the device can enter a low-power idle state in between WiFi scans. This leads to an average energy consumption of 0.16 W for WiFi scans at a 10 s interval, compared to 0.45 W for running the



(a) Device runs IMU (b) Device performs WiFi scan every 10 s

Figure 7.1.: Energy consumption of a mobile device over time. Source: [BPDR14]

IMU. Therefore, we save energy by deactivating the IMU and switching to WiFi position tracking for as long as possible, while the device travels in an AM-area.

Note that the focus of this work is the Quality Model, as detailed in the next section. To demonstrate how the Quality Model is used to improve the energy efficiency of MapGenie, we will briefly sketch the remaining components as proposed by Baier et al. in the subsequent section. For a more detailed explanation of these components, we refer to the original work of Baier et al. [BPDR14].

7.1. Quality Model

The goal of our quality model is to label areas of the floor plan as accurately mapped (AM) or inaccurately mapped (IM), thus informing the scheduler whether additional traces should be recorded for an area. Intuitively, an area is AM if no more information can be gained from traces. Theoretically, this is the case when an area is completely filled with floor plan objects (hallways and rooms) with no empty space left in between. However, due to inaccessible spaces, e.g., maintenance access rooms with practically no visits or furniture blocking access to walls, an area may contain blank spots that cannot be covered by traces. From traces alone it is in general undecidable whether there is no information for an area due to missing traces (no user traveled there although it would have been possible) or due to an obstacle (no user can physically travel there). Therefore, it is impossible to define a perfect quality metric based on traces alone. Hence,

such a metric has to be based on heuristics. In the following, we define a generic quality metric applicable to all trace-based algorithms.

We define the quality of an area A based on a threshold $\hat{T}^{area}$ of the relative spatial coverage Q_A of A by accurately mapped objects (AM-objects). Whether an objects is accurately mapped (AM) or inaccurately mapped (IM) is defined by the number of observations, i.e., how many times the object was observed in the underlying trace data. More formally, our quality model is defined as follows. An individual floor plan object o is classified as AM-object if the number of observations $|o|$ is above a certain threshold: o is AM if $|o| \geq \hat{T}^{type(o)}, type(o) \in \{room, hall\}$. Otherwise, o is IM. Based in this definition, we define an area A of the floor plan as AM, if the relative spatial area of all AM-objects in A is above a certain threshold: A is AM if $Q_A = \left(\bigcup_{o \in A, o \text{ is AM}} geoArea(o)\right) / geoArea(A) \geq \hat{T}^{area}$. Otherwise, A is IM.

This metric ensures that we do not consider A to be an AM-area if there is a very large number of observations for a small part of A (e.g., few rooms with many visits) and few observations for the remainder. It also tolerates a certain fraction of inaccessible space per area (up to $\left(1 - \hat{T}^{area}\right) \cdot geoArea(A)$) without having to accept a lot of inaccurately mapped objects (IM-objects). The rationale behind this metric is based on the following observations about mapping approaches and real user mobility.

All mapping approaches based on traces require users to cover the complete space of an object to determine its correct size. As it is unlikely that any single user will cover a complete object with a single trace—e.g., covering the complete length *and* width of a hallway or room—we require multiple observations of each object. However, with an increasing number of observations already available, the chance to discover a new feature of the object diminishes. We therefore use threshold values $\hat{T}^{room}$ and $\hat{T}^{hall}$ to define whether a floor plan object is an AM-object or an IM-object. Note that we use different threshold values for $\hat{T}^{room}$ and $\hat{T}^{hall}$, as movement in hallways is typically less constrained than in rooms and thus fewer traces are sufficient to cover the full width of a hallway.

However, only basing the area quality metric on the relative spatial coverage by AM-objects will often lead to sub-optimal results due to the fact that large popular objects will dominate small and/or unpopular objects. In particular, hallways often cover large parts of an area and are visited by many users. Therefore, after

Algorithm 7.1 Algorithm for computing the quality of an area A. *geoArea*() defines the area covered by an object, $|o|$ the number of observations of o. h_i denotes a hallway, and r_i denotes a room.

1: **if** $\exists h_i \in A : |h_i| < \hat{T}^{hall} \vee$ Observation of h_i not finished **then**
2: **return** IM-area ▷ Phase 1: Detect Hallways
3: **else**
4: $Q_A \leftarrow \frac{\sum\{geoArea(r_i) || r_i| \geq \hat{T}^{room}\}}{geoArea(A) - \sum\{geoArea(h_j)\}}$ ▷ Phase 2: Detect Rooms
5: **if** $Q_A \geq \hat{T}^{area}$ **then**
6: **return** AM-area
7: **else**
8: **return** IM-area
9: **end if**
10: **end if**

a large popular hallway has been accurately mapped—which due to its popularity typically happens quickly—, no more rooms would be detected since the area is well-covered by AM-objects. To avoid this problem, we use a two-phase approach as shown in Algorithm 7.1: First, we ensure in Phase 1 that all hallways have been detected (lines 1–2). Then in Phase 2 we focus on the mapping of rooms only (lines 4 ff.).

Note that as mentioned above it is undecidable from traces alone whether we have detected all objects (including all hallways). However, for the specific case of hallways, topological constraints typically apply that make the problem decidable for hallways. Hallways are typically connected to other objects (further hallways or rooms) or an outer wall on both sides. Thus, as long as there is empty space at either side of the hallway (we say that the hallway observation is *not finished*), it is not accurately mapped since its coverage is still unknown, and the hallway is marked as IM-area (line 1). If we have detected both ends of a hallway and it received enough traces, it is considered to be accurately mapped. If every hallway is accurately mapped, we start Phase 2 and focus from there on the mapping of rooms based on the relative coverage of the area by AM-objects (line 4). Since at this point we know that hallways have been accurately mapped already, we subtract their area from the total area under consideration.

Note that all threshold values $\hat{T}^{\{room,hall,area\}}$ are parameters of the quality model that need to be set by an operator. $\hat{T}^{hall}$ and $\hat{T}^{room}$ must be adjusted depending on the floor plan generation component, e.g., to account for the vary-

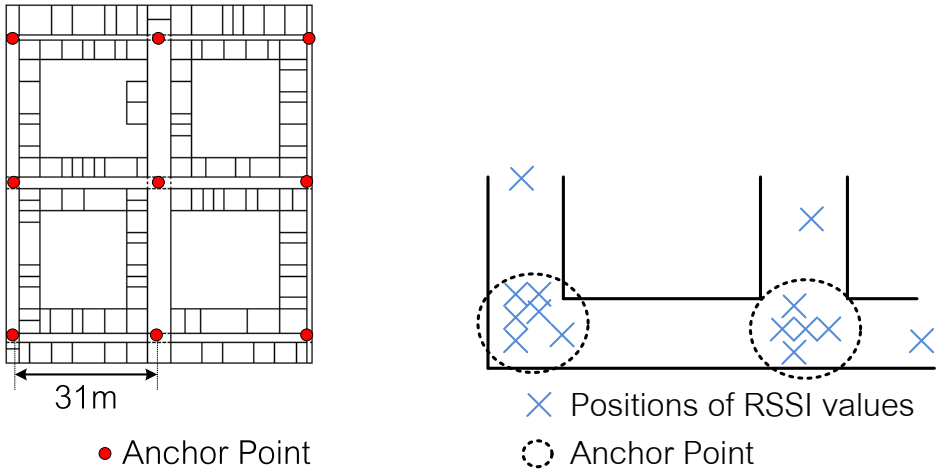
ing degree of accuracy of different mapping algorithms. $\hat{T}^{area}$ can be adjusted according to the desired level of accuracy of the indoor map and thus defines an energy/accuracy-tradeoff. $\hat{T}^{area} = 1.0$ would only classify areas as AM-areas, when every detail about them is known. From our evaluations, we could conclude that choosing $\hat{T}^{area} \leq 0.4$ allows significant energy savings without compromising on the quality of the derived floor plan.

7.2. Energy-Efficient Mapping

The quality model presented in the previous section can be used to improve the energy efficiency of the mapping process by disabling the IMU while the device is in an AM-area, and reenabling it when the device enters an IM-area. However, to determine whether the device is about to leave the AM-area and enter an IM-area, we must continuously track the (rough) position of the device, even when the IMU is disabled. This can be solved by using a coarse-grained WiFi Position Tracking system (WPT), based on RSSI fingerprinting.

In overview, the energy-efficient mapping process works as follows. Whenever there is an update to the indoor map, the *Area Task Creation* component verifies the current set of tasks. For example, when a new hallway is discovered, a new task t_2 is defined for its area $geoArea(t_2)$. The task that previously covered the area of the new hallway, say t_1, is modified to exclude the area of t_2: $geoArea(t_1) := geoArea(t_1) \setminus geoArea(t_2)$. Next, the *Quality Model* recomputes the IM/AM classification for all areas. On any change in the set of tasks or the quality classification, the respective updates are transmitted to participating devices. On each device, the *Scheduler* component tracks the current position of the device as provided by the IMU or WPT, respectively. Depending on the quality information about the current area and the past mobility, the Scheduler enables or disables the IMU, and thus the Trace Data Acquisiton component along with it.

Next, we briefly look at the operation of the WiFi Position Tracking system, before we present the decision-making process for the scheduler in subsequent sections.



(a) Placement of WiFi anchor points. (b) Anchor points derived from positions where RSSI values were recorded.

Figure 7.2.: Building the WiFi anchor point database. Source: [BPDR14]

7.2.1. WiFi Position Tracking

The task of the WiFi Position Tracking component is to provide a coarse-grained position. This is achieved by defining a set of anchor points at well-known positions and recording their RSSI fingerprint. A device can then at any time record the current RSSI fingerprint and find the anchor point with the best-matching fingerprint. The WPT then returns the position of the anchor point as the devices' current position.

The placement of the anchor points is subject to application requirements. In our system, these are as follows. First, the Scheduler uses the anchor points to track the sequence of AM-areas visited by the device while the IMU is disabled, i.e., the density of anchor points must be sufficient to always determine the current area. Second, the IMU uses the WPT to find an initial position when it is reenabled. Simply using the position of the closest anchor point leads to huge positioning errors in the order of magnitude of the distance between anchor points. Thus, anchor points must be placed so that a device can detect when it is at the actual position of the anchor point.

Based on these requirements and our definition of areas (cf. Section 4.3), we place anchor points at each hallway intersection as shown in Figure 7.2a. Intersections can be detected by the IMU whenever a device takes a sharp turn

(a) WiFi scan taken at 90° turn

(b) Trace located in the building according to WiFi fingerprint

Figure 7.3.: Finding an absolute position for a trace without an initial position using WPT. Source: [BPDR14]

(around 90°, cf. Figure 7.3). Furthermore, these anchor points can be set up on-the-fly when initial information about the hallway skeleton is recorded. When the IMU is recording a trace with a known initial position, the device performs a WiFi scan at every such turn. The information about the recorded RSSI values and the associated position within the building is then uploaded to the gateway. The gateway combines this information based on the RSSI fingerprint and fixes the position of each anchor point at the average position of the measurements (cf. Figure 7.2b). This database of fingerprints is transmitted to all devices along with task updates.

Note that when a device crosses an intersection without turning, no anchor point is detected. Thus, in this case, the IMU can not be reenabled. This is a known limitation of the WPT that is handled by the scheduler presented in the next section.

7.2.2. Scheduler

The *Scheduler* uses the information provided by the WPT to decide whether the IMU should record a trace or be disabled. As we do not want to miss any information about an IM-area, the IMU should already be recording a trace when entering an IM-area. To reenable the IMU in a timely manner, we use a mobility model to predict whether the device will enter an IM-area in the near future. In the following section, we detail this mobility model before presenting the decision algorithm to enable or disable the IMU and the algorithm to determine the initial position for the IMU upon reenablement.

Mobility Model

Given the current user trace as input, the mobility model outputs the probabilities of possible continuations of the trace. From these probabilistic trajectories, we then determine the probability $P(a|t)$ that the device, having traveled along trace t, will visit a certain area a. Based on $P(a|t)$, the scheduler can then decide whether to stop or resume IMU positioning.

Note that we make no further assumptions on the capabilities of the indoor mobility model. Designing a sophisticated indoor mobility model is an open research question beyond the scope of this work. For the purpose of this work, we define a simple mobility model based on the (reasonable) assumption that users take the shortest path towards their destination. Obviously, more complex models taking, for instance, user habits into account, can lead to even more accurate predictions.

Given the path t that the user has previously traveled, we first compute the set A of reachable areas. An area a is reachable from trace t if t is a prefix of a shortest path from the origin of t to a (note that depending on the building layout, multiple shortest paths may exist). Next, we assign visiting probabilities to all $a_i \in A$. In our simple model there is no information whether for any two destinations $a_i, a_j \in A$ one is more likely than the other. Hence, we assign a uniform visiting probability $P(a_i|t) = |A|^{-1}$ to each area $a_i \in A$.

IMU decision

Based on the information from the quality model and the set of reachable areas, we now present the decision algorithm to enable or disable the IMU. To disable the IMU, two conditions must be fulfilled:

1. The device is currently located in an AM-area.
2. For every path leading to an IM-area, IMU positioning can be resumed before entering that area.

While the first condition is easy to evaluate, the second condition is more challenging. To reenable the IMU in a timely manner, a 90° turn (indicating an anchor point) must be detected before entering any IM-area. Therefore, we first determine the next intersection X the device will pass. Next, we partition

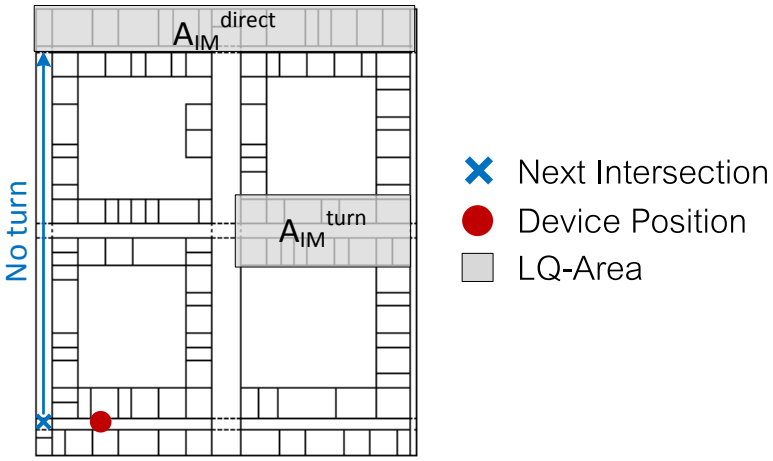


Figure 7.4.: Only the IM-area at the top can be reached without a turn from the next intersection the device passes.

the set of reachable areas A into disjoint subsets A_{IM}^{direct}, i.e., IM-areas that are reachable from X without taking a turn (cf. Figure 7.4), and A_{IM}^{turn}, i.e., IM-areas that are only reachable from X via at least one turn. As condition 2 holds for all $a \in A_{IM}^{turn}$, our decision is based on A_{in}^{direct}. We therefore compute the probability p_{IM} for the device moving into any area $a \in A_{IM}^{direct}$ as

$$p_{IM} = 1 - \prod_{\forall a \in A_{IM}^{direct}} 1 - P(a|t)$$

Using a random sample s_r from a uniform distribution, the IMU is disabled iff $s_r > p_{IM}$ and (re-)enabled otherwise. Note that this probabilistic decision also ensures that every now and then IMU traces are recorded for areas that are already sufficiently mapped. This gives our system a chance to discover features that are observed by only very few traces, i.e., rooms that are rarely ever entered. Furthermore, it provides a way to notice changes in the floor plan, i.e., when an obstacle such as a desk has been moved.

IMU recovery

To perform the position recovery, the device enables its IMU and takes a trace without a known initial position. When a sharp turn is discovered in the trace, it performs a WiFi scan to determine the current anchor. The trace is fixed to this anchor by setting its initial position $\overrightarrow{\pi} = \overrightarrow{\pi}_{anchor} - \sum_{i=1}^{k} s_i$, where $\overrightarrow{\pi}_{anchor}$ is

the position as reported by the WPT, and s_k is the step at which the WiFi scan was performed.

7.3. Summary

In this chapter, we addressed the challenge of improving the energy efficiency of the indoor map construction process. To this end, we first detailed a *quality model* that classifies areas of the indoor map as accurately mapped (AM-area) or inaccurately mapped (IM-area), depending on whether additional information should be recorded for this area.

We then showed how our *Scheduler* component exploits this information by focusing the mapping effort on IM-areas. While the device is located in an IM-area, the Scheduler uses the inertial measurement unit (IMU) to record an odometry trace. When the device moves to an AM-area, the Scheduler can disable the IMU and switch to a more efficient but less accurate *WiFi position tracking system* (WPT). While the position accuracy of the WPT is insufficient to provide an odometry trace, it is sufficient to determine when a device moves into another IM-area. Thus, the Scheduler can reenable the IMU in time to record information about the next IM-area.

8. Evaluation

To demonstrate the effectiveness and energy efficiency of MapGenie, we tested our system in a real-world scenario. In the following section, we present our setup for gathering real-world trace data. Next, we analyze the effectiveness of the grammar-based approach for the improvement of indoor maps. We first describe the general setup of our evaluation, before we discuss evaluation results for *sparse* and *dense scenarios*, showing the performance of our system during trace acquisition and when a full set of trace data for the building is available. Finally, we discuss methodology and results for the evaluation of our energy-efficient mapping approach.

8.1. Real-World Trace Data

To set up the inertial positioning system, we designed an Android App, communicating with a foot-mounted Inertial Measurement Unit (IMU) via Bluetooth (see Figure 8.1). We used a Zero-Velocity-Update protocol [Fox05] to limit the sensor drift of the foot-mounted unit. All other components are implemented as

(a) Android App (b) Foot-mounted IMU

Figure 8.1.: Utilities used for recording the pedestrian traces: A foot-mounted IMU that sends relative position changes to an Android App which merges these steps to a trace and shows it on a map.

Parameter	Description	Value
$minLength$	minimum length of hallway segments	4 m
$minHallwaySegments$	minimum number of independent observations for hallways	10
ϵ	neighborhood size for DBSCAN	2 m
$minPts$	minimum neighborhood population for DBSCAN	4
σ_{max}^{merge}	maximum variance of initial doors for clustering rooms	5
$minRoomDepth$	minimum depth of space for rooms	2 m
$\hat{T}_{LE}$	lower threshold for layout error	0.8 m
$numRandomWalk$	repetitions of grammar-based map generation	10
$\hat{T}^{hall}$	minimum number of segments for AM classification of hallways	10
$\hat{T}^{room}$	minimum number of segments for AM classification of rooms	5
$\hat{T}^{area}$	minimum area covered for AM classification of areas	11 %, 18 %

Table 8.1.: Parameters used in the evaluation

a backend-service on a Linux server (cf. Section 4.1). The parameters used in our evaluation are shown in Table 8.1.

To test our implementation, we collected 250 odometry traces, measuring a total of over 22 km from four volunteers. Traces are collected from the 2nd floor of the computer science building in Stuttgart (see Figure 8.5). Volunteers were given a list of trips to complete, containing both *visitor trips* and *employee trips*. Visitor trips specify an entrance to use and a destination office to search for, without further instructions on how to get there. These trips mimic the random travel of people unfamiliar with the building layout, searching for a room. In contrast, employee trips specified a sequence of locations (offices, meeting rooms, coffee machine, printer room, etc.) and the exact path to each destination, mimicking the everyday behavior of people working in the building to be mapped. Note that this includes trips of regular office staff as well as cleaners and security guards. Apart from these instructions, volunteers could freely walk all hallways in the building. Room access was limited to only a subset of office rooms, mainly located in the top left quadrant. Thus, hallways in this quadrant are well traveled, whereas only very few traces passed through hallways at the right and lower edges of the building. Furthermore, users did not visit maintenance rooms,

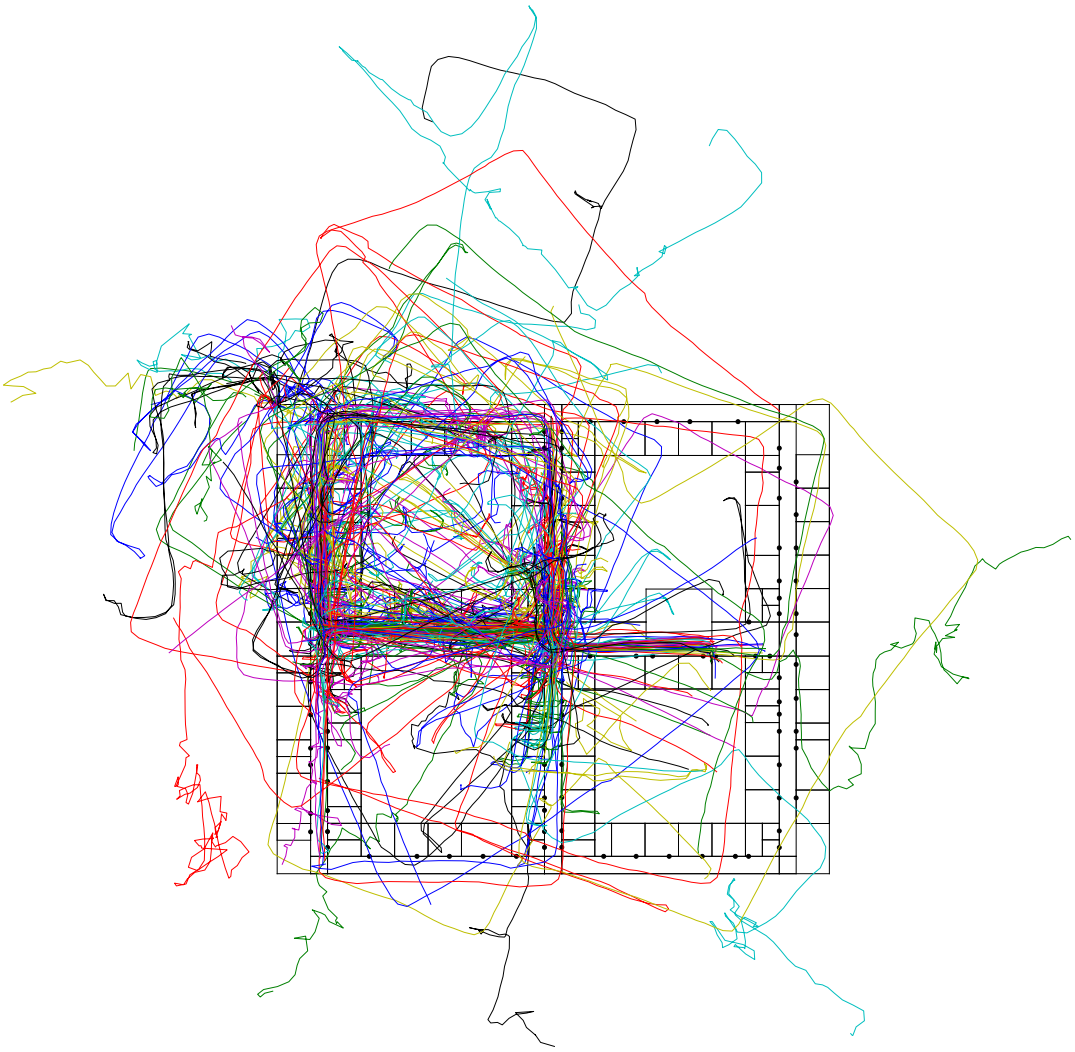


Figure 8.2.: Real-World Traces used in the Evaluation, shown on the ground truth map.

which are not publicly accessible. In total, volunteers visited 21 hallways (i.e., all hallways) and 26 rooms. Unfortunately, there are several traces in which no hallway segments were found. Thus, none of our correction steps can be used to reduce the noise in these traces. This affected mainly the trips of cleaners, who do not travel along a hallway but enter into every room. However, as detecting these traces is trivial, we configured MapGenie to exclude such traces, leaving 147 traces to be used in our evaluation. Figure 8.2 shows the complete set of raw traces used in our evaluations. The data set is available from the ComNSense Project Website [com14].

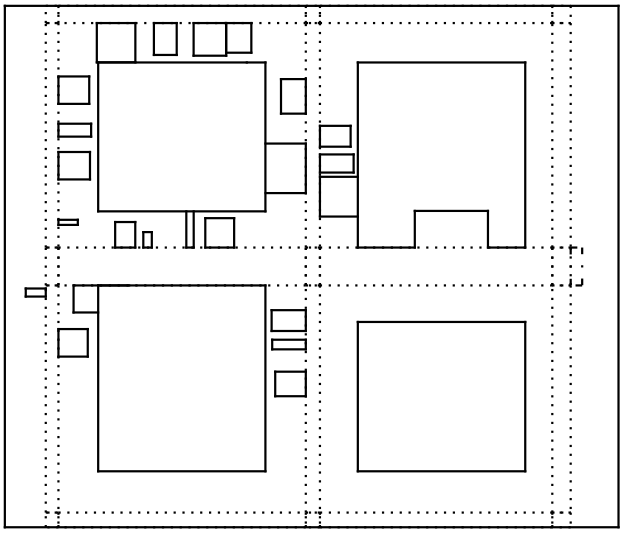


Figure 8.3.: Trace-based Indoor Map

8.2. Effectiveness of the Grammar-based Improvement of Indoor Maps

We evaluate our grammar-based approach for the improvement of indoor maps in two scenarios. In the *dense trace scenario*, we show the output of the system when using all available information, i.e., we derive the indoor map using all traces and using an accurate grammar for the floor. This demonstrates how even a well-mapped area can benefit from using a grammar-supported approach. Furthermore, we demonstrate the performance of our algorithm for clustering initial rooms using Gaussian Mixture Models. Later evaluations use perfect external knowledge for the clustering of rooms, i.e., have a-priori knowledge, which actual room was observed when an initial room is found. This allows us to abstract from the limitations of this algorithm, focusing on the individual benefit of the respective approach under evaluation.

In the *sparse trace scenario*, we analyze the performance of the system with incomplete information, i.e., using only a fraction of the traces, showing how the grammar-based approach is used to enhance the incomplete information of these traces. Furthermore, in the sparse scenario we use two grammars for the room layout generation: An *accurate grammar* derived from the floor plan of the 2nd floor and a *semi-accurate grammar* derived from the 1st floor of the same building,

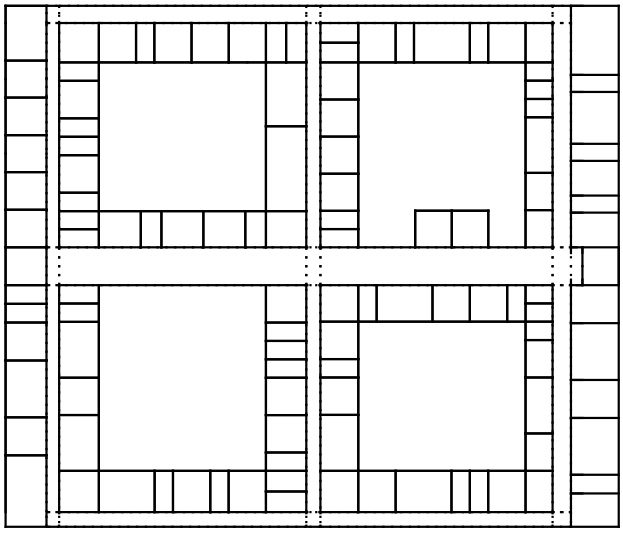


Figure 8.4.: Grammar-based Indoor Map

which, while overall similar to the 2nd floor, features different room types, room units and neighborhood relationships. Neither grammar includes knowledge of maintenance access rooms, which we cannot find without having an accurate floor plan in the first place. The semi-accurate grammar illustrates the performance of our system when using data obtained from a similar building. Note that using a fully inaccurate grammar will lead to fully inaccurate floor plans.

In the following, we first present the results for the dense trace scenario, followed by the results for the sparse trace scenario in the next subsection.

8.2.1. Dense Trace Scenario

Figure 8.3 shows the trace-based indoor map from the dense trace scenario, i.e., using all traces. Hallways, on the one hand, were estimated almost perfectly in areas covered by a large number of traces, i.e., the top-left quadrant and the center hallways. The rightmost and bottommost hallways are derived from only four traces each, which do not contain a sufficient number of samples to accurately find their width and position. The room layout, on the other hand, is very poor. Out of a total of 74 office rooms in the building, only 23 were found. Rooms are only of the right size if they are either very narrow or were detected from multiple (in our case at least seven) traces.

The remaining rooms, that were visited by volunteers but not found, are miss-

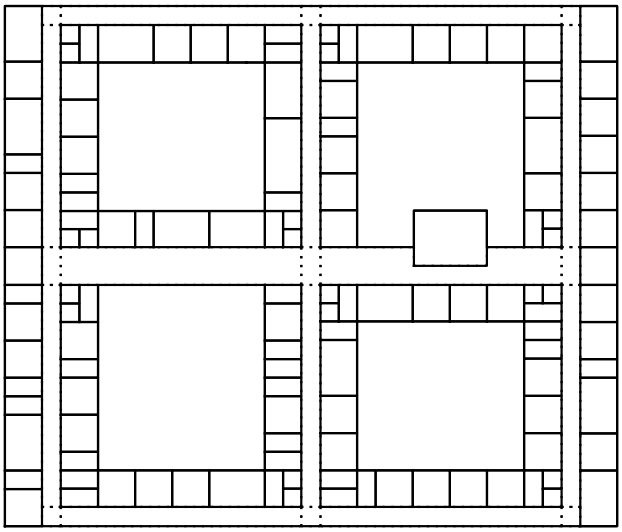


Figure 8.5.: Ground-Truth Indoor Map

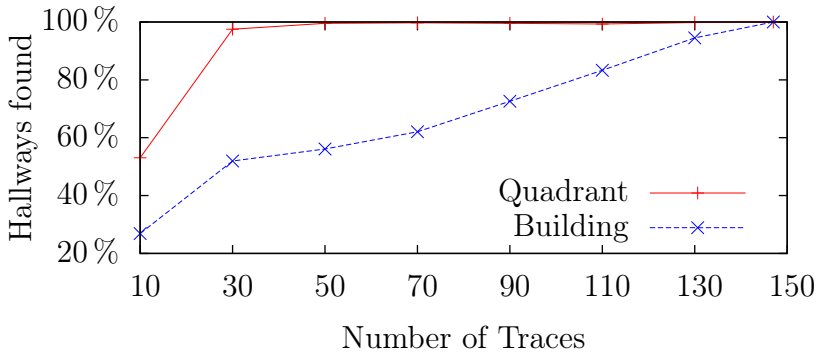
ing due to one of two reasons. In some cases, very short hallway segments have not been detected as such and were thus wrongly included in an initial room. The resulting initial rooms were discarded, leaving the system with too few observations to derive actual rooms. In other cases, multiple very small rooms have been merged into one.

Next, we look at the grammar-based indoor map, depicted in Figure 8.4. When comparing the grammar-generated room layout to the trace-based indoor map and the ground truth, we see that in areas where no rooms were found, the grammar provides a plausible but not necessarily perfectly accurate layout of rooms. In areas with observations, the observed rooms are reproduced by the grammar. Overall, the basic structure of the building is reflected in the map.

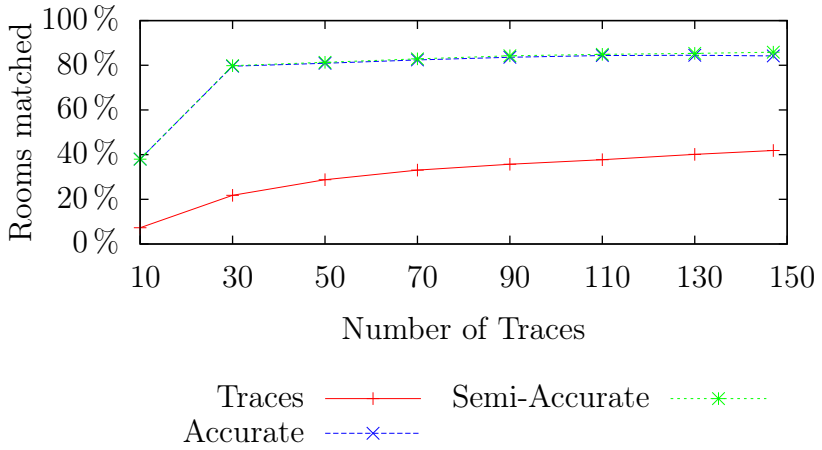
8.2.2. Sparse Trace Scenario

In the sparse trace scenario, we analyze the performance of our system when operating on a limited set of traces only. For varying numbers of traces n, we repeatedly selected 1000 random subsets of n traces out of the set of all traces. For each set of traces we construct the trace-based indoor map and 10 grammar-based indoor maps using either grammar.

We first compare the hallway skeleton from the computed indoor map to the ground truth hallway skeleton. Figure 8.6a shows the average number of hallways



(a) Detected Hallways

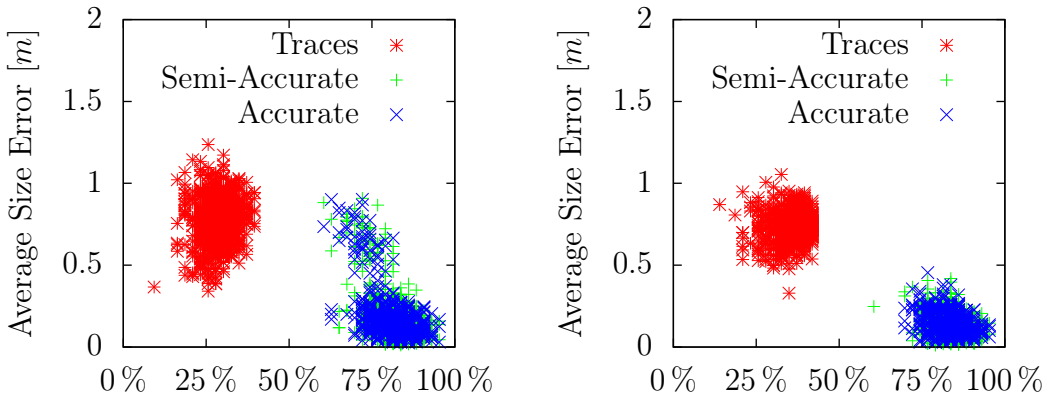


(b) Matched Rooms

Figure 8.6.: Average fraction of floor plan objects over the number of traces used.

detected for each number of traces. Using 30 traces, we can find 48 % of hallways in the building ("Building"). Finding more than 75 % of hallways requires at least 110 traces. This is due to our requirement of a minimum number of independent observations, before a hallway is accepted. While a smaller number of traces is sufficient to observe all hallways, requiring fewer observation introduces false-positives, i.e., non-existing hallways are detected. When limiting the analysis to the top left quadrant ("Quadrant"), results are much improved. On average, we find half of the hallways in this quadrant with only 10 traces, and all hallways when using at least 30 traces. This is due to the concentration of traces in that area, i.e., we are more likely to find hallways here than in any other part of the building.

Next, we analyze the accuracy of the room layout generation. For each com-



(a) 50 traces (b) 90 traces

Figure 8.7.: Fraction of matched rooms vs. average error in room size. 700 samples from each experiment shown.

puted indoor map, we match detected rooms to rooms from the ground truth map. Matching rooms lie on the same side of the same hallway and have similar size. Furthermore, we respect the ordering of rooms, i.e., if room b is located to the right of room a in the computed map and a was matched to x from the ground truth, then b cannot be matched to a room left of x. From here on we limit our analysis to the top left quadrant, where room information from traces is available. Note that maintenance access rooms, which account for about 19 % of rooms in the top-left quadrant, can not be detected from either traces or from the grammar-based room layout.

Figure 8.7a shows the average error in the size of matched rooms compared to the fraction of rooms that have been found on detected hallways for all indoor maps built from 50 traces. The trace-based indoor map ("Traces") finds only very few rooms (up to 40 %) and shows a large error in the room size (0.3 m to 1.3 m), whereas using the accurate grammar ("Accurate"), many more rooms (at least 60 %) are found. Furthermore, the accurate grammar reduces the size error of rooms over the trace-based indoor map. At 1.0 m, the maximum size error using the grammar is less than that of traces. In addition, the grammar also provides room layouts with no error. The semi-accurate grammar ("Semi-Accurate") finds the same number of rooms, with a slightly higher average size error. Thus, using grammar support greatly increases the accuracy of the floor

plan even when the grammar is not perfect. Figure 8.7b shows the same metric for 90 traces. Note that while the trace-based indoor map hardly improves (42 % of rooms detected, maximum size error 1.1 m), the accuracy of the grammar-based approaches drastically increases (maximum size error reduced to 0.5 m). Thus, additional information has little impact on the trace-based indoor map, but significantly boosts the performance of the grammar-based approach.

To analyze the impact of the grammar further, we look at the overall fraction of matched rooms. Figure 8.6b shows the average number of matched rooms for each sample size of traces. Using 147 traces, we find only 40 % of rooms from traces alone, while using either grammar we detect 80 % of rooms, i.e., almost all rooms that can be found, with only 30 traces. Overall, we find up to 4 times as many matching rooms using the grammar-based map. Note the steep increase in the number of matching rooms from 10 to 30 traces. As argued before, more traces lead to detecting more hallways. For the trace-based map, the number of detected rooms grows with the number of traces. For the grammar-based map, the number of detected hallways is the important factor, as on each hallway, multiple matching rooms are detected. With only a few hints in existing rooms from the trace-based map, the grammar provides a far more accurate indoor map.

8.3. Energy-Efficient Mapping

To evaluate our energy-efficient mapping approach, we simulate the trace data acquisition process by replaying traces in random order. To include the WiFi Position Tracking system (WPT) in our simulation, we first collected a database of WiFi fingerprints for all hallway intersections and annotated our traces with real-world RSSI information taken at corresponding positions in the building. When a WiFi scan is performed, we use the annotated RSSI values to determine the closest anchor point.

During the replay of each trace, the Scheduler decides when to record trace data, i.e., enabling the IMU, and when to discard trace data, i.e., disabling the IMU and relying on WPT. Additionally, we record the energy for position tracking using the IMU or WPT, respectively, using the energy model from [BPDR14] (cf. Chapter 7). After the replay of additional traces, we update the trace-based map from all recorded data and recompute the quality classification (AM/IM) for each

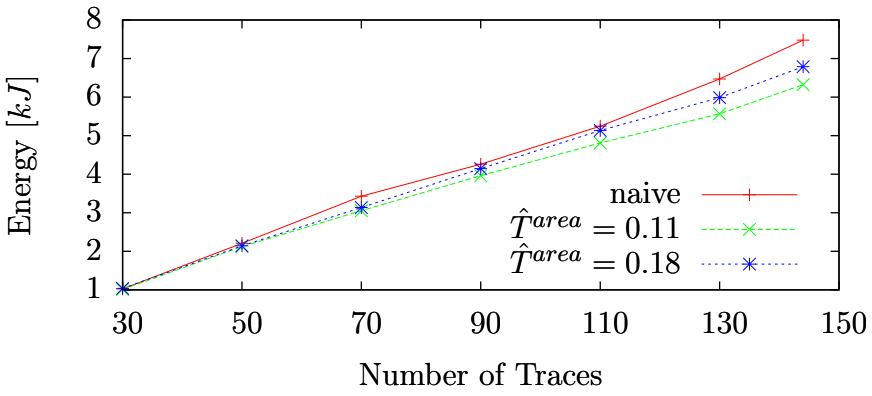


Figure 8.8.: Average Cumulated Energy Consumption over the number of traces.

area. To improve the runtime of our simulation, we replay batches of 20 traces before updating the trace-based map and the quality information.

Note that our data set of real-world odometry traces is an extremely challenging setup for our energy-efficient mapping approach. The traces in the set visit rooms on only 10 of the 21 hallways. Thus, most areas can never be classified as AM areas. As these permanent IM areas are located along the bottom and right side of the building, there is almost always an IM area directly reachable (i.e., without turning at an intersection), resulting in a high probability p_{IM} for visiting an IM area and, in turn, a low probability to disable the IMU.

In the following, we analyze the energy efficiency and quality of the resulting trace-based indoor map for our energy-efficient mapping approach for multiple quality thresholds $\hat{T}^{area}$ (cf. Table 8.1). Furthermore, we compare the results to that of a naive algorithm, where the IMU is always enabled. For each set of parameters, the simulation was repeated 50 times. Results are averaged over all simulations.

8.3.1. Evaluation Results

Figure 8.8 shows the average cumulated energy consumption of all participating devices over the number of traces. We can see that the energy consumption decreases, as we decrease the quality threshold $\hat{T}^{area}$ for considering an area to be accurately mapped. Furthermore, when at least 50 traces are recorded, the energy usage under our approach is consistently lower than the energy usage

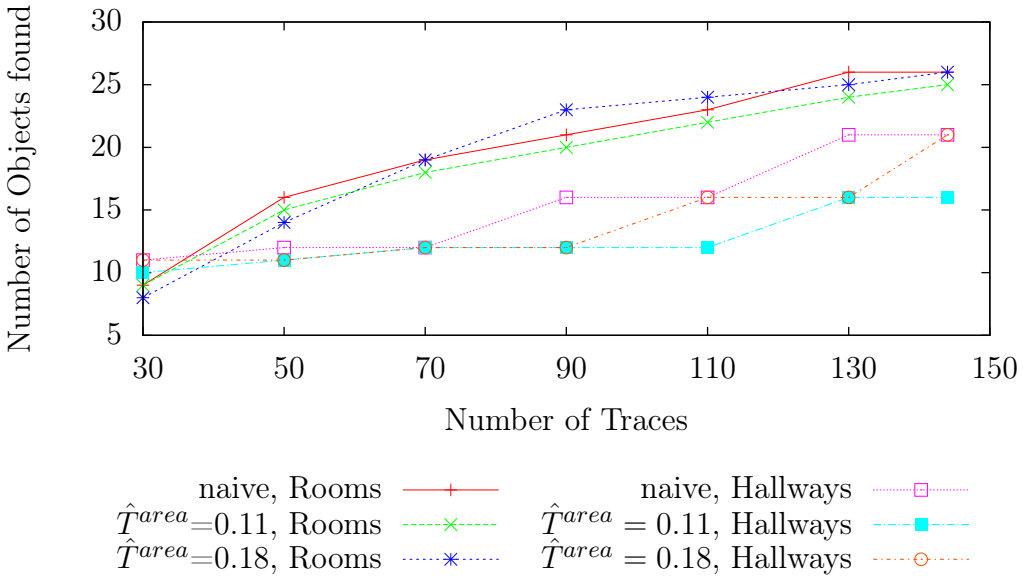


Figure 8.9.: Number of Floor Plan objects found over the number of traces.

under the naive approach. Using all traces, we can save 15 % of energy on average with $\hat{T}^{area} = 11\,\%$. However, if we consider the number of floor plan objects that were detected (depicted in Figure 8.9), we see that when setting $\hat{T}^{area} = 11\,\%$, we miss one room and all five hallways along one side of the building. This typically affects hallways with the least number of traces passing through them, i.e., those at the right side and the bottom of the building. If we set $\hat{T}^{area} = 18\,\%$, we find all floor plan objects while still saving 9 % of energy.

In addition, we also analyzed the influence of our energy-efficient approach on the accuracy of the position and size of rooms. Under the naive approach using all available traces, rooms are on average shifted by 3.9 m from their true position. Using our energy-efficient approach, this value slightly increases to 4.3 m for $\hat{T}^{area} = 18\,\%$ and 4.6 m for $\hat{T}^{area} = 11\,\%$. The difference in the recorded size of rooms is even smaller. Where the error in the width of rooms is 1.8 m for the naive approach, we observe 2.0 m and 2.2 m for $\hat{T}^{area} = 18\,\%$ and $\hat{T}^{area} = 11\,\%$, respectively. To explain the increase in these errors, we look at the accuracy of the IMU recovery process. The analysis of Baier et al. showed that the position error during IMU recovery is less than 3 m in 50 % of cases (less than 5 m in 95 % of cases). Thus, the increase in these errors can be explained by the limited accuracy of the IMU recovery process.

In summary, note that adjusting the quality threshold $\hat{T}^{area}$ is not a trade-off between energy consumption and quality. We can see from Figure 8.9 that the lower the quality threshold, the more traces are required to find the same number of floor plan objects. Thus, while setting $\hat{T}^{area} = 18\,\%$ achieves the same quality as the naive approach with a reduced energy consumption, we can reduce the energy consumption even further when we are willing to accept a longer delay until all floor plan objects have been found.

8.4. Summary

In this chapter we have demonstrated the effectiveness and efficiency of MapGenie. We evaluated MapGenie using 250 real-world odometry traces from four volunteers. Our evaluations show that the *trace-based mapping approach* can accurately extract the hallway skeleton and the set of rooms visited by participants, when the number of observations is sufficiently high. However, despite the size of our dataset, many rooms in the building have not been visited, leading to blank spots in the map. Even for visited rooms, the number of observations is often insufficient, leading to rooms being represented with the wrong size.

Our *grammar-based approach* significantly improves over these results. Even when given a small fraction of our data set, the grammar can provide a complete indoor map that features almost all rooms actually present in the building. When the amount of input information is increased, the accuracy of the trace-based map hardly changes while the accuracy of the grammar-based map drastically increases. These results hold even when the grammar is semi-accurate, i.e., does not perfectly match the observed floor. Thus, grammar-based mapping is a feasible approach for transferring knowledge about the indoor layout from a well-mapped building to other buildings with similar floor layouts.

Our evaluations also show the benefit of our *energy-efficient mapping approach*. Despite the challenging setup of our dataset where only few areas contain enough information to be classified as accurately mapped, we can decrease the energy consumption of MapGenie by up to 15 %. Note that the tradeoff for this increase in efficiency is a longer delay until all floor plan objects are detected, not a reduced quality of the resulting map. Thus, if the longer delay can be tolerated, our approach can save energy while still providing an accurate indoor map.

9. Related Work

OpenStreetMap [HW08, opea] has pioneered the construction of maps for outdoor areas from Public Sensing data on a large scale. By relying on GPS information to record the movement of participants, the world-wide road network can be mapped in an opportunistic manner. Recent approaches have adapted this concept to the construction of indoor floor plans, as we will show in the next section. In the subsequent section, we analyze how architectural principles are represented and exploited through the use of grammars and methods for the procedural generation of buildings and landscapes. Finally, we show what efforts have been made to analyze the quality of architectural information and to optimize the energy consumption of mapping approaches.

Indoor Mapping using Public Sensing In contrast to mapping outdoor areas, indoor mapping approaches can not rely on absolute positioning systems to track the movement of participants with sufficient accuracy [KBG+10]. The problem of generating a map of an unknown area in the absence of an absolute positioning system is known as "Simultaneous Localization And Mapping" (SLAM). In the area of robotics, Dissanayake et al. proposed a solution where a robot uses radar sensors to track its movement relative to landmarks in the vicinity [DNC+01]. When the robot encounters more landmarks over time, a Kalman filter is used to fuse the observations and generate an accurate map containing the relative locations of encountered landmarks. As this approach is computationally expensive and thus frequently not applicable to real-world problems, other research improved over this approach by proposing more efficient algorithms, e.g., FastSLAM [MTKW02]. Similar approaches based on other ranged sensors have been proposed for both robotic and human-operated applications, e.g., using sonar [TNNL02], laser scanning [FJVF10, BZF12], time-of-flight cameras [MDF+09], the Microsoft Kinect [ACMR11, NDI+11], or a combination of the above [pro15].

There are also dedicated mapping solutions commercially available, using a combination of multiple cameras and laser scanners to build full 3D indoor models [Gmb].

Lifting the requirement for specific hardware, i.e., a ranged sensor with depth information, other approaches aim to extract floor plan information from 2D images as provided by ordinary smartphone cameras. Wang et al. show how to extract information about the geometry of a room from a single photo of a cluttered scene [WGR13]. Peter et al. use a photo of an emergency evacuation plan to extract a floor plan for indoor positioning using map matching [PHSO10, PHF11]. Gao et al. use a set of 2D images to extract 3D geometry information about larger areas [GZY+14]. They assume the use of odometry data to record the translation and rotation of the image sensor between images.

However, all of the approaches presented so far have the drawback of requiring the respective sensor to be exposed, e.g., the smartphone may not be covered by clothing or stored in a backpack, thus severely limiting the time at which a device can be used for mapping. Shin et al. alleviate this problem by constructing a map purely from WiFi fingerprints [SC10]. However, they provide a topological map only, i.e., do not give any geometrical information.

Therefore, other approaches focus on creating the indoor floor plan entirely from odometry data recorded from inertial sensors, i.e., without any ranged sensors. Such approaches have the drawback of providing only maps of walkable areas instead of complete architectural features. However, odometry data provides geometry information and can be recorded regardless of the context of a participating device, e.g., even when the device is placed in the pocket of a participant. Examples include FootSLAM [AR12], SmartSLAM [SCC12], CrowdInside [AY12], ActionSLAM [HMC+13], CIMloc [ZJTS14], or Walkie-Markie [SCZ+13]. Almost all of these systems require the detection of landmarks, often in the form of turns or WiFi fingerprints, to combine multiple observations and to limit the drift error of odometry traces. A notable exception is FootSLAM, which in its original form corrects and combines traces using a particle filter. However, this approach is computationally complex and thus infeasible for a larger number of traces. Therefore, the authors have since improved it using various landmarking techniques [RAK10, BR11, RFA+13].

Closest to our approach, CIMloc and CrowdInside use DBSCAN for clustering

hallway segments, which has inspired our algorithm. However, CrowdInside relies on WiFi fingerprints for clustering rooms, which is not suitable for rooms smaller than the accuracy of the WiFi fingerprints. CIMloc does not consider rooms at all.

None of these approaches consider using knowledge about architectural principles or typical user behavior, such as frequently traveling parallel to an exterior wall, to improve the quality of odometry traces.

Applications of Grammar in Architectural Modeling Formal grammars and rule-based procedural modeling are frequently used to generate life-like objects and structures. Lindenmayer-systems [PL90] were originally proposed for modeling the structure of plants and artificially generating life-like plants. They have later been shown to be a practical tool for generating life-like cities as well. Parish and Müller [PM01] showed how to generate a street network and building shapes from information about population density and impassable areas (e.g., water, mountains). Talton et al. presented a general approach to use Lindenmayer-systems to generate plants and city layouts whose outlines follow a given shape [TLL+11]. While their work inspired our grammar-based modeling approach, their algorithm can only limit the outline of the resulting object, not its internal structure. This, however, is required to accommodate observations of rooms in a floor plan in our approach.

Other approaches focus on the layout of individual buildings. Huang et al. use a generative model to construct building roofs from a set of primitives to explain the observations of low-resolution cluttered aerial photographs and subsequently derive the correct slope of roofs and building outlines from these photos [HBS11]. Müller et al. used a grammar-based approach to combine simple building primitives and textures to generate a variety of complex, life-like building shapes [MWH+06], and to extract relevant textures from photos of building facades [MZWVG07]. Becker et al. extended this work to augment low-quality observations of a building facade (or unobserved parts) using information extracted from parts of the same facade that were observed with high quality [BHF08, BH09], which again inspired our work.

To model the indoor space of a building, Gröger and Plümer use attributed split grammars to create a 3D layout for the interior of a given building outline [GP10].

The indoor layout is set up according to geometrical and semantic constraints, e.g., so that every generated hallway has a connection to the outside world and every point in the building is actually reachable. Similar to this, Becker et al. developed a 2D version of the attributed split grammar [BPF+13]. The goal of their approach is to enable the automatic construction of the grammar from observations made through odometry traces. Peter et al. demonstrated how this grammar can be used to enhance the quality of a floor plan map that is fully known but possibly inaccurate [PBF13], while we used this grammar in our work to augment incomplete floor plans.

Optimized Mapping While a lot of effort has been put into making general Public Sensing system more energy-efficient (cf. Chapter 14), hardly any work has considered the optimization of mapping approaches. Similar to our approach, Baier et al. optimized the process of collecting mobility traces using GPS for the construction of outdoor maps [BWDR11]. They reduce the duty cycle of the GPS receiver when a device is located on a map element that has already been observed with high quality. To this end, they propose metrics that derive the quality of a map element from the joint quality of all observations for that element. However, their approach requires information about the measurement accuracy of individual observations, i.e., information about the accuracy of GPS position fixes, which is not available in approaches using odometry traces.

In the context of SLAM, Dissanayake et al. [DNC+01] also used a notion of quality to determine when a landmark was observed with sufficient quality to be used for navigation, or whether additional confirmations are required. However, their focus is on the quality of known objects. In contrast, our quality model also considers the question whether additional observations are likely to provide information about additional objects.

The question of quality of data is also of considerable interest to the Geospatial Information Systems (GIS) community. For example, ISO 19157 specifies general data quality topics and guidelines for evaluating quality metrics [ISO13]. Filippovska, for example, has developed actual metrics to determine the quality of a simplified building outline [Fil12]. However, such approaches rely on knowing the true shape of the object in question and are thus not applicable to the SLAM setting, where no ground truth information is available.

Part III.

Environmental Public Sensing

10. The DrOPS System for Optimized Environmental Public Sensing

Environmental data acquisition using Public Sensing systems gives experts and common citizens access to a wealth of data about their surroundings. This allows for the creation of novel, data-intensive applications, as well as increased environmental awareness. To facilitate the development of such PS systems, we present DrOPS (model-*Dr*iven *O*ptimization of *P*ublic *S*ensing systems), a flexible system for the efficient collection of spatially distributed environmental data using participating smartphones. DrOPS offers a mobility-independent application interface, centered around the concept of a *virtual sensor*, that lets applications use the system as a drop-in replacement for a static sensor network. Furthermore, we show how DrOPS aims to reduce the effort of sensor data acquisition by taking application-defined quality bounds into account. This reduces the energy consumption on participating devices and thus avoids driving participants from the system.

The purpose of this chapter is to give an overview of the DrOPS system. The first section defines our application interface, the Sensor Network Abstraction. Next we present the architecture of the DrOPS system and detail its components in relation to the overall system architecture in Section 10.2. Finally, in Section 10.3, we show a basic task execution algorithm for Request-Driven Execution as a basic implementation of our Sensor Network Abstraction, which we improve upon in the following chapters. To this end, we first present the *local optimization* in Chapter 11, where nearby devices coordinate to locally achieve k-coverage, i.e., to avoid taking and transmitting redundant readings. Chapter 12 details our model-driven approach for the *global optimization*, where we

exploit correlations between observed values to take and transmit only the most interesting readings, and its extensions to deal with smaller sets of participating devices. To demonstrate the performance of DrOPS, we discuss evaluation results from both a small-scale real-world testbed and a large-scale simulated setting in Chapter 13, before Chapter 14 concludes with a review of related work.

10.1. Sensor Network Abstraction

Previously, static sensor networks were the method of choice for large-scale sensor data acquisition. Such sensor networks are specially crafted for each application by experts, both from the application domain and from the sensor network operation domain. Application experts define requirements on what data should ideally be collected by specifying Points of Interest (POIs), data types and sampling rates. Operations experts then determine a placement of sensors so that the system best collects data from the requested POIs while ensuring that the system can operate, e.g., by deploying additional hardware to ensure the availability of communication channels among the sensors, or by adapting the collection methodology to maximize the battery lifetime of wireless sensors.

In Public Sensing, instead of deploying fixed sensors, we query for data from participating mobile devices. Due to the high mobility of mobile devices, manual selection of devices to gather data at a fixed set of POIs is not feasible, as we cannot expect applications to have intimate knowledge on the current state of the Public Sensing system, e.g., where mobile devices are located or where they will be moving next. Thus, the system must automatically select devices, constantly adapting to device mobility.

We tackle these problems in our Sensor Network Abstraction. This abstraction is based on the concept of *Virtual Sensors* as a mobility-independent data centric abstraction to request readings from a Public Sensing system. Using Virtual Sensors, applications can specify queries as the deployment of a static sensor network. In the following, we first present a formal definition of Virtual Sensors, before we define our query model that mimics the specification for the deployment of a static sensor network.

10.1.1. Virtual Sensor

In DrOPS, a *Virtual Sensor* (*v-sensor* for short) represents a Point of Interest, where data readings should be obtained. More formally, it is defined as follows. A v-sensor v is associated with a type of reading *v.type*, e.g., temperature or light intensity, a point in space $\overrightarrow{v}$, and a coverage area *v.area* defined relative to $\overrightarrow{v}$. While $\overrightarrow{v}$ indicates the location for which data is requested, *v.area* specifies a wider area surrounding $\overrightarrow{v}$, where data readings for v may be taken. The rationale behind this is twofold. On the one hand, it is unlikely that a participating device will pass the exact location of $\overrightarrow{v}$. Therefore, we allow readings to be taken at a distance. On the other hand, observations from environmental phenomena may be location-dependent, i.e., we assume that for two readings a, b taken some distance $\delta(\overrightarrow{a}, \overrightarrow{b}) > 0$ apart, their difference is limited by an (unknown) monotonically increasing error function of the distance: $|a - b| \leq e(\delta(\overrightarrow{a}, \overrightarrow{b}))$. For example, given a sound source and a device measuring the sound volume, the recorded volume decreases when the distance between source and device increases. The same holds true for every phenomenon that is subject to a diffusion process, such as the spatially distributed environmental phenomena we are focusing on in this work, e.g., air pollution or temperature. Therefore, applications may use *v.area* to bound the distance from $\overrightarrow{v}$ at which a reading is taken, thus ensuring that every device in *v.area* can register any phenomenon occurring in *v.area* with bounded error. Note that despite the definition of $e(\delta(\overrightarrow{a}, \overrightarrow{b}))$, we do not require *v.area* to be of circular shape. Thus, applications may use domain knowledge beyond $e(\delta(\overrightarrow{a}, \overrightarrow{b}))$ to further limit the area where data readings may be taken.

V-sensors provide measurements called virtual readings, which can either be *effective readings* or *inferred readings*. An effective reading r is taken by a participating device at position $\overrightarrow{r} \in$ *v.area*, whereas an inferred reading is computed using a model of the observed phenomenon without interaction with any participating device. If there is at least one participating device located in *v.area* to capture readings for v, we say that v is *available*. Otherwise, v is *unavailable*.

10.1.2. Query Model

Using the previous definition of v-sensors, applications can now request data readings from the PS systems by formulating a query $Q = (V, p, QoS)$, consisting of

a set of v-sensors $Q.V = \{v_1, v_2, \dots\}$ denoting where data should be obtained for this query, a sampling period $Q.p$ defining the interval at which data readings for all v-sensors in $Q.V$ should be provided, and a set of quality parameters $Q.QoS = \{q_1, q_2, \dots\}$ that control the operation of our algorithms. For example, quality parameter $k \in Q.QoS$ ($Q.QoS.k$ for short) configures the system for k-coverage: The system should report $Q.QoS.k$ independent readings $r_1, \dots, r_k$ for every v-sensor $v_i \in Q.V$ in every sampling period. Additional quality parameters depend on the algorithm being used and will thus be presented in the corresponding sections.

Note that we place two restrictions on the coverage areas $v_i.area$ over the definition given in the previous section. First, our local optimization requires the diameter of the coverage area to be less than the maximum wireless communication range: $\forall v \in Q.V \forall \overrightarrow{a}, \overrightarrow{b} \in v.area : \delta(\overrightarrow{a}, \overrightarrow{b}) \leq$ *ad-hoc communication range* (cf. Section 3.2). This enables the use of ad-hoc communication between all devices in the same coverage area. Second, all of our optimization algorithms require the coverage areas of all $v \in Q.V$ to be pairwise disjoint to ensure a unique mapping of participating devices and effective readings to v-sensors, regardless of the types of reading. While our local optimization may be extended to support overlapping coverage areas, e.g., by running separate instances of the algorithm for every covered v-sensor, disjoint coverage areas are a necessary prerequisite for the global optimization. If the same effective reading is assigned to multiple v-sensors, the global optimization will incorrectly detect a strong correlation between these v-sensors and may then provide unacceptably bad data.

No restrictions are placed on the shape of the coverage areas. For example, applications may specify circular areas with a fixed diameter δ_{max} to limit the total distance ($\overrightarrow{\pi} \in v_i.area$ iff $\delta(\overrightarrow{\pi}, \overrightarrow{v}) \leq \delta_{max}$), or use a voroni decomposition of the observed area if no limit should be imposed ($\overrightarrow{\pi} \in v_i.area$ iff $\forall v_j \in Q.V, i \neq j : \delta(\overrightarrow{\pi}, \overrightarrow{v_i}) < \delta(\overrightarrow{\pi}, \overrightarrow{v_j})$).

10.2. DrOPS System Overview

In this section we give an overview of the components of the DrOPS system (see Figure 10.1) that implement our Sensor Network Abstraction. *Applications* request data from the DrOPS system by posting a query Q, specifying the deploy-

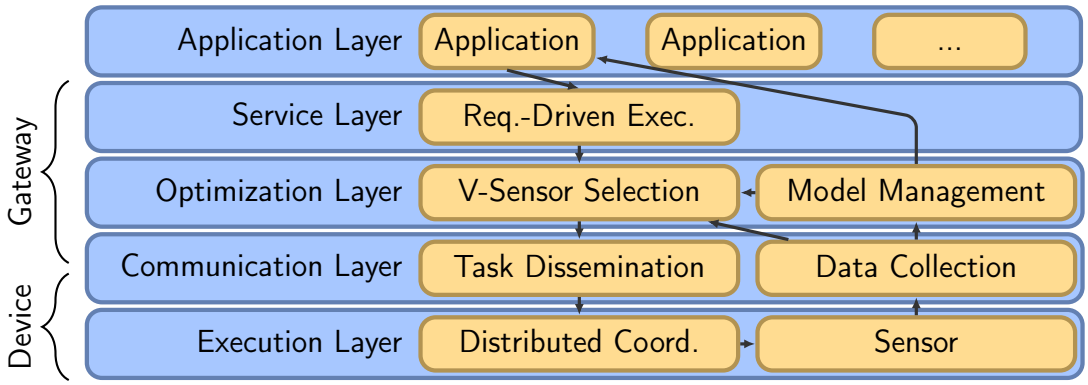


Figure 10.1.: Overview of the DrOPS Architecture

ment of a static sensor network, i.e., specifying a set of Points of Interest (POIs), where data should be obtained. The *Request-Driven Execution* component receives these queries and performs a *temporal decomposition*, generating a set of *tasks* for each query. Tasks represent a one-shot request for data readings from participating devices. Due to the constant mobility of participating devices, we have to reconsider which devices to select for taking readings for each query, or, in the case of periodic queries, even for each period. Thus, for periodic queries, a one-shot sensing task is generated in each period. For one-shot queries, we consider them to be periodic queries that are canceled after the first period has expired. Tasks are then forwarded to the *V-Sensor Selection* component. Based on available information about data received for current and past queries encoded in a data model, the V-Sensor Selection splits the set of v-sensors requested in the task into subsets for *effective readings* and *inferred readings*. It then temporarily modifies the task to remove the subset of v-sensors for inferred readings. The modified task, containing only requests for effective readings, is then forwarded to the *Task Dissemination* component for distribution to participating devices. Inferred readings will be provided later by the Data Collection component.

When a device receives a task, it uses its *Distributed Coordination* component to determine whether it should take an effective reading. If so, the *Sensor* component samples the respective sensor and forwards the reading to the Data Collection component. The *Data Collection* component maintains a database of past and current effective data readings. Using current effective readings and the data model used by the V-Sensor Selection component, it computes the values of the inferred readings that were removed from the task by the V-Sensor Selection

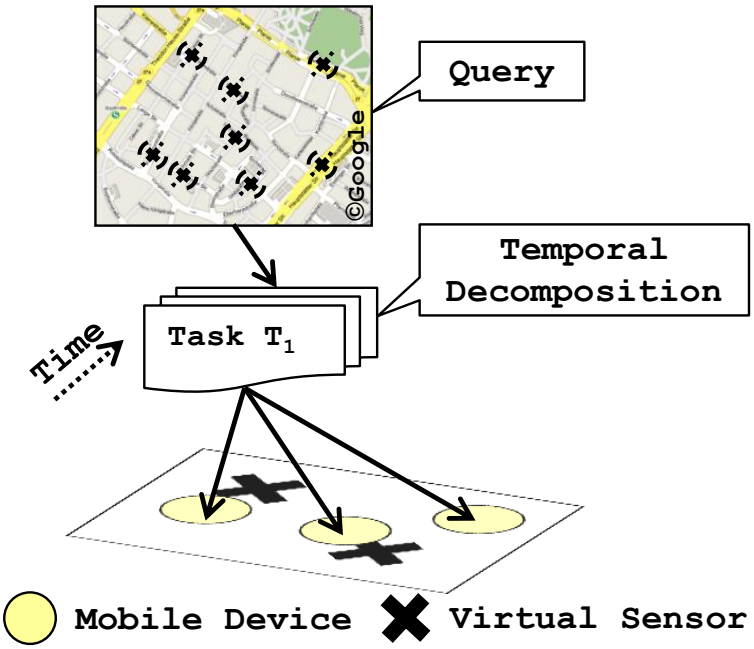


Figure 10.2.: Overview of the Basic Task Execution Algorithm

component, and adds them back to the task. In addition, inferred readings are annotated with additional information to inform an application about their quality. Furthermore, it constantly informs the V-Sensor Selection component about the progress of the collection of effective readings, potentially prompting the Model-Driven Optimization component to select additional effective readings to successfully complete the data acquisition task in the current period. At the end of each period, all readings are forwarded to the Model Management component. The *Model Management* component verifies whether the data model used by the Model-Driven Optimization component is still accurate, i.e., whether the inferred readings match the quality bound, or whether the model must be updated. Finally, effective and inferred readings are integrated into a unified task result R_Q and returned it to the application.

10.3. Basic Task Execution Algorithm

Next, we present the *Basic Task Execution Algorithm* for *Request-Driven Execution* as a simple implementation of our Sensor Network Abstraction. The Basic Task Execution is used to illustrate the basic operation of DrOPS and will later serve as a baseline for comparing the performance of our optimized approaches.

When a query $Q = (V, p, QoS)$ is submitted to the gateway, the *Request-Driven Execution* component performs a *temporal decomposition* of the query by generating a new task $T = (V, QoS)$ every p seconds at the beginning of each sampling period. Per our requirement on mobility-independent operation, the gateway service does not track the exact position of participating devices (cf. Section 3.1). Thus, it does not know which devices are close to a v-sensor. Therefore, the *Task Dissemination* component distributes each T to all participating devices located in a partition service area that intersects with the coverage area of any v-sensor in $T.V$. This process is illustrated in Figure 10.2.

On receiving a task, the *Distributed Coordination* component of a device n samples the positioning system and determines whether n is located within the coverage area of any v-sensor, i.e., $\exists v \in T.V : \overrightarrow{n}_{sens} \in v.area$. In this case, it takes a reading r of type $v.type$ and returns $(v, r, \overrightarrow{r})$, where $\overrightarrow{r} = \overrightarrow{n}_{sens}$, to the *Data Collection* component on the gateway. Otherwise, it discards the task. Note that devices execute tasks instantly, thus we do not introduce any delay in waiting for device positions to improve. In case there were more than $T.QoS.k$ readings taken for v, the Data Collection component selects the k readings $\overrightarrow{r_i}$ with minimum Euclidean distance $\delta(\overrightarrow{r_i}, \overrightarrow{v})$. The Data Collection component stores all selected readings together with the starting time of the corresponding sampling period. At the end of each sampling period, the collected effective readings are returned to the application as result R_Q. Note that with this basic algorithm readings are returned only for available v-sensors.

In the following chapters, we extend this basic sensing algorithm to account for v-sensors with a surplus number of participating devices in their vicinity, to minimize the number of necessary effective readings, and to compensate for unavailable v-sensors.

11. Local Optimization

In this chapter we present extensions to the *Distributed Coordination* component that implement the *local optimization*. While in the previous chapter the Distributed Coordination component was simply responsible for determining the proximity to a v-sensor, it now has the additional goal of reducing the effort for taking effective readings at a v-sensor. This is achieved by adjusting the number of readings taken for the v-sensor, i.e., selecting a (sub)set of participating devices to take and transmit effective data readings.

While selecting the subset of devices purely at random would provide the desired increase in efficiency, it would also decrease the quality of the data returned to the application, e.g., readings at a greater distance may be chosen. To quantify the quality and thus guide the selection of devices, we identified a set of quality metrics, i.e., coverage and distance of readings, as presented in the following section. Based on these quality metrics, we formulate the problem to be solved by our algorithms in Section 11.2. Finally, Section 11.3 presents our distributed algorithms for device coordination, that aim to optimize the selection of devices according to our quality metrics.

11.1. Quality Metrics

Intuitively speaking, a query result is of high quality if the data gives a sufficiently complete and accurate picture of the real world, e.g., no phenomenon was missed and measurements carry an error that is acceptable for the application. Whether or not given data can be considered complete and accurate is dependent on the application and thus out of the scope of the PS system. We have, however, identified two quality metrics that will allow a user to make this determination: Task coverage and distance of readings.

Task coverage is defined as the fraction of v-sensors for which k-coverage was

achieved, i.e., the fraction of v-sensors for which at least k readings were obtained. The environmental readings we are focusing on exhibit a spatial variance. Thus, a user can determine from the task coverage value whether any phenomenon might have been missed or the data is sufficient. When $Q.V$ was designed appropriately to observe the phenomenon of interest, i.e., a complete set of readings for all v-sensors will contain all required information but no redundant information, maximizing the task coverage leads to an increased chance of capturing more properties of the observed variable and, therefore, to a more meaningful query result as well.

The distance of a single reading is defined as $\delta(\overrightarrow{r}_{true}, \overrightarrow{v})$, the distance of the true position $\overrightarrow{r}_{true}$ where a reading was taken and the v-sensor $\overrightarrow{v}$ the reading is assigned to. As we assume that a reading may deviate from the true value at $\overrightarrow{v}$ if $\delta(\overrightarrow{r}_{true}, \overrightarrow{v}) > 0$, knowing $\delta(\overrightarrow{r}_{true}, \overrightarrow{v})$ indicates to the user the accuracy of the query result, i.e., how trustworthy the readings are. To determine the overall quality of a query result, the user is presented with the full set of reading distances and can then map these to a quality value according to his own requirements. However, as we know that the deviation is bounded by a monotonically increasing error function e, we can conclude that the quality of a query result increases as the single reading distances decrease.

11.2. Problem Description

In the *local optimization*, the basic problem that we solve is that given a task T, for each v-sensor $v \in T.V$, we want to return $T.QoS.k$ readings $r_i, i \in \{1, \ldots, T.QoS.k\}$ at a location $\overrightarrow{r_i}_{true}$. To provide a high quality result, we should minimize the distance $\delta(\overrightarrow{v}, \overrightarrow{r_i}_{true})$ for each v and maximize the task coverage. With perfectly accurate device positions, this goal is easy to achieve. However, we have to deal with position errors.

In addition, we also consider the efficiency of query execution. To this end, we design our algorithms for taking as few readings as possible, since every additional (redundant) reading increases the energy consumption on each device (reading the sensor, potentially processing the reading) and increases network load, as each reading has to be transmitted to the gateway.

	Independent	Coordinated
Probabilistic	Nearest-Neighbor Candidates	Coordinated Nearest-Neighbor Candidates
Distance-based	Deterministic Nearest-Neighbor Basic Task Execution	Distance-Coordinated Nearest-Neighbor

Table 11.1.: Taxonomy of selection algorithms

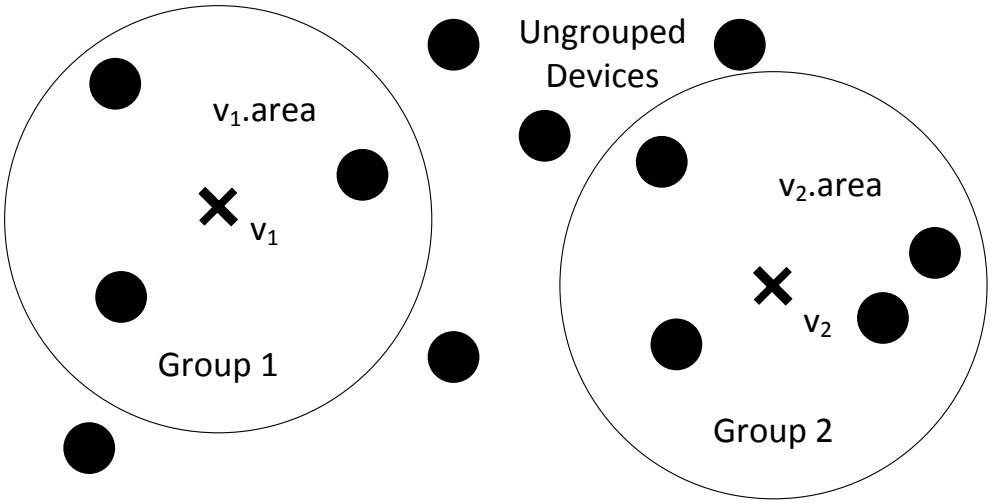


Figure 11.1.: Devices in the vicinity of v-sensors form groups. Dots are participating devices; crosses are v-sensors (v_1, v_2).

11.3. Distributed Algorithms for Device Coordination

We solve the problem outlined in the previous section using a set of distributed algorithms for device coordination. These algorithms are used by the Distributed Coordination components on participating devices to locally select a subset of devices for taking effective readings.

In the following section we present independent selection algorithms, where devices use only local knowledge to decide on whether they should take a sensor reading. Coordinated algorithms, where devices communicate to determine which devices should take sensor readings, are presented in the next section. A taxonomy of our algorithms is depicted in Table 11.1.

Algorithm 11.1 Independent Nearest-Neighbor Candidate (NNC) selection algorithm. $\min_k$ denotes the k'th smallest value from the set.

Require: Task $T = (V, QoS)$
  $\overrightarrow{r}_{sens} \leftarrow$ GPS fix
  $v \leftarrow v \in V$ where $\overrightarrow{r}_{sens} \in v.area$
  $distance \leftarrow \delta(\overrightarrow{v}, \overrightarrow{r}_{sens})$
  broadcast$(v, distance)$
  $\Delta \leftarrow \{distance\}$
  repeat
    receiveBroadcast$(v', distance')$
    if $v' = v$ **then**
      $\Delta \leftarrow \Delta \cup \{distance'\}$
    end if
  until timeout t_0 is reached
  $\delta_{limit} \leftarrow \min_{QoS.k}(\Delta) + \sigma$
  if $distance \leq \delta_{limit}$ **then**
    $v_{NNC} \leftarrow \{d | d \in \Delta \wedge d \leq \delta_{limit}\}$
    $p_{sensing} = \frac{QoS.k}{|v_{NNC}|}$
    return read$(v.type)$ with probability $p_{sensing}$
  end if

11.3.1. Independent Selection Algorithms

We present two independent selection algorithms where devices first form groups according to their location in a v-sensor area (see Figure 11.1), and then independently decide whether or not they will take a sensor reading. In all of our algorithms, the devices in one group have to be able to communicate directly using wireless communication, hence the requirement on the maximum diameter for the coverage areas of v-sensors in our query model (see Section 10.1.2).

The basic idea of the first algorithm, Nearest-Neighbor Candidates (NNC), is to probabilistically select devices from a set of candidate devices v_{NNC} for each v-sensor v. A device is included in v_{NNC} if it is located in $v.area$ and its distance to $\overrightarrow{v}$ is less than $\delta_{limit} = \delta(\overrightarrow{v}, \overrightarrow{n^k}_{sens}) + \sigma$, where $\overrightarrow{n^k}_{sens}$ is the the k'th closest sensed position to $\overrightarrow{v}$ of any device covering v and σ denotes the standard deviation of the sensed position as indicated by the positioning system. Each device in v_{NNC} computes a sensing probability $p_{sensing} = \frac{QoS.k}{|v_{NNC}|}$ and records a sensor reading with probability $p_{sensing}$. In detail, NNC works as follows (cf. Algorithm 11.1).

Upon receiving a task T, a device determines its sensed position $\overrightarrow{r}_{sens}$ and

Algorithm 11.2 Deterministic Nearest-Neighbor (DNN) selection algorithm. $\min_k$ denotes the k'th smallest value from the set.

Require: Task $T = (V, QoS)$
  $\overrightarrow{r}_{sens} \leftarrow$ GPS fix
  $v \leftarrow v \in V$ where $\overrightarrow{r}_{sens} \in v.area$
  $distance \leftarrow \delta(\overrightarrow{v}, \overrightarrow{r}_{sens})$
  broadcast$(v, distance)$
  $\Delta \leftarrow \{distance\}$
  repeat
    receiveBroadcast$(v', distance')$
    if $v' = v$ **then**
      $\Delta \leftarrow \Delta \cup \{distance'\}$
    end if
  until timeout t_0 is reached
  $\delta_{limit} = \min_{QoS.k}(\Delta)$
  if $distance \leq \delta_{limit} \vee |\Delta| \leq QoS.k$ **then**
    return read$(v.type)$
  end if

finds the v-sensor $v \in T.V$ where $\overrightarrow{r}_{sens} \in v.area$. If no such v exists, the device discards the task. Otherwise it broadcasts the identity of $v \in V$ and its distance to v, $\delta(\overrightarrow{v}, \overrightarrow{r}_{sens})$ to other devices. To learn about other devices in v_{NNC}, every device collects a set of broadcasts $\Delta = \{(v', \delta(\overrightarrow{v'}, \overrightarrow{r}_{sens}))|v' = v\}$ from other devices for a duration of t_0 from the reception of the task. t_0 is a system parameter whose value is empirically determined. In our evaluations, we use a value of 3 s (cf. Section 13.1.1). Next, each device locally computes δ_{limit} from the received distance values as given above and uses it to determine its own membership in v_{NNC} as well as the number of members $|v_{NNC}| = |\{\delta \in \Delta|\delta \leq \delta_{limit}\}|$. Note that v_{NNC} itself is never computed, as each device is only interested in the number of (other) members of v_{NNC}, not their identity.

For $\sigma = 0\,m$, v_{NNC} includes exactly the k nearest neighbors of v. For larger values of σ, v_{NNC} grows to increase the probability that the true nearest neighbors are included, even if other devices erroneously report closer positions. Note that as large errors in positioning are less likely than small errors, it is unlikely that a device at a very long distance is included in v_{NNC}, even when σ grows.

A major problem of this approach is that there is a significant probability $p_\emptyset = (1 - \frac{QoS.k}{|v_{NNC}|})^{|v_{NNC}|}$ that no device will take a sensor reading, e.g., for $k = 1$

and $|v_{NNC}| = 3$, $p_\emptyset \approx 30\%$. Thus, the task coverage is reduced. The reason for this is that the decision for taking a reading is statistically independent on all devices. While we could reduce $p_\emptyset$ by increasing $p_{sensing}$, this would also increase the amount of redundant sensor readings taken.

To alleviate this problem, we introduce a deterministic nearest-neighbor selection algorithm called DNN. For DNN we choose the devices that report the k smallest distances $\delta(\overrightarrow{v}, \overrightarrow{r}_{sens})$ to record a sensor reading instead of computing a random selection. In detail, DNN works as follows (cf. Algorithm 11.2).

Upon receiving a task, an exchange of distance information identical to NNC is performed. After the timeout t_0 has passed, each devices independently determines δ_{limit} as the k'th smallest distance of any device to $\overrightarrow{v}$. If the device is closer to $\overrightarrow{v}$ than δ_{limit}, it records a sensor reading.

By directly using the reported position of devices, DNN is subject to the same influence of location uncertainty as the basic approach. However, as shown by our evaluations (cf. Figure 13.2), it provides a better task coverage than NNC.

11.3.2. Coordinated Selection Algorithms

Another approach to reduce $p_\emptyset$ is to remove the statistical independence of the sensor selection. By introducing coordination amongst the devices, we ensure that at least k devices in a group will take a reading.

In the Coordinated Nearest-Neighbor Candidate (CNNC) selection algorithm, devices first perform an exchange of distance information identical to NNC to compute δ_{limit} (cf. Algorithm 11.3). Instead of taking a reading at a fixed probability, every nearest-neighbor candidate now chooses a random backoff time in the interval $[0, t_{CNNC}]$, where t_{CNNC} is a system parameter whose value is empirically determined. In our evaluations we used a value of 1 s (cf. Section 13.1.1). When the backoff timer expires, a device takes a reading and broadcasts a sensing notification message (SNM) containing the identity of v. Upon receiving the k'th SNM for v, a device aborts the backoff timer and discards the task.

We further introduce a coordinated variant of DNN, Distance-Coordinated Nearest-Neighbor selection (DCNN). Similar to CNNC, DCNN uses a backoff timer t and SNMs to decide which devices will take a reading. However, in DCNN t is chosen based on $\delta(\overrightarrow{v}, \overrightarrow{r}_{sens})$. Therefore, we do not need to exchange

Algorithm 11.3 Coordinated Nearest-Neighbor Candidate (CNNC) selection algorithm. $\min_k$ denotes the k'th smallest value from the set.

Require: Task $T = (V, QoS)$
  $\overrightarrow{r}_{sens} \leftarrow$ GPS fix
  $v \leftarrow v \in V$ where $\overrightarrow{r}_{sens} \in v.area$
  $distance \leftarrow \delta(\overrightarrow{v}, \overrightarrow{r}_{sens})$
  broadcast$(v, distance)$
  $\Delta \leftarrow \{distance\}$
  repeat
    receiveBroadcast$(v', distance')$
    if $v' = v$ **then**
      $\Delta \leftarrow \Delta \cup \{distance'\}$
    end if
  until timeout t_0 is reached
  $\delta_{limit} \leftarrow \min_{QoS.k}(\Delta) + \sigma$
  if $distance \leq \delta_{limit}$ **then**
    $t \leftarrow$ uniform$(0, t_{CNNC})$
    $count \leftarrow 0$
    repeat
      receiveBroadcast(snm(v'))
      if $v' = v$ **then**
        $count \leftarrow count + 1$
        if $count = QoS.k$ **then**
          Abort timeout and discard task
        end if
      end if
    until timeout t is reached
    broadcast(snm(v)) and **return** read$(v.type)$
  end if

information about other devices in the group prior to taking a reading. In detail, DCNN works as follows (cf. Algorithm 11.4).

First, a device determines its sensed position $\overrightarrow{r}_{sens}$ and computes the v-sensor $v \in V$ where $\overrightarrow{r}_{sens} \in v.area$. If no such v exists, the task is discarded. Next, it finds δ_{max} as the maximum distance from $\overrightarrow{v}$ to any point in $v.area$. The backoff timer t is then set to a value in $[0, t_{DCNN}]$ proportional to $\frac{\delta(\overrightarrow{v}, \overrightarrow{r}_{sens})}{\delta_{max}}$, i.e., proportional to the distance of the device from $\overrightarrow{v}$. t_{DCNN} is a system parameter whose value is determined empirically. In our evaluations we used a value of 0.5 s (cf. Section 13.1.1). When the backoff timer expires, the device takes a reading

Algorithm 11.4 Distance-Coordinated Nearest-Neighbor (DCNN) selection algorithm

Require: Task $T = (V, QoS)$
  $\overrightarrow{r}_{sens} \leftarrow$ GPS fix
  $v \leftarrow v \in V$ where $\overrightarrow{r}_{sens} \in v.area$
  $distance \leftarrow \delta(\overrightarrow{v}, \overrightarrow{r}_{sens})$
  $\delta_{max} \leftarrow \max(\{\delta(\overrightarrow{v}, \overrightarrow{x}) | \forall x \in v.area\})$
  $t \leftarrow \frac{distance}{\delta_{max}} \cdot t_{DCNN}$
  $count \leftarrow 0$
  repeat
    receiveBroadcast(snm(v'))
    if $v' = v$ **then**
      $count \leftarrow count + 1$
      if $count = QoS.k$ **then**
        Abort timeout and discard task
      end if
    end if
  until timeout t is reached
  broadcast(snm(v)) and **return** read($v.type$)

and broadcasts an SNM containing the identity of v. Upon receiving the k'th SNM for v, the device aborts the backoff timer and discards the task.

11.4. Summary

In this chapter we presented our *local optimization* approach to improve the energy efficiency of PS systems for environmental data acquisition. The idea of our approach is to minimize the number of readings per v-sensor: When a query has been configured for k-coverage and there are more than k devices present in the coverage area of a v-sensor, only k of these devices should take a reading.

To quantify the quality of the obtained data and thus guide the selection of devices, we identified a set of *quality metrics*: task coverage and distance of readings. We then presented a range of *selection algorithms* that optimize the selection of devices for these metrics: Two independent algorithms, where devices use local knowledge to decide whether to take a reading, and two coordinated algorithms, where devices communicate to determine which devices should take sensor readings.

12. Global Optimization

In this chapter we present our approach for the model-driven global optimization. In the global optimization we exploit correlations between observed values in spatially distributed environmental phenomena. On the one hand, we can improve the task coverage of the result by computing inferred readings for unavailable v-sensors. On the other hand, this allows us to increase the efficiency of the data acquisition process by reducing the number of effective readings taken.

In detail, our goal is to efficiently provide sensor data on spatially distributed environmental phenomena according to an application-defined quality bound $Q.QoS$, independent of the current distribution of devices in the observed area. We want to minimize the number of requested effective readings while at the same time compensating for unavailable v-sensors and maximizing the number of v-sensors $|V'|$ for which the quality constraints are fulfilled. To this end, DrOPS learns and maintains a model of the phenomenon observed in a query Q to optimize the data acquisition process.

We begin with a brief discussion of how to use a data model to optimize the data acquisition process in principle in the following section. In Section 12.2 we apply this concept to the Public Sensing setting, before we detail the algorithms used by our system in Sections 12.3 and 12.4.

12.1. Model-Driven Optimization

In the past, many approaches for modeling spatially distributed environmental phenomena have been proposed (cf. [Cre93, Slu08, MLR13]), the most promising of which are inverse distance weighing (IDW), Markov random fields (MRF), and multivariate Gaussian distributions (MGD). Next, we will discuss these approaches to identify which approach is most suitable for our global optimization.

In IDW an inferred reading is computed as a weighed sum of effective read-

ings from surrounding v-sensors. The weights are defined by a monotonically decreasing function of the distance of the corresponding v-sensors. Thus, IDW assumes that the distance of a pair of v-sensors is an indication of the similarity of their readings. However, consider a placement of light sensors on buildings of a university campus. Sensors placed on walls facing the sun will report similar values and sensors placed on walls facing away from the sun will report similar values, i.e., sensors in each group show a high correlation. Computing an inferred reading from only nearby effective readings ignores other available information, e.g., from v-sensors in the same group, that could improve the accuracy.

MRF and MGD do not make the assumption that only effective readings taken nearby are relevant to the value of an inferred reading. In MRF readings for a v-sensor v are modeled as a probability distribution that is dependent on the current values at v and its immediate neighbors. By iteratively resampling the reading of each v-sensor from the corresponding probability distribution, knowledge about all effective readings is propagated to all v-sensors. In an MGD a set of v-sensors $v \in V$ is modeled as a set of correlated one-dimensional Gaussian distributions, stored as a mean vector and a covariance matrix. Inferred readings are again computed by a weighed sum. However, in contrast to IDW, the weights are defined by the actual correlation of observed values rather than an indirect criterion such as spatial distance.

Sluiter [Slu08] and Mendez et al. [MLR13] have compared the performance of these modeling techniques. They found that they all provide usable inferred readings when effective readings for most v-sensors are available. However, when only few effective readings are available, MGD provided the most accurate inferred readings. As our goal is to minimize the number of effective readings, we will focus on using MGD to model the observed phenomena. Note that capturing other phenomena, e.g., discrete environmental phenomena such as individual events (e.g., lightning strikes), may require a different model.

In our system, an MGD model is used in two ways: Inferring missing values from a set of incomplete observations and selecting the best set of v-sensors to observe. We will first present the principle of how to apply MGDs to infer readings and select interesting effective readings in more detail, before we show how to use these principles in a PS system. For more in-depth information about the use of MGDs, we refer to [Cre93, GKS05].

12.1.1. Inference of Readings for Unavailable V-Sensors

Given a model MGD_V over a set V of v-sensors and a vector of effective readings $P_{V_{\text{eff}}}$ for a subset of v-sensors $v \in V_{\text{eff}} \subset V$, we can compute inferred values for all v-sensors $u \in V_{\text{inf}} = V \setminus V_{\text{eff}}$. To present the computation of inferred readings, we will use the notation defined by Guestrin et al. [GKS05] as follows.

μ_V denotes the *mean vector* of MGD_V. The entries of μ_V are the mean values of the individual Gaussian distributions for each v-sensor $v_i \in V$, denoted as μ_{v_i}, i.e., $\mu_V = (\mu_{v_1}, \mu_{v_2}, \ldots, \mu_{v_n})^T$. For $V' \subset V$, $\mu_{V'}$ denotes a projection of μ_V.

Similarly, $\Sigma_{V,V}$ denotes the *covariance matrix* of MGD_V, where individual entries $\Sigma_{u,w}$ denote the covariance value of v-sensors u and w. Entries on the diagonal denote the variance of the individual Gaussian distributions: $\forall v \in V : \Sigma_{v,v} = \sigma_v^2$. Furthermore, when $U, W \subseteq V, u \in U, w \in W$, then $\Sigma_{U,W}$ denotes a projection of $\Sigma_{V,V}$. $\Sigma_{u,W}$ and $\Sigma_{U,w}$ denote vectors of covariance values between u and all v-sensors $w \in W$ or vice-versa.

Using these definitions, we now show how an MGD can be used to compute inferred readings. Given $P_{V_{\text{eff}}}$ and a v-sensor $u \in V_{\text{inf}}$, the inference process derives a (univariate) Gaussian distribution for u from the Gaussian distribution for u in the MGD as follows:

$$\mu_{u|P_{V_{\text{eff}}}} = \mu_u + \Sigma_{u,V_{\text{eff}}} \Sigma^{-1}_{V_{\text{eff}},V_{\text{eff}}} (P_{V_{\text{eff}}} - \mu_{V_{\text{eff}}}) \tag{12.1}$$

$$\sigma^2_{u|V_{\text{eff}}} = \Sigma_{u,u} - \Sigma_{u,V_{\text{eff}}} \Sigma^{-1}_{V_{\text{eff}},V_{\text{eff}}} \Sigma_{V_{\text{eff}},u} \tag{12.2}$$

Here $\mu_{u|P_{V_{\text{eff}}}}$ refers to the inferred mean value (the inferred reading) for v-sensor u given the input of effective readings $P_{V_{\text{eff}}}$ for v-sensors in V_{eff}. $\mu_{u|P_{V_{\text{eff}}}}$ is computed as a linear combination of the mean value stored for u and the deviation of effective readings $P_{V_{\text{eff}}}$ from their respective stored mean values, weighed by a function of their covariance with u. $\sigma^2_{u|V_{\text{eff}}}$ denotes the variance of the inferred distribution and thus indicates whether the observations at V_{eff} were a good choice for inferring $\mu_{u|P_{V_{\text{eff}}}}$, i.e., whether the inferred value is likely to be within the user-specified tolerance. Note that $\mu_{u|V_{\text{eff}}}$ is independent of the actual values of effective readings but only depends on the identity of v-sensors in V_{eff}.

In the following, we will refer to the inference process as function INFER(MGD, P), where P denotes available effective readings.

Algorithm 12.1 Original GREEDY algorithm by Guestrin et al.

Require: Task $T = (V, QoS)$, Model $MGD_V = (M_V, \Sigma_{V,V})$, Budget of effective readings b.

$V_{\text{eff}} \leftarrow \emptyset$
$V_{\text{inf}} \leftarrow V$
while $|V_{\text{eff}}| < b$ **do**
  $I_{max} \leftarrow 0$
  $w \leftarrow \perp$
  for all $u \in V \setminus V_{\text{eff}}$ **do**
    $V'_{\text{eff}} \leftarrow V_{\text{eff}} \cup \{u\}$
    $V'_{\text{inf}} \leftarrow V_{\text{inf}} \setminus \{u\}$
    $I_u \leftarrow \frac{\Sigma_{u,u} - \Sigma_{u,V'_{\text{eff}}} \Sigma_{V'_{\text{eff}},V'_{\text{eff}}} \Sigma_{V'_{\text{eff}},u}}{\Sigma_{u,u} - \Sigma_{u,V'_{\text{inf}}} \Sigma_{V'_{\text{inf}},V'_{\text{inf}}} \Sigma_{V'_{\text{inf}},u}}$
    if $I_u > I_{max}$ **then**
      $w \leftarrow u$
      $I_{max} \leftarrow I_u$
    end if
  end for
  $V_{\text{eff}} \leftarrow V_{\text{eff}} \cup \{w\}$
  $V_{\text{inf}} \leftarrow V_{\text{inf}} \setminus \{w\}$
end while
return V_{eff}

12.1.2. Near-Optimal V-Sensor Selection

Given a budget of $b < |V|$ v-sensors to observe, the goal is to select effective readings $V_{\text{eff}} \subset V, |V_{\text{eff}}| = b$ so that the resulting inferred readings for $V_{\text{inf}} = V \setminus V_{\text{eff}}$ are most accurate. To identify the best V_{inf}, we use a mutual information criterion. The mutual information $MI(V_{\text{inf}}, V_{\text{eff}})$ of two sets of v-sensors (V_{inf} and V_{eff}, respectively) is defined as the difference in the Shannon entropy of inferred readings for v-sensors in V_{inf} when given no information (denoted as $H(V_{\text{inf}})$) and when given full knowledge about current sensor readings for v-sensors in V_{eff} (denoted as $H(V_{\text{inf}}|V_{\text{eff}})$):

$$MI(V_{\text{inf}}, V_{\text{eff}}) = H(V_{\text{inf}}) - H(V_{\text{inf}}|V_{\text{eff}})$$

By selecting V_{eff} so that $MI(V_{\text{inf}}, V_{\text{eff}})$ is maximized, we obtain effective readings that carry the most information about values at all other v-sensors and will thus allow us to compute the most accurate inferred readings. Guestrin et al.

showed that finding V_{eff} is an NP hard problem, but can be solved using a near-optimal heuristic [GKS05]. To this end, they proposed the GREEDY algorithm.

As shown in Algorithm 12.1, GREEDY partitions V into two subsets V_{eff} and V_{inf}. V_{eff} contains all v-sensors for which effective readings should be taken and V_{inf} denotes all v-sensors for which readings are computed using INFER. Partitioning is performed iteratively as follows. Initially, $V_{\text{inf}} = V$ and $V_{\text{eff}} = \emptyset$. In each iteration the v-sensor v that provides the maximum increase in mutual information is moved to V_{eff}, i.e., the v-sensor that reduces the uncertainty about the values at v-sensors in $V_{\text{inf}} \setminus \{v\}$ the most. Note that the selection only depends on the properties of the model, not on current observations. For a detailed discussion of this algorithm and the mutual information criterion, see [GKS05].

12.2. Model-Driven Global Optimization in Public Sensing

Overall, to apply model-driven optimizations as presented in the previous section to the PS setting, we face three challenges. The first challenge is that given a model of the observed phenomenon, we must adapt the selection of effective readings so that V_{eff} is selected according to the quality bounds specified in $T.QoS$ and independent of the current distribution of mobile devices. Second, the result R_Q returned to applications must be extended to include inferred readings computed using the model. Third, we must extend DrOPS to find the correct model to be used for inference and the selection of effective readings.

Existing approaches for optimizing data acquisition, e.g., [DGM+04, GKS05] target long-running queries and use either expert knowledge or a pilot deployment, typically lasting at least several days, to create a model that is accurate at all times. For instance, consider modeling temperatures which rise in the morning and drop in the afternoon. Training data from several days is required for an accurate model. However, to provide significant energy savings in PS, any optimization technique may take at most a fraction of the recharge cycle duration, which is typically one or two days, to set up. Furthermore, PS systems are built for heterogeneous types of queries with a-priori unknown duration, spontaneously issued by applications. For example, people commuting to their workplace may be

interested in the microclimate along potential routes to avoid bad weather conditions. Such queries, each defining an individual set of v-sensors, would be posted just before the start of a commute and canceled shortly thereafter. Clearly, maintaining models proactively for all possible types of queries is counter-productive if the query is never issued. Therefore, the basic idea of our approach is to use an *online learning algorithm (OLA)* to derive a model of the observed phenomenon on demand that is sufficiently accurate for the near future.

To tackle the challenges outlined above, we divide the runtime of a query into basic operation phases and into optimized operation phases. During a basic operation phase, queries are executed using the basic task execution algorithm. The V-Sensor Selection component outputs $V_{\text{eff}} = V$ and the Data Collection component returns effective readings for available v-sensors only. In parallel, the Model Management component executes OLA to create a new MGD. As soon as a new model is output by OLA, the basic operation phase ends.

After the basic operation phase, we switch to the optimized operation phase, where the V-Sensor Selection component uses variants of the GREEDY algorithm to modify tasks for a reduced number of effective readings $V_{\text{eff}} \subseteq V$. Furthermore, we extend the Data Collection component to compute inferred readings at the end of each task execution. Using INFER, readings for v-sensors in $V_{\text{inf}}^{+} = V_{\text{inf}} \cup$ *unavailable v-sensors in* V_{eff} are inferred locally from the effective readings reported by mobile devices. Thus, in contrast to the Basic Task Execution Algorithm, virtual readings for all v-sensors are provided by the system. We output the inferred mean values $\mu_{v|P_{V\setminus V_{\text{inf}}^{+}}}$ for each $v \in V_{\text{inf}}^{+}$ as inferred readings. All inferred readings are annotated with their inferred variance $\sigma^2_{u|V\setminus V_{\text{inf}}^{+}}$ to inform applications about the quality of the result. Since the model might not reflect changes happening over a longer time period—such as the temperature profile in the previous example—, the Model Management component continuously monitors model accuracy using an *online model validity check algorithm (MOCHA)*. When MOCHA considers the MGD to be inaccurate, we switch to the next basic operation phase.

The operation of the V-Sensor Selection component during the optimized operation phase is detailed in Section 12.3, followed by the presentation of our OLA and MOCHA algorithms in Section 12.4. The modification of the Data Collection component is straight-forward and thus not discussed further.

12.3. V-Sensor Selection Component

In this section we present the operation of the *V-Sensor Selection* component (cf. Figure 10.1). During the basic operation phase, tasks are passed on to the Task Dissemination component without further modification. Therefore, we will only discus the operation during the optimized operation phase.

We begin by presenting our ModifiedGreedy algorithm for *quality-based v-sensor selection* in PS. In the subsequent section, we extend our algorithm to AdaptiveGreedy. AdaptiveGreedy takes the availability of v-sensors into account and thus enables operation in a sparsely populated area where many interesting v-sensors may be unavailable.

12.3.1. Quality-based V-Sensor Selection

To optimize the operation of our system, we strive to minimize the size of V_{eff} while ensuring a data quality that matches the application-defined quality bound, i.e., limiting $\sigma^2_{u|V_{\text{eff}}}$ to an application-defined threshold $Q.QoS.\sigma^2_{max}$ for every $u \in V_{\text{inf}}$. The rationale behind this is that for sets of strongly correlated v-sensors, effective readings for a small subset are sufficient to yield good quality inferred readings for all v-sensors. However, Greedy selects a fixed number of v-sensors and can not adapt to a given quality bound (cf. Algorithm 12.1). Thus, we extend it to our ModifiedGreedy algorithm by changing the termination criterion (line 3). In our ModifiedGreedy algorithm, v-sensors are added to V_{eff} until $\forall u \in V_{\text{inf}} : \sigma^2_{u|V_{\text{eff}}} \leq T.QoS.\sigma^2_{max}$, ensuring that the variance of any v-sensor is less or equal to $T.QoS.\sigma^2_{max}$.

Given an MGD and a sensing task $T = (V, QoS)$, the V-Sensor Selection component now works as shown in Algorithm 12.2. We use ModifiedGreedy to select a set $V_{\text{eff}} \subseteq V$ of v-sensors. Then, a modified task T' is created, that requests effective readings for V_{eff} only. The task is subsequently passed to the Task Dissemination component for execution.

As the selection of V_{eff} only depends on the model, it is not strictly necessary to recompute V_{eff} for every task. However, this execution model allows us to easily integrate our validity check algorithm later on. While V_{eff} remains constant for the same model, due to device mobility, it is unlikely that a device has to provide multiple consecutive readings unless it remains stationary for a longer period of

Algorithm 12.2 Model-driven sensing task execution

Require: Task $T = (V, QoS)$, Model MGD_V
  $V_{\text{eff}} \leftarrow$ ModifiedGreedy(T, MGD_V)
  $T' \leftarrow$ new Task$(V_{\text{eff}}, T.QoS)$
  taskDissemination(T')

time. Therefore, we did not include explicit mechanisms for fair load balancing.

Note that the achievable degree of optimization depends on the magnitude of the correlations found in the data. If only weak correlations exist, ModifiedGreedy will select $V_{\text{eff}} = V$. Furthermore, the accuracy of the selection as well as the inference relies on the accuracy of the MGD. As we will show, the Model Management component ensures that the MGD in use always reflects current data.

As discussed previously, the gateway does not track positions of mobile devices and thus is not aware of v-sensor availability (cf. Section 3.2). Therefore, selecting V_{eff} is done in an optimistic fashion, assuming that all $v \in V$ will be available. This is not a problem for a dense coverage, where most v-sensors are available. For instance, as our experiments show, selecting unavailable v-sensors reduces the quality of inferred readings by at most 4 percentage points (cf. Section 13.2.4). We show how to compensate for unavailable v-sensors in sparse environments in the next section.

12.3.2. Adaptive V-Sensor Selection for Sparse Networks

The optimized task execution presented in the last section *assumes* that most or all of the v-sensors are constantly available. This assumption does not hold in a sparse network setting, which is characterized by a low probability for each individual v-sensor to be available. Thus, V_{eff} may be chosen so that no available v-sensor is selected. In principle, our model can still output inferred values, namely, its stored mean value and individual variance for each v-sensor. However, the mean values represent the average over all observations made in basic operation phases and the variances merely inform the application about the general variance of all readings at that v-sensor. These values carry no information about the current state of the observed phenomenon. This is a problem for both applications and our model management algorithms presented later.

Algorithm 12.3 AdaptiveGreedy algorithm for selecting V_{eff} in sparse network settings

Require: Task $T = (V, QoS)$, Model $MGD_V = (M_V, \Sigma_{V,V})$, Known unavailable V-Sensors V_{unav}, Known available V-Sensors V_{avl}

```
V_eff ← V_avl
V_inf ← V \ V_eff
while (∃v ∈ V_inf : σ²_{v|V_eff} > T.QoS.σ²_max) ∧ (V_eff ≠ V \ V_unav) do
    I_max ← 0
    w ← ⊥
    for all u ∈ V_inf \ V_unav do
        I_u ← mutualInformation(u, Σ_{V,V}, V_eff)
        V'_eff ← V_eff ∪ {u}
        V'_inf ← V_inf \ {u}
        I_u ← (Σ_{u,u} − Σ_{u,V'_eff} Σ_{V'_eff,V'_eff} Σ_{V'_eff,u}) / (Σ_{u,u} − Σ_{u,V'_inf} Σ_{V'_inf,V'_inf} Σ_{V'_inf,u})
        if I_u > I_max then
            w ← u
            I_max ← I_u
        end if
    end for
    V_eff ← V_eff ∪ {w}
    V_inf ← V_inf \ {w}
end while
return V_eff
```

To alleviate this problem, we begin by introducing the AdaptiveGreedy algorithm that includes *knowledge* about the (un)availability of v-sensors in the selection. Finally, we present how to use AdaptiveGreedy in our *Round-based Adaptive V-Sensor Selection*, that provides this knowledge, to allow DrOPS to compensate for unavailable v-sensors.

AdaptiveGreedy Algorithm

Compared to the previously described ModifiedGreedy algorithm, AdaptiveGreedy depicted in Algorithm 12.3 takes two additional parameters: A set of v-sensors *known* to be unavailable $V_{\text{unav}} \subseteq V$ and a set of v-sensors *known* to be available $V_{\text{avl}} \subseteq V, V_{\text{avl}} \cap V_{\text{unav}} = \emptyset$. V_{avl} and V_{unav} are determined by comparing the set of v-sensors selected for effective readings to the set of effective readings that were received. This is the task of the Round-based Adaptive V-

Sensor Selection presented in the next section. The availability of v-sensors not contained in $V_{\text{avl}} \cup V_{\text{unav}}$ is unknown. Using an optimistic strategy, AdaptiveGreedy *assumes* these v-sensors to be available, although they may turn out to be unavailable during task execution. A pessimistic strategy would need to probe the availability of all v-sensors beforehand by querying all participating devices for their position. This would cause the PS system to use as much energy as an approach without any optimization just for probing v-sensor availability, thus voiding the entire optimization approach.

Given these parameters, AdaptiveGreedy computes a new selection of v-sensors V_{eff} analogous to ModifiedGreedy under the additional constraints that no v-sensor known to be unavailable is selected and that all v-sensors known to be available are selected, i.e., $V_{\text{eff}} \cap V_{\text{unav}} = \emptyset$ and $V_{\text{avl}} \subseteq V_{\text{eff}}$. Forcibly selecting all of V_{avl} is warranted by the fact that in our system detecting the availability of v-sensor v coincides with getting an effective reading for v (see Section 12.3.2). Thus, not selecting all of V_{avl} would be a waste of effort.

Round-based Adaptive V-Sensor Selection

The quality-based v-sensor selection algorithm presented in the previous section is effective only in dense network settings, where most v-sensors are available. To adapt DrOPS to the sparse network setting, we now introduce the *Round-based Adaptive V-Sensor Selection*, depicted in Algorithm 12.4, that provides explicit knowledge about availability and unavailability and then uses AdaptiveGreedy to achieve a better selection.

In the *Round-based Adaptive V-Sensor Selection*, the V-Sensor Selection component subdivides the execution of each task into a number of rounds as determined by the application-specified quality parameter *T.QoS.rounds*. The duration of each round is $\frac{Q.p}{T.QoS.rounds}$, where $Q.p$ denotes the sampling period of the query, i.e., the duration of a task. Thus, the duration of task execution is evenly spread amongst all rounds. At the beginning of each round we first update our knowledge about current v-sensor availability. Based on this knowledge, we then update the set of v-sensors for which effective readings should be acquired. Note that for long durations of task execution $Q.p$, round duration should be limited to, e.g., 5 s each to ensure that participating devices cannot move too

Algorithm 12.4 Round-based adaptive v-sensor selection

Require: Task $T = (V, QoS)$, Model $MGD_V = (M_V, \Sigma_{V,V})$
  $V_{\text{avl}} \leftarrow \emptyset$
  $V_{\text{unav}} \leftarrow \emptyset$
  $E_0 \leftarrow \emptyset$
  $V_{\text{eff},0} \leftarrow \emptyset$
  for $i = 1..T.QoS.rounds$ **do**
    $V_{\text{avl}} \leftarrow V_{\text{avl}} \cup E_{i-1}$
    $V_{\text{unav}} \leftarrow V_{\text{unav}} \cup (V_{\text{eff},i-1} \setminus E_{i-1})$
    $V_{\text{eff},i} \leftarrow \text{AdaptiveGreedy}(T,\ MGD_V,\ V_{\text{unav}},\ V_{\text{avl}})$
    $V_{\text{eff},i} \leftarrow V_{\text{eff},i} \setminus \bigcup_{j=1}^{i-1} V_{\text{eff},j}$
    if $V_{\text{eff},i} = \emptyset$ **then**
      return $\bigcup_{j=1}^{i-1} E_j$
    end if
    $T_i \leftarrow$ new Task$(V_{\text{eff},i}, T.QoS)$
    taskDissemination(T_i)
    $E_i \leftarrow$ dataCollection(T_i)
  end for

much between individual rounds. Otherwise, the availability of v-sensors may significantly change during each round, thus voiding the knowledge on v-sensor availability built so far. For the same reason, we do not carry over knowledge from past sampling periods, as devices may have moved significantly between sampling periods.

In the first round, $V_{\text{avl}} = \emptyset = V_{\text{unav}}$, i.e., we assume all v-sensors to be available. Therefore, the initial selection of $V_{\text{eff},1}$ is identical to using ModifiedGreedy as in the non-adaptive system. In fact, when setting $T.QoS.rounds = 1$, the system behaves exactly as presented in the previous section. The resulting subtask T_1 is distributed to the mobile devices. For all v-sensors in $V_{\text{eff},1}$ that are actually available an effective readings will be reported to the gateway. All readings received in this round are stored in set E_1.

In subsequent rounds $i = 2 \ldots T.QoS.rounds$, we first update our knowledge on v-sensor availability by setting $V_{\text{avl}} = V_{\text{avl}} \cup E_{i-1}$ and $V_{\text{unav}} = V_{\text{unav}} \cup (V_{\text{eff},i-1} \setminus E_{i-1})$. Thus, all v-sensors for which a reading was requested but no effective reading was received are known to be unavailable for the remainder of the sampling period. Based on this new knowledge, we then compute a new selection $V_{\text{eff},i}$ using AdaptiveGreedy. A new subtask $T_i = (V_{\text{eff},i} \setminus \bigcup_{j=1}^{i-1} V_{\text{eff},j})$ is then

distributed to the mobile devices. We repeat this process until either the maximum number of rounds has been reached or no additional v-sensors were selected. At this point, the Data Collection component computes inferred readings from all effective readings that have been collected. Note that adjusting the number of rounds allows the application to define the trade-off between quality of inferred readings and efficiency of operation. We will analyze the performance of DrOPS for varying values of *T.QoS.rounds* in our evaluation (cf. Section 13.2.4).

12.4. Model Management

Our optimization algorithms presented in this chapter so far have assumed the availability of a correct MGD. Providing this MGD for an observed phenomenon is the task of the *Model Management* component (cf. Figure 10.1). To this end, it computes a MGD that is sufficiently accurate for the near future, and continuously monitors its accuracy to determine when a new model is required.

Next, we begin by detailing our online model validity check algorithm MOCHA, before we present our online learning algorithm OLA in detail.

12.4.1. MOCHA

The goal of MOCHA is to check a model for correctness. Intuitively, a model is correct if the Gaussian distributions of inferred readings fit the real data, i.e., inferred values from v-sensors $v \in V_{\text{inf}}^{+} = V_{\text{inf}} \cup$ *unavailable v-sensors in* V_{eff} (cf. Section 12.2) center around the true mean value and the variance matches the true variance. However, checking this property for every v would require constant sampling of all v-sensors and thus render the optimization useless.

Therefore, we take a different approach for MOCHA (see Algorithm 12.5). We modify the V-Sensor Selection component to randomly choose a set of control sensors $V_{\text{ctl}} \subseteq V_{\text{inf}}$ at the beginning of each sampling period, where the size of V_{ctl} is determined by the application-defined quality parameter *T.QoS.ctrl*. In addition to the v-sensors V_{eff} selected by the ModifiedGreedy algorithm, we request effective readings for V_{ctl}. When the round-based alternate V-Sensor Selection is used, we select additional v-sensors for V_{ctl} in each round, until effective readings for *T.QoS.ctrl* v-sensors have been received or the sampling period has

Algorithm 12.5 MOCHA algorithm. C denotes the set of v-sensors used for control readings.

Require: Final Result R_Q, Control Readings $C_{V_{\text{ctl}}}$, Threshold $T.QoS.T$, Acceptable Violations $T.QoS.violations$

RMSE $\leftarrow 0$
for all $c \in V_{\text{ctl}}$, c available **do**
    RMSE $\leftarrow$ RMSE $+ (R_c - C_c)^2$
end for
RMSE $\leftarrow \sqrt{\frac{\text{RMSE}}{|V_{\text{ctl}}|}}$
if RMSE $\geq T.QoS.T$ **then**
    Add "violation" to window
else
    Add "no violation" to window
end if
if Number of violations in window $> T.QoS.violations$ **then**
    return "Model Invalid"
else
    return "Model Valid"
end if

passed. At the end of the sampling period, only effective readings from (available) v-sensors in V_{eff} are used as input for the inference algorithm. We then compute the root mean squared error (RMSE, lines 1 to 5) of mean values of inferred readings and their corresponding effective control readings. Using the RMSE, we avoid the problem of comparing individual samples to inferred distributions since we can compare absolute values directly. Furthermore, by adjusting the size of V_{ctl}, we can trade off the costs for effective sampling and the probability of detecting inaccurate models (Quality of Service).

If RMSE $> T.QoS.T$, where $T.QoS.T$ is a predefined threshold (part of the quality specification $T.QoS$ of a sensing task), we say that the threshold has been violated. To avoid discarding an accurate model in case the observed violation was an outlier, we use a sliding window approach to dampen the reactivity of MOCHA (lines 6–15). We define a model to be inaccurate if there are $T.QoS.violations$ within the last $T.QoS.win$ samples. For example, for $T.QoS.win = 2$ and $T.QoS.violations = 1$, we discard the model after two consecutive violations. $T.QoS.win$ and $T.QoS.violations$ are part of the application-defined quality specification of a sensing task. Applications set these values to define a trade-

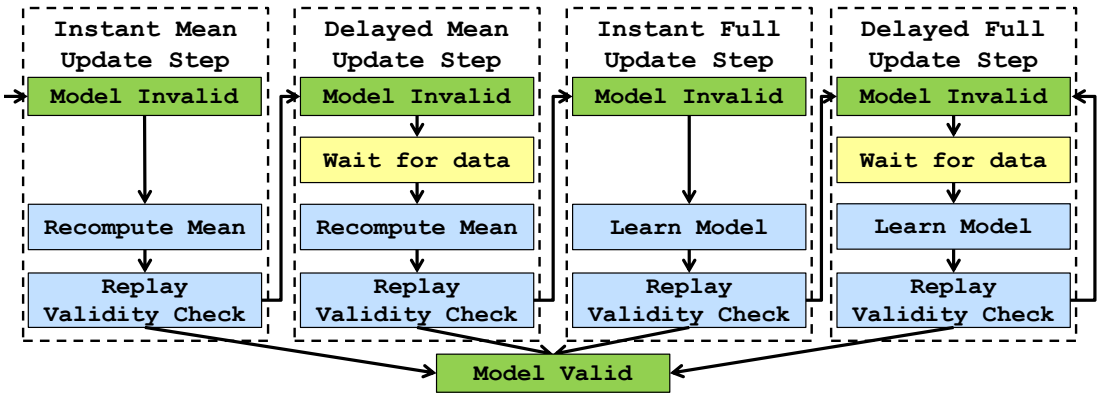


Figure 12.1.: Overview of the basic operation phase

off between result quality, i.e., avoiding low-quality inferred readings by quickly abandoning a model, and efficiency, i.e., accepting some low-quality inferred readings to reduce the amount of effective readings, according to their requirements. In our evaluations (cf. Section 13.2.2) suitable values were empirically determined from preliminary experiments.

Using MOCHA, we now introduce our online learning algorithm OLA.

12.4.2. OLA

During the basic operation phase, OLA aims to create a new MGD of the observed phenomenon. To this end, the basic operation phase is subdivided into four steps (cf. Figure 12.1). In each step, a new model is created using all available data obtained in the current basic operation phase and all previous phases (both basic and optimization) for the same query up to a certain age, given in the application-defined quality parameter $T.QoS.maxAge$. While reducing $T.QoS.maxAge$ reduces the runtime of the learning algorithm, it should be set to a large enough value to accommodate expected periodic shifts. For example, for temperature shifts in day/night cycles, $T.QoS.maxAge$ should be set to a multiple of the cycle duration. In preliminary experiments using week-long datasets of temperature readings in our simulation setup (cf. Section 13.2.2), $T.QoS.maxAge = 3$ days showed best results.

At the end of every step, we replay ModifiedGreedy, Infer, and MOCHA on data from the last $T.QoS.win$ sampling periods. If the new model is considered valid by MOCHA, we immediately switch to the next optimized operation phase

Algorithm 12.6 Instant Mean Update Step

Require: $T = (V, p, QoS)$, MGD_V
$\forall v \in V : M'_V[v] = \perp$
2: $W \leftarrow \emptyset$
for all $v \in V$ **do**
4: $data \leftarrow$ DataCollection.getHistoricData$(v, [QoS.maxAge, \text{now}()]))$
if $|data| > 0$ **then**
6: $M'_V[v] \leftarrow \text{mean}(data)$
else
8: $W \leftarrow W \cup \{v\}$
end if
10: **end for**
$M'_V[W] \leftarrow$ Infer(MGD_V, M'_V) **return** MGD'_V

using this model. This ensures that OLA terminates in the earliest possible step. In the following, we explain each step in detail.

In the **instant mean update step** (Algorithm 12.6) we update only the mean vector of the previous MGD from existing data. This is motivated by the observation that when an MGD is considered invalid, the covariance matrix is often still correct, while the mean vector failed to account for a global shift in the observed phenomenon. To avoid costly effective readings from all v-sensors, the mean vector is directly computed from data currently available from the Data Collection component (lines 3–10). If no effective readings have been reported in the considered history for a v-sensor, a mean value for this v-sensor is inferred using the old model (line 11). This might yield a greater error in inferred readings, which, however, would be detected by MOCHA.

If the new model is invalid, all v-sensors are queried for effective readings in the **delayed mean update step** (Algorithm 12.7). When $T.QoS.minRdg$ effective readings have been received from at least $T.QoS.minSens$ v-sensors each (line 6), a new model is constructed as in the previous step (line 7). Here $T.QoS.minRdg$ specifies the desired number of new readings per v-sensor to be used in the creation of a new model. Setting $T.QoS.minSens < |T.V|$ allows a new model to be created even when for a small subset of v-sensors fewer than $T.QoS.minRdg$ readings are available. This prevents OLA from getting stuck in the delayed mean update step when a (small) subset of v-sensors has low availability. Both parameters are defined by the application according to its quality-requirements

Algorithm 12.7 Delayed Mean Update Step

Require: $T = (V, p, QoS)$, MGD_V
$waitStart \leftarrow$ now()
repeat
  wait for next sampling period
  $V_{\text{eff}} \leftarrow V$
  request data
until $|\{v \in V || \text{DataCollection.getHistoricData}(v, [waitStart, \text{now}()])| \geq QoS.minRdg\}| \geq QoS.minSens$ or $totalWaitTime > QoS.totalWaitTime$
return InstantMeanUpdate(R, MGD_V)

using expert knowledge of the observed phenomenon. Values in our experiments were empirically derived from preliminary experiments (cf. Section 13.2.2).

If the model is still considered invalid after the delayed mean update step, we update both the mean vector and covariance matrix in the **instant full update step** (Algorithm 12.8). This update is performed using the existing offline learning algorithm by Schwaighofer et al. [STY05], which roughly works as follows.

Initially, we set the mean vector and covariance matrix to seed values, computed as the sample mean and sample covariance from effective readings for a small fraction of sampling periods in the existing data. The mean vector and the covariance matrix are then updated iteratively as follows. In each iteration we first compute inferred readings for all v-sensors and all sensing periods in the existing data using the current model. Next, we compute a new mean vector and covariance matrix as a weighed sum of the seed values and the difference of inferred readings to the respective effective readings in the existing data. This process is repeated until additional iterations no longer change the model.

Note that this algorithm requires at least three effective readings for a v-sensor v to be able to include v in the model. Thus, if the number of effective readings available for v at the end of the basic operation phase is insufficient, v is excluded from the model.

If the new model remains invalid, we continue with the **delayed full update step** (Algorithm 12.9). In this step, we request fresh effective readings from all v-sensors and execute the offline learning algorithm when sufficient data has arrived. We repeat this step until a valid model has been created.

Note that OLA will remain in this step indefinitely if either a sufficiently large

Algorithm 12.8 Instant Full Update Step

Require: $T = (V, p, QoS)$

$data \leftarrow$ DataCollection.getHistoricData(V, [$QoS.maxAge$, now()])

return learnModel($data$)

fraction of v-sensors remains unavailable or the replay of sensor data causes a model to be falsely considered invalid. Therefore, applications may specify a hard runtime limit $T.QoS.totalWaitTime$, e.g., a multiple of the expected duration of the delayed full update step, after which a new full model is learned from the available data and considered to be valid without further checking. Should the new model be invalid, this is detected by MOCHA after at most $T.QoS.window$ sampling periods. This strategy ensures that an attempt is made to optimize the data acquisition even under adverse conditions, e.g., for only a small subset of v-sensors. As v-sensors not contained in the model are queried for effective readings in every sensing period, these v-sensors can then be included in the model in later basic operation phases.

In the worst case, i.e., when the model is invalid every time the runtime limit is reached, this strategy will lead to inferred readings that violate the user-specified quality bound for $T.QoS.window$ sampling periods after every time interval $T.QoS.totalWaitTime$. If this behavior is undesirable for an application, it may set $T.QoS.totalWaitTime = \infty$ to disable the runtime limit.

12.5. Summary

In this chapter we presented the model-driven *global optimization*. Here we exploit correlations between observed values, modeled as a multivariate Gaussian distribution (MGD). On the one hand, an MGD can be used to infer readings for v-sensors where no effective reading is available. On the other hand, this allows us to increase the efficiency of the data acquisition process by reducing the number of effective readings taken.

To apply MGDs to the public sensing setting, we presented an *online model learning algorithm* (OLA) for the quick construction of MGD models and an *online model validity check algorithm* (MOCHA) to verify whether a given model still provides sufficiently accurate inferred readings. Furthermore, we detailed

Algorithm 12.9 Delayed Full Update Step

Require: $T = (V, p, QoS)$
$waitStart \leftarrow \text{now}()$
repeat
  wait for next sampling period
  $V_{\text{eff}} \leftarrow V$
  request data
until $|\{v \in V || \text{DataCollection.getHistoricData}(v, [waitStart, \text{now}()])| \geq QoS.minRdg\}| \geq QoS.minSens$ or $totalWaitTime > QoS.totalWaitTime$
$data \leftarrow \text{DataCollection.getHistoricData}(V, [QoS.maxAge, \text{now}()])$
return learnModel($data$)

two algorithms for finding the minimum-size set of effective readings to request so that the resulting inferred readings still match a user-defined quality bound. The *quality-based v-sensor selection algorithm* selects v-sensors only according to the user-specified quality bound. Thus, it is only suitable for dense networks, where all selected v-sensors are available. Our *adaptive v-sensor selection* improves over the quality-based approach by also taking v-sensor availability into account. It thus extends our approach to sparse networks where many v-sensors may be unavailable.

13. Evaluation

In this chapter, we evaluate the performance of our optimizations for the DrOPS system. We first show the benefit of the local optimization in a simulated setting in Section 13.1. Then, we focus on the global optimization. In Section 13.2 we show a small-scale real-world prototype running our quality-based v-sensor selection algorithm as a proof-of-concept. Furthermore, to get more insight about the behavior of our system for large-scale tasks, we also implemented and evaluated both the quality-based and the adaptive v-sensor selection in a simulated environment.

13.1. Local Optimization

We present the methodology used for our evaluation in subsection 13.1.1. Subsections 13.1.2 and 13.1.3 present and discuss the results for quality metrics (cf. Section 11.1) and efficiency metrics.

13.1.1. Simulation Setup

We implemented our sensor selection algorithms for the OMNeT++ network simulator [Var] using the INETMANET extension. For the ad-hoc WiFi communication we used the 802.11 implementation from INETMANET. To improve the runtime of our simulation we restricted the maximum WiFi communication range to 150 m. For the mobile internet connection we created a simple model of a 3G network with a data rate of 386 kbps shared amongst all devices and a delay modeled according to empirical measurements [CGGPC06]. Device mobility was generated using CanuMobiSim [Ste] on a 1 km^2 street graph fragment of the city-center of Stuttgart.

For the Task Dissemination component, we use a multi-hop clustering algorithm

Number of participating devices	#devices	200, 500, 1000
Standard deviation of sensed positions	σ $[m]$	0, 1, 3, 5, 10, 30, 50
Radius of v-sensor area	δ_{max} $[m]$	5, 25, 50
Timeout for grouping devices	t_0 $[s]$	3
Maximum backoff for CNNC	t_{CNNC} $[s]$	1
Maximum backoff for DCNN	t_{DCNN} $[s]$	0.5
Device speed range	$[m/s]$	$0.5 \ldots 1.7$
Readings per v-sensor	k	1

Table 13.1.: Values for the variables changed in our simulations

[HFP12]. It provides a routing layer between the gateway and all mobile devices. In each cluster, only the cluster head (CH) uses its mobile data connection for internet access. Tasks are sent to the CH and then relayed via WiFi broadcast. Readings from cluster members are collected at the respective CH and then sent via the mobile data connection. Intra-cluster routing is done using a standard ring-based routing approach. Cluster membership of a device is determined by the hop count from the device to a CH. Devices always join the cluster of the closest CH, if that CH is closer than some system defined maximum hop count. If no CH is available, unclustered devices form a new cluster and elect a new CH.

To create uncertain position fixes, we generate an offset by picking a direction uniformly at random and drawing a distance from a normal distribution with standard deviation σ. This offset is then added to the real position of a device.

Our simulations ran for roughly 10 simulated minutes each. In all simulations we introduced about one query per minute, yielding 10 queries per simulation. At most one query was active at any given time during a simulation run. V-sensors were placed on roads only. The minimum distance of v-sensors was set to $2 \cdot \delta_{max}$, as the area of each v-sensor v was defined as a circle with radius δ_{max} centered at $\overrightarrow{v}$. The parameters chosen for our simulation are presented in Table 13.1. For every set of parameters we repeated the simulation ten times.

Simulations with a varying number of devices and different values for δ_{max} showed the expected behavior: Coverage goes down when there are fewer devices and the mean distance goes up when δ_{max} grows. Since these results are obvious, we will not consider different device numbers and δ_{max} values in the following simulations, and use the following values instead: #devices = 1000, $\delta_{max} = 100\,m$. k was fixed to 1.

13.1.2. Quality of Query Results

The first quality metric we consider is the distance of readings to v-sensor locations. To get a more accurate impression of how well our algorithms perform, we calculate the distance of each reading based on its true position r_{true} rather than the sensed position that would be recorded in a real system. Our measurements are shown in Figure 13.1. Note that each data point is averaged over all readings returned for any query of the corresponding parameter set. The average distance of the basic task execution algorithm ("basic") for $\sigma = 0\,m$ defines the optimum average distance of $14.4\,m$ for our scenario.

The overall increase in average distance of all algorithms is a direct consequence of location uncertainty. With a growing error in positioning, the actual distance of the device reporting to be closest to the v-sensor location may increase for varying position uncertainties σ. The distance-based algorithms, which always use the reading that was reportedly taken closest to the v-sensor location—basic task execution, DNN and DCNN—yield the smallest distance for any σ. For the probabilistic approaches the average distance of readings quickly increases as σ grows. Since the nearest-neighbor set grows with σ, these approaches basically select any covering device at random for large σ.

Second, we look at the task coverage of queries. Figure 13.2 shows our measurements for the task metric. Each data point is averaged over all queries for the corresponding parameter set. The basic task execution algorithm again defines the optimum task coverage at a stable percentage of 79%. Thus 21% of v-sensors were not covered by any device.

Compared to the basic task execution algorithm, the task coverage of all our algorithms degrades as σ increases. Apart from NNC, the reason for this degradation is that for the task coverage we did not include readings where $\delta(r_{true}, \overrightarrow{v}) > \delta_{max}$. As the location uncertainty increases, there is a chance for taking a reading at a distance larger than δ_{max}, which is what happens for DNN. DCNN improves over DNN because, as we will see later, it does take a large amount of redundant readings. It is, therefore, likely that at least one of the readings for each v-sensor was taken at a distance smaller than δ_{max}. The task coverage of CNCC is even lower than that of DNN since CNNC generally picks readings at a larger distance, as can also be seen in Figure 13.1. NNC

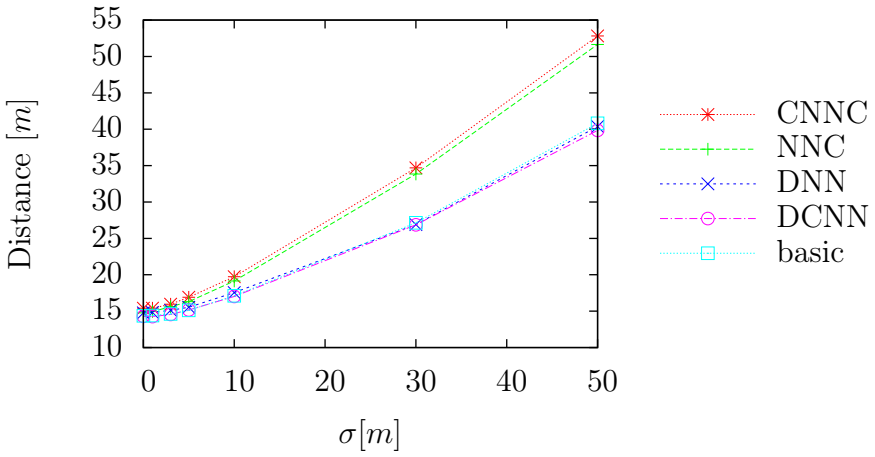


Figure 13.1.: Mean distance of true reading positions to v-sensor locations

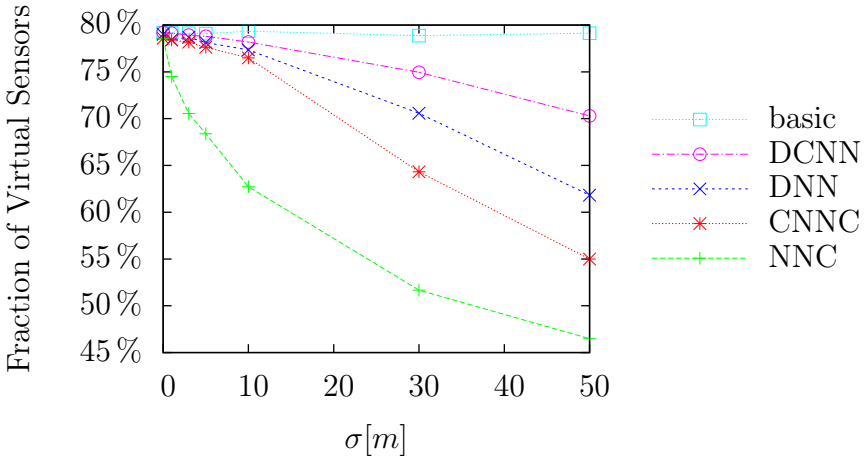


Figure 13.2.: Mean task coverage

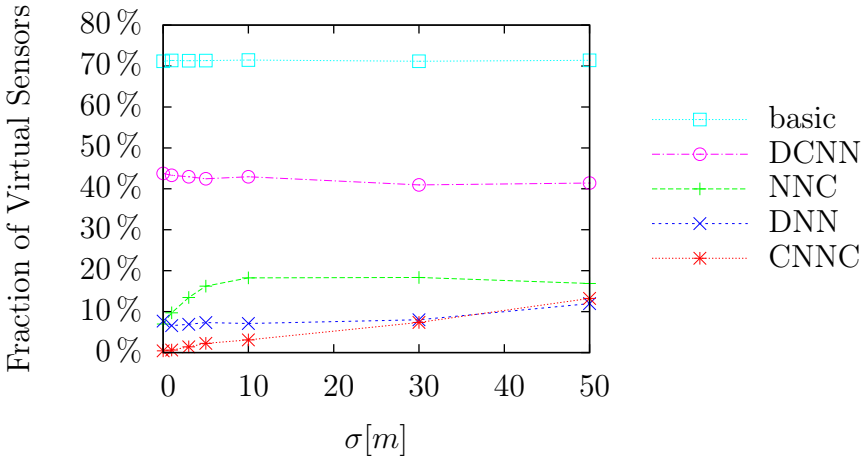


Figure 13.3.: Fraction of v-sensors where more than one reading per query was taken

clearly shows the worst task coverage. As explained in Section 11.3.1, there is a significant probability $p_\emptyset$ that no device in a coverage group will take a reading. This probability grows with the number of devices to pick from. This grows with δ_{limit}, which, in turn, is dependent on σ.

13.1.3. Efficiency of Query Execution

To analyze the efficiency of our algorithms, we use three performance metrics: The number of redundant readings, the sensor load, and the network load.

The first measure of efficiency we consider is the amount of redundant sensor readings. Redundant readings occur when there are more than k readings taken for a v-sensor in a single query execution. They increase the energy consumption on devices and put additional load on the network for transmitting these readings. Thus an algorithm taking fewer redundant readings is more efficient. Figure 13.3 shows the fraction of all v-sensors where more than one reading was taken for a query execution. As we set $k = 1$ in our simulations, each additional reading that was taken is redundant. The basic task execution algorithm shows redundant sensor readings at about 70% of v-sensors which is the worst case. Comparing this to the results for the task coverage metric in Figure 13.2 we see that about 9% of v-sensors were covered by exactly one device.

Of all our optimized approaches, DCNN clearly shows the largest fraction of v-sensors with redundant readings. The cause of this is the selection of the backoff time t_{DCNN}. As t_{DCNN} is directly proportional to the reported distance, devices with a similar reported distance will choose t_{DCNN} so that they will obtain a sensor reading before overhearing each others notification messages. Figure 13.4 shows an analysis of the amount of v-sensors with redundant readings for different backoff times in DCNN. Increasing the maximum backoff from $0.1\,s$ to $0.5\,s$ reduces the amount of v-sensors with redundant readings by up to 20%. Increasing the maximum backoff further is beneficial only in cases of low position uncertainty.

DNN shows an almost constant amount of v-sensors with redundant readings. Redundant readings are due to devices having different views on the number of devices n' and the distances $\delta(\overrightarrow{v}, \overrightarrow{n'_{i\,sens}})$ in their coverage group v_{NNC}, most likely caused by message collisions during the grouping phase.

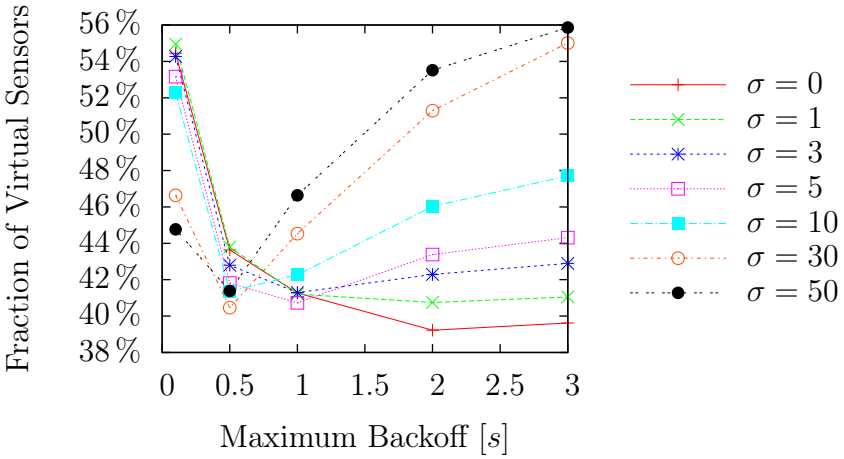


Figure 13.4.: Influence of maximum backoff time on the fraction of v-sensors with redundant readings in DCNN.

Redundant readings in NNC are caused by multiple devices choosing to take a reading for the same v-sensor due to the statistical independence of the devices decisions. Looking at CNNC we can see that coordination improves this problem, as the overall fraction of v-sensors with redundant readings is lower. Redundant readings in this case are caused by devices choosing similar backoff times. This is shown in detail in Figure 13.5. If the maximum backoff time is increased, the chance for two devices choosing a similar backoff time is drastically reduced, and, in turn, the number of redundant readings is also reduced. Increasing the maximum backoff time does, however, have a negative impact on the average distance of readings and, therefore, the quality of the result, as devices have more time to move away from their corresponding v-sensor.

Next, we look at the sensor load. Sensor load is defined as the number of sensor readings a mobile device has acquired during the whole simulation. It is used as an indicator for the amount of on-device resources that are used during query execution, e.g., energy for sampling the sensor and CPU time for processing the reading. Similar to the number of redundant readings, the basic task execution algorithm also shows the highest sensing load (see Figure 13.6). About 90% of devices had to acquire a reading 10 times. As we introduced 10 queries in each simulation and set $k = 1$, we can conclude that 90% of devices took part in every query. All of our approaches show a much lower sensing load. In our approaches, 90% of devices took part in only one query except for DCNN, where

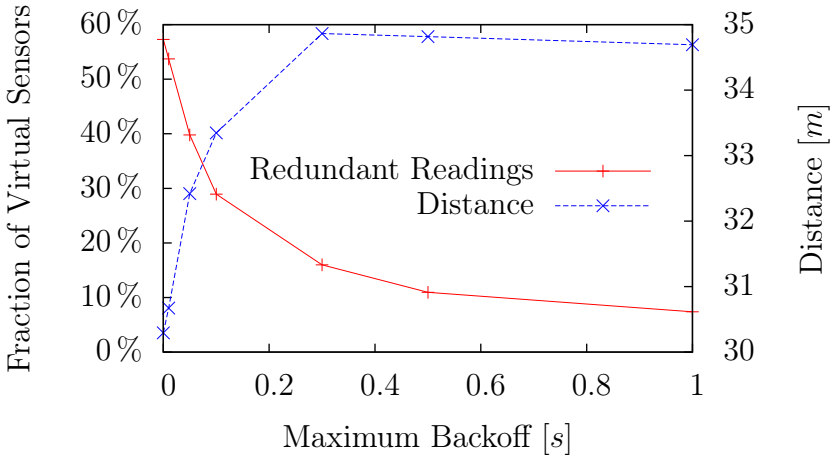


Figure 13.5.: Influence of maximum backoff time on distance and redundant readings in CNNC, $\sigma = 30\,m$. Left y-axis: Percentage of v-sensors with redundant readings. Right y-axis: Average Distance of readings to v-sensor locations.

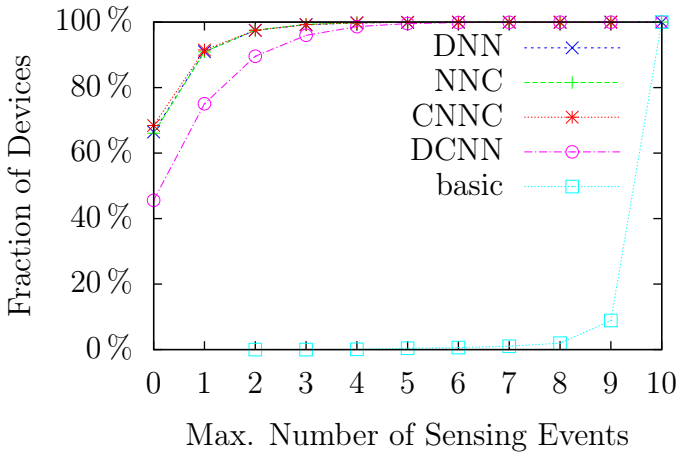


Figure 13.6.: Cumulated sensor load. Values are cumulated over all repetitions where $\sigma = 0\,m$. Plot shows fraction of devices where the number of sensing events $<$ maximum number of sensing events.

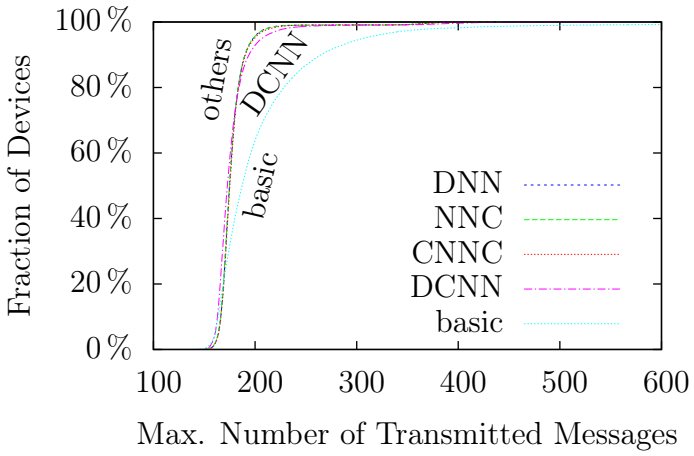


Figure 13.7.: Cumulated number of messages sent. Plot is truncated at 600 messages. Values are cumulated over all repetitions where $\sigma = 0\,m$. Plot shows fraction of devices where #messages < max. #messages.

95% of devices took part in three queries. The large amount of redundant sensor readings taken by DCNN does also show in the sensor load. Since more readings are being taken per query, each individual device also has to take more readings.

Last we look at the network load (see Figure 13.7). Network load is defined as the number of messages sent by each device over both the WiFi interface and the mobile data connection. Note that this includes all application messages as well as all protocol messages for, e.g., grouping and clustering. Values are cumulated over all simulations where $\sigma = 0\,m$. By comparison, the basic task execution algorithm causes the highest network load, where 90% of devices had to transport up to 250 messages each, as a reading is reported for every single device in the system. In each of our algorithms the network load was reduced to less than 200 messages for 90% of the devices, thus yielding a 20% increase in efficiency. Also note that under the basic task execution algorithm, devices had to transport up to 1300 messages, whereas in our approaches no device had to transport more than 550 messages. We can see that reducing the number of redundant readings does also pay off with respect to network load. The increased number of messages required for grouping and coordinating devices is easily compensated by the reduced effort for collecting and transmitting the resulting readings.

We performed a brief analysis of energy consumption for communication, using empirical energy models for communication [BBV09, XSK+10] and built-in

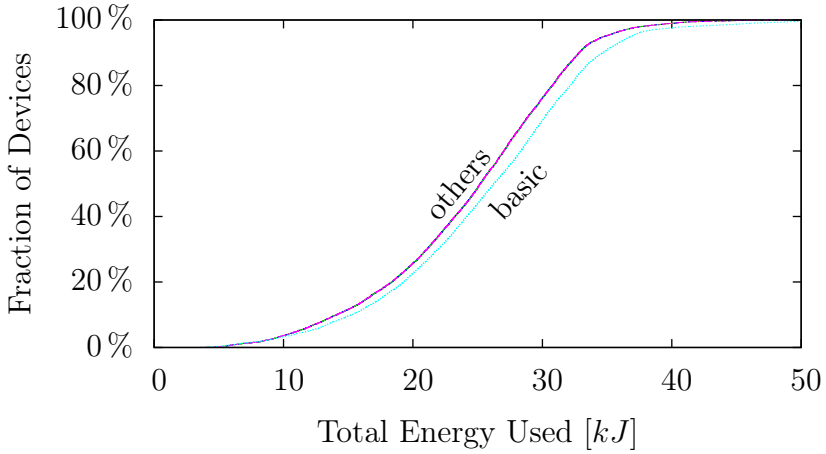


Figure 13.8.: Cumulated Energy Consumption for WiFi and mobile data. Plot shows fraction of devices where energy usage < total energy used. Image is truncated at $50\,kJ$.

sensors [PLL10]. We can see from Figure 13.8 that as with the network load, the added messages for grouping and coordinating devices are fully compensated by the reduced number of transmitted results. Currently, the energy savings resulting from our algorithm are relatively small. This is due to the ring-based routing algorithm that we use, which in itself produces duplicate messages. In the future, using a more efficient multi-hop-routing algorithm will allow our algorithms to reduce energy usage even further.

13.1.4. Discussion

Our evaluation showed that the distance-coordinated nearest-neighbor algorithm (DCNN) yields the same average distance as the basic algorithm and shows 88% task coverage in the worst case compared to the optimum value. However, the independent deterministic nearest-neighbor (DNN) performs better than DCNN under all efficiency metrics, e.g., up to 80% reduction of redundant readings over DCNN, while still providing 78% of the optimal task coverage in the worst case. The probabilistic approaches, based on nearest-neighbor candidates, show a significant reduction in quality, however, they show an improved efficiency over the corresponding distance-based approaches. In more detail, the evaluation showed that the proposed set of algorithms allows the user to define the trade-off between quality and efficiency by choosing a suitable algorithm from the proposed ones.

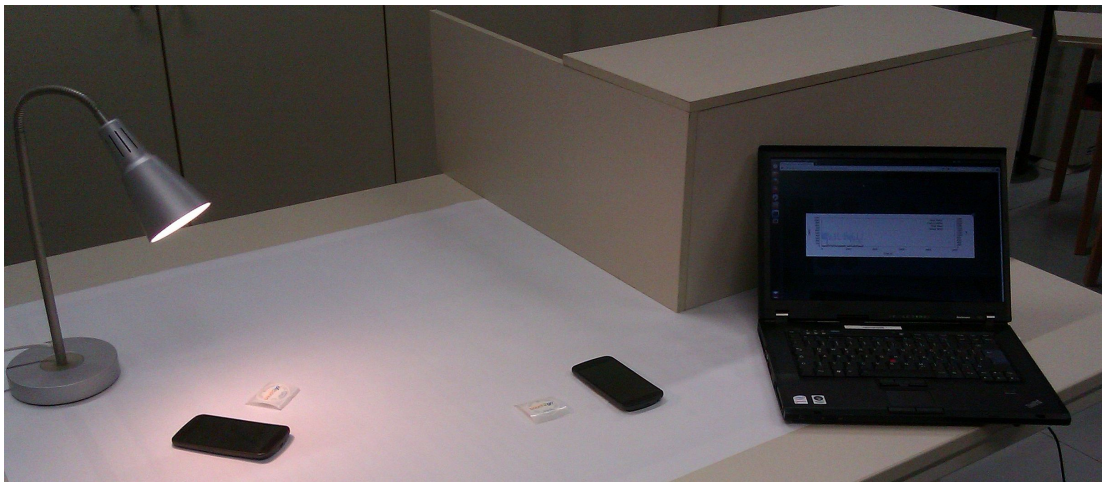


Figure 13.9.: Public Sensing Testbed

13.2. Global Optimization

In this section, we evaluate the performance of the global optimization. We first show our real-world proof-of-concept testbed for the quality-based v-sensor selection in the following section. To get more insight about the behavior of our system for large-scale tasks, we implemented both the quality-based and the adaptive v-sensor selection algorithms in a simulated environment as presented in Section 13.2.2. Results for the quality-based and adaptive v-sensor selection algorithms are discussed in Sections 13.2.3 and 13.2.4, respectively.

13.2.1. Testbed Evaluation

Our testbed, depicted in Figure 13.9, consists of a laptop, serving as the gateway and presenting a user interface for submitting queries and browsing obtained data. While it is possible to use our implementation to run a full-scale PS system, an experiment with hundreds of people participating is not feasible. Therefore, we select an evaluation scenario that can be handled by two people. Two smartphones are used for taking light intensity readings. Furthermore, we use a light source with predetermined movement to mimic the changing position of the sun over a day. Two v-sensors are placed directly under the initial position of the light source, two are placed at a distance, and another two are placed behind an obstacle shadowing the v-sensors from the light source. The smartphones then move among these v-sensors.

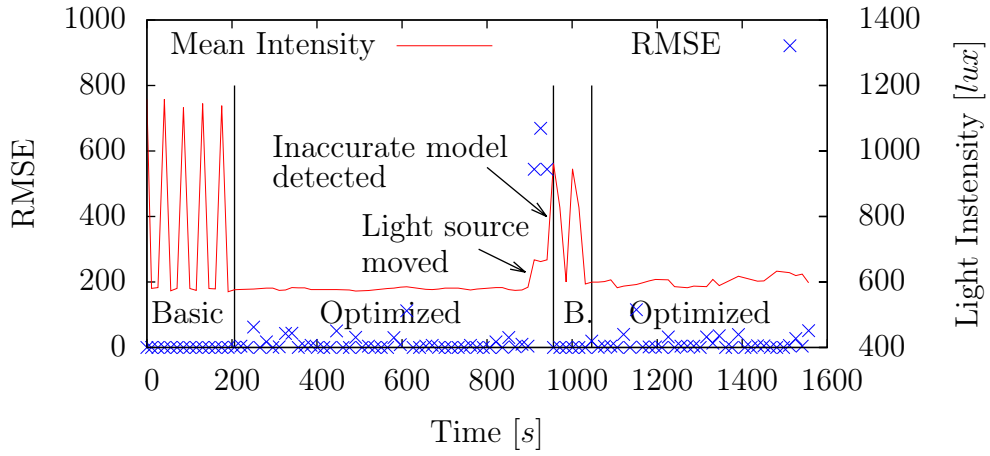


Figure 13.10.: Output of the testbed evaluation. 15 s sampling period.

Figure 13.10 shows the output of our testbed evaluation. DrOPS runs for roughly three minutes in a basic operation phase to learn a model of the light intensity at the v-sensors. Note that in the basic operation phase in each sampling period only readings for available v-sensors are included in the mean light intensity. Thus, device mobility leads to shifting availability causing readings to appear fluctuating. During the optimized operation phase, readings for all v-sensors are included, yielding a smooth mean. Up to 900 s, we do not change system conditions. Thus, the model remains accurate and readings are inferred with low error except for outliers. At 900 s the light source is moved, causing the existing model to become inaccurate. At this point, the error briefly goes up before MOCHA recognizes the model as inaccurate. After an additional 100 s, a new model is available and readings are again inferred with low error.

13.2.2. Simulation Setup

While our real-world testbed allows for insight into the operation of a model-driven PS system, it is far too small to provide information on the overall performance of the global optimization. Therefore, we extend our evaluation to a large-scale system using an implementation of the global optimization in a simulated environment implemented in the OMNeT++ network simulator.

To make our evaluation comparable to a real deployment, we use two real-world datasets as input: Ten days from the Intel Lab data set (LAB) [DGM+04] and

Parameter		LAB	LUCE
Error Threshold	$Q.QoS.T$	$1\,°C$	$1\,°C$
Max. Variance	$Q.QoS.\sigma^2_{max}$	0.1	0.1
Window Size	$Q.QoS.win$	10	10
Violations	$Q.QoS.violations$	3	4
Control Readings	$Q.QoS.ctrl$	3	1
Max. Learning Time	$Q.QoS.totalWaitTime$	1 hour	1 hour
Maximum Age	$Q.QoS.maxAge$	3 days	3 days
OLA Paramters	$Q.QoS.minRdg$	5	5
	$Q.QoS.minSens$	49	98
Number of Rounds	$Q.QoS.rounds$	1,2,3	1,2,3
Readings per v-sensor	$Q.QoS.k$	1	1

Table 13.2.: Quality parameters used in the simulation

three non-consecutive weeks of data from the Lausanne Urban Canopy Experiment (LUCE) [NBP+09]. Both data sets contain environmental readings, e.g., temperature reported by a large set of fixed sensors.

We generate queries by placing a v-sensor for temperature data at the position of every real sensor in each data set in order to generate a temperature map of the area. We again use a circular model for the area of each v-sensor, where the radius of the area is half the distance to the closest other v-sensor. The sampling period is adapted to match the interval at which data is provided by the data set. Other quality parameters used in the evaluation are shown in Table 13.2.

To generate device mobility for the LAB data, we place 200 mobile devices on an abstract representation of the lab's floor plan. They move around randomly along the available paths. For the LUCE data, we use CanuMobiSim [Ste] to generate random mobility traces for 400 devices on a road graph of the deployment area. We also run our algorithms in a static sensor network to analyze how close our approach is to optimum performance. Simulations run for 6 h each, with an offset between simulations increasing in steps of 3 h from the start of the data set.

To independently evaluate the properties of the global and local optimizations, we use an abstracted version of the local optimization in this simulation. We assume all positions to be accurate, as dealing with inaccurate positions is the task of the local optimization. Other aspects of the local optimization are not reflected in the simulation.

Note that inaccurate positions will not affect the performance of our system as long as mobile devices can correctly determine which v-sensor area they are located in, i.e., as long as effective readings can be assigned to the correct v-sensor. In our experiments this requires positions to be accurate to within 2.8 m for the LAB data and 4.8 m for the LUCE data. When position accuracy degrades further, the probability for assigning an effective reading to the wrong v-sensor increases. As any effective reading assigned to the wrong v-sensor is effectively a reading with a large error, this will reduce the benefit of the global optimization. On the one hand, the model constructed in the basic operation phase will no longer accurately represent the correlations of values in the underlying phenomenon, e.g., correlations may appear weaker than they are. This, in turn, increases the number of effective readings requested during the optimized operation phase. On the other hand, the increased error in effective readings collected during the optimized operation phase reduces the quality of inferred readings. While this can be detected by MOCHA, it will frequently cause the system to switch to a basic operation phase, even though the current model could still provide accurate inferred readings when given accurate effective readings.

For task dissemination, we use the 3G internet connection on every participating device as modeled by Frosch [Fro11]. Initially, participating devices register with the Task Dissemination component, which then forwards tasks via unicast to each registered device. Readings are transmitted via unicast to the Data Collection component.

We use empirical energy models for a 3G radio [BBV09] and built-in sensors [PLL10] to model the energy consumption of simulated mobile devices. Energy for positioning is not taken into account, since none of our algorithms change the number of position fixes taken by each device with respect to the basic sensing algorithm. Exploiting the additional potential for saving energy by reducing the number of position fixes is part of future work.

13.2.3. Evaluation of the Quality-Based V-Sensor Selection

We analyze the performance of global optimization using the quality-based v-sensor selection with regard to several metrics. *Effectiveness* is the fraction of tasks executed in an optimized operation phase, thus characterizing the time

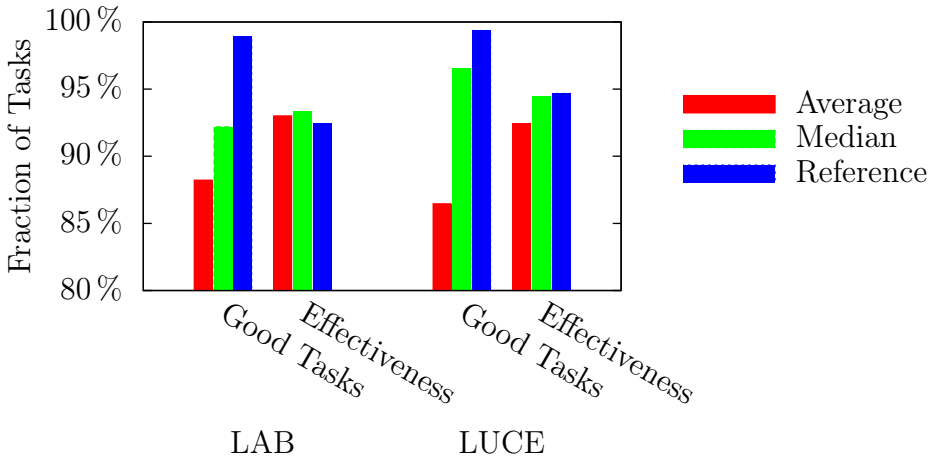


Figure 13.11.: Results for Good Tasks and Efficiency under the Quality-Based V-Sensor Selection. Average over all simulations, Median over all simulations and average over all reference simulations.

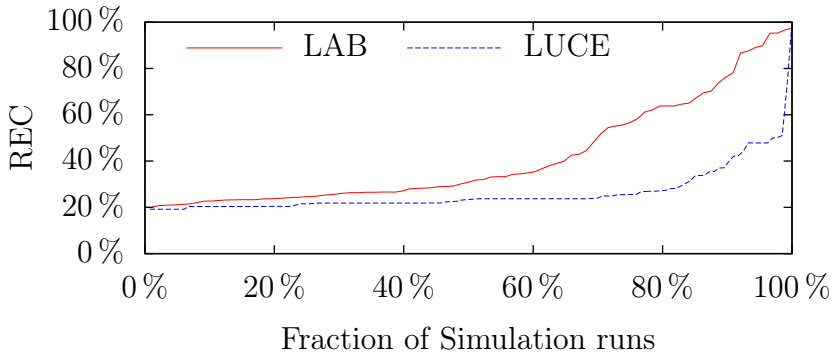


Figure 13.12.: Cumulated relative energy for communication and sensing.

spent where the model-driven sensing algorithm is used. *Good Tasks* is the fraction of tasks where QoS-constraints are met, thus characterizing the data quality an application can expect. In addition, we compute the *average duration of basic operation phases*. Finally, the *Relative Energy Consumption (REC)* classifies the energy consumption. As the absolute energy consumption varies greatly for different time offsets, e.g., due to a varying number of sensing tasks, the REC is computed by normalizing the energy consumption for each device in a simulation to the average energy consumption per device using the basic task execution algorithm for the same simulation parameters. Note that the energy drain is nearly uniformly distributed among all devices in a simulation. The maximum difference between devices in a simulation was 14.4 percentage points.

Effectiveness & Good Tasks

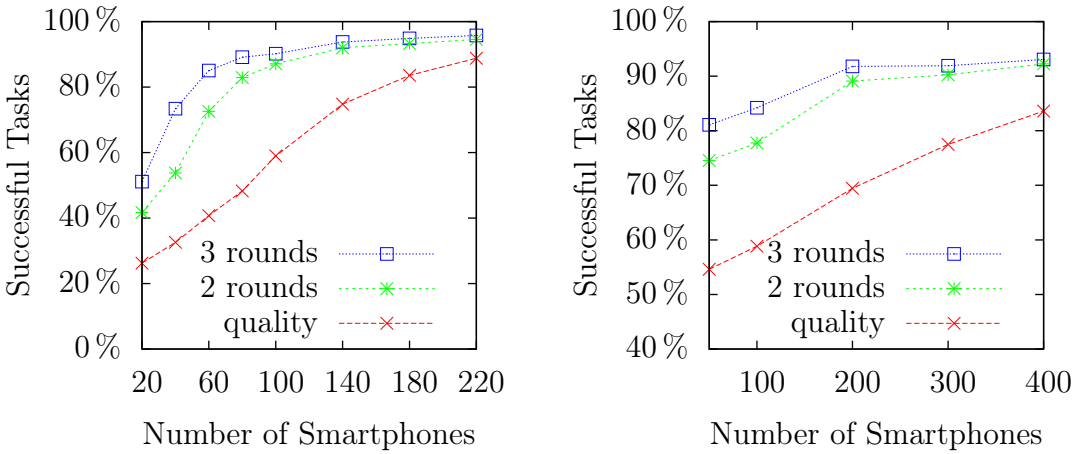
Figure 13.11 depicts the results for *effectiveness* and *good tasks*, shown as the average and the median of all simulations, and the average of the reference simulations. For *effectiveness*, results for both data sets are almost identical to the reference values from the sensor network whereas for *good tasks*, average values are 10 to 13 percentage points below the reference values. This indicates that the overall performance of our approach is basically similar to the reference system, except for a larger number of individual outliers caused by unavailable v-sensors both while learning a model and while optimizing execution. Note that for the LUCE data, in only 5 % of cases *good tasks* is below 50 %. For the Intel Lab data, in less than 5 % of cases the error threshold is violated by more than 0.2 $°C$. On average, basic operation phases last for 7.5 minutes for the Intel Lab data and 16 minutes for the LUCE data, showing that DrOPS can learn a model in a matter of minutes.

Relative Energy Consumption

Finally, we analyze the energy consumption of our system. The fact that the energy consumption for all devices in each simulation is nearly uniformly distributed indicates that the energy consumption is dominated by the reception of tasks, especially by the size of task messages, which in turn depends on the number of v-sensors selected for effective readings. We can see from Figure 13.12 that energy savings up to 80 % compared to the basic sensing algorithm are achieved in 3 % and 6 % of simulations (LAB data and LUCE data, respectively). On average, 59 % of energy is saved for the Intel Lab data, and 74 % for the LUCE data. In a few extreme cases, next to no energy is saved. This happens when effective readings for most or all v-sensors are requested even in an optimized operation phase, i.e., when no correlation could be identified.

13.2.4. Evaluation of the Adaptive V-Sensor Selection

We analyze the performance of the Adaptive V-Sensor Selection by taking a closer look at the operation during the optimized operation phase. We now use two more detailed metrics: *Successful Tasks* and *Empty Tasks*. The *Successful Tasks* metric is defined as the fraction of tasks in which the QoS-constraints are



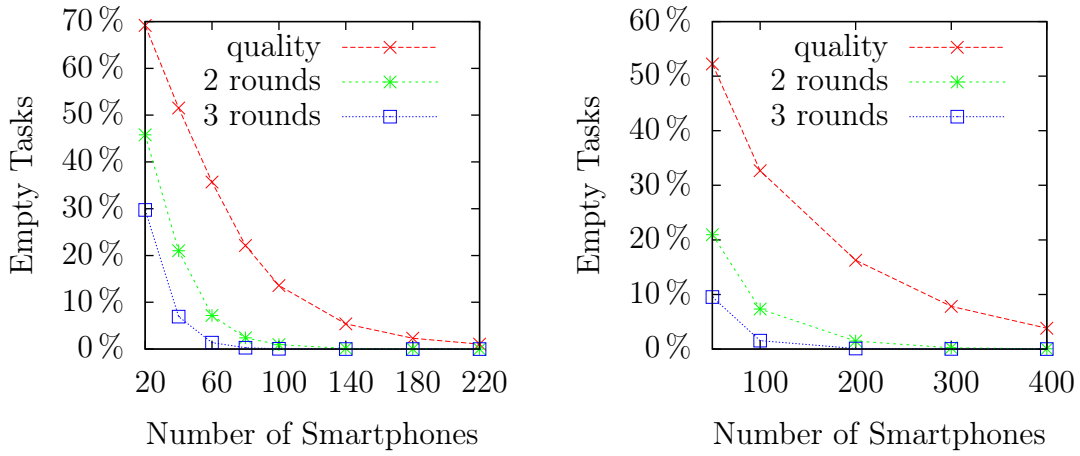
(a) LAB data (b) LUCE data

Figure 13.13.: Results for Successful Tasks metric. Fraction of tasks in which the QoS constraints are met.

met out of all tasks for which at least one effective reading was received, i.e., Good Tasks with at least one effective reading. The *Empty Tasks* metric denotes the fraction of tasks for which no effective readings were received at the Data Collection component, i.e., characterizing how an approach performs at finding available v-sensors. Furthermore, we analyze the energy consumption using the *Relative Energy Consumption* metric as before. We compare the performance of our adaptive approach (labeled "2 rounds", "3 rounds" in figures) to the basic task execution algorithm without optimization, i.e., $V_{\text{eff}} = V$ always, and the quality-based v-sensor selection for dense networks (labeled "quality").

Successful Tasks

Results for the *Successful Tasks* metric are depicted in Figure 13.13. Values are averaged over all simulation runs for each number of mobile devices. Under the quality-based selection, the number of successful tasks is good at just under 90 % for both datasets in a dense system, i.e., when using the maximum number of devices, but quickly degrades to under 60 % for 100 devices or less. Using our adaptive approach, the number of successful tasks increases to over 90 % in a dense system. Furthermore, it is far more robust to a decreasing number of devices. In the LUCE data, for example, using 3 rounds we can still provide 81%



(a) LAB data (b) LUCE data

Figure 13.14.: Results for Empty Tasks metric. Fraction of tasks for which no effective readings were obtained.

quality using 50 devices, whereas the quality-based approach requires 400 devices to match this performance.

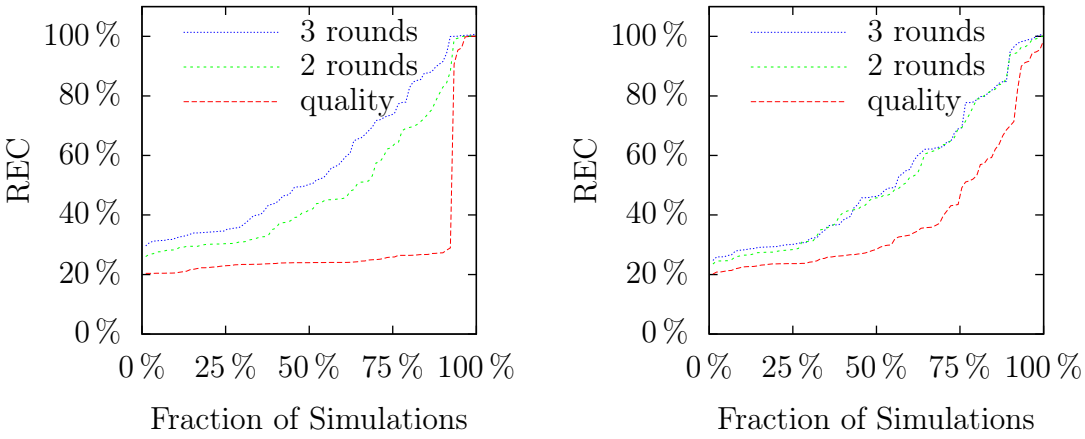
Empty Tasks

Next, we analyze results for the *empty tasks* metric. Evaluation results are depicted in Figure 13.14. Again, values are averaged over all simulation runs for each number of devices. Similar to the quality metric, the number of empty tasks under the quality-based approach drastically increases for a decreasing number of devices, while our extended algorithm is much more robust. Under the LAB data, for a single round the fraction of empty tasks increases to 5% for 140 devices whereas using 3 rounds, we can provide 7% of empty tasks with only 40 devices. For the LUCE data, the quality-based approach cannot match the fraction of empty tasks when using 3 rounds and at least 100 devices.

Relative Energy Consumption

We again use the *relative energy consumption (REC)* metric to characterize the energy consumption. Figures 13.15 and 13.16 show the cumulated average REC per simulation for the LAB data and LUCE data, respectively.

We see that additional communication for additional rounds increases the en-



(a) 60 devices

(b) 140 devices

Figure 13.15.: Cumulated relative energy consumption, LAB data

ergy consumption. The difference is greatest in a sparse setting, where few effective readings are collected in early rounds, i.e., most work is done in later rounds. In a denser setting, the difference diminishes, as later rounds add fewer readings, and thus less energy is used in later rounds. Note the sharp increase in REC for the quality-based selection in Figure 13.15a. For about 90 % of simulations, hardly any data is collected, i.e., only few available v-sensors are found and thus little energy is spent (cf. Figure 13.13a, 13.14a), whereas for the few cases where available v-sensors are found, only a weak model can be derived, i.e., V_{eff} is very large. As the round-based approach is better at finding available v-sensors, it does not exhibit this behavior. Using our round-based approach, we still can save up to 77% of energy (compared to 81% for quality-based selection in the LAB data, 80 % in the LUCE data). When the system contains at least as many devices as v-sensors, energy consumption is at most that of the basic task execution algorithm. When fewer devices are present, the round-based approach may use up to 6% more energy.

In summary, using our round-based adaptive v-sensor selection strategy will vastly improve the robustness of the system regarding a reduced number of participating devices by increasing the number of opportunities to gather data. Thus, it allows for operation in sparse networks. Even in a dense network, the quality of results returned to the application is improved. Furthermore, in a sparse

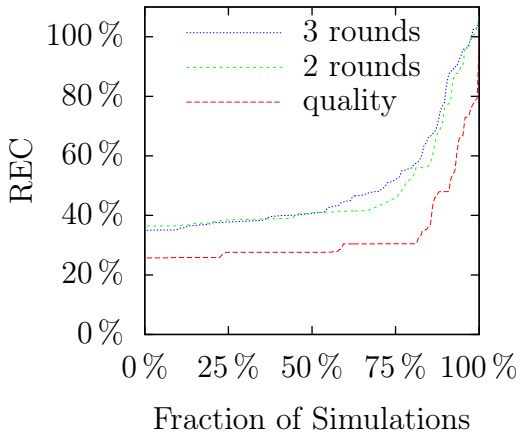


Figure 13.16.: Cumulated relative energy consumption, LUCE data, 100 devices

network much of the energy consumed under the quality-based selection goes to waste, as no data is obtained for that energy. In the adaptive approach, the increased energy consumption results in many more useful data readings and thus less wasted energy. Finally, robustness and result quality further increase when using additional rounds, while the energy cost for using additional rounds only increases when the additional rounds provide an actual benefit.

13.3. Summary

In this chapter we have demonstrated the effectiveness and efficiency of DrOPS, a system for the model-driven optimization of Public Sensing systems. We first presented the results for the *local optimization*. Our evaluations show that all of our algorithms noticeably decrease the workload of each mobile device, i.e., the number of readings a device has to acquire. Furthermore, a user can define the trade-off between quality and efficiency by choosing a suitable algorithm from the proposed ones.

Next, we presented the results from our evaluations for the *global optimization* in a dense network setting. Experiments in a real-world testbed show the general feasibility of our approach. In several simulated settings we analyzed the properties of our v-sensor selection approaches in more detail. Our evaluations show that using our online learning algorithm, we obtain optimization models in a matter of minutes on average. Furthermore, using the quality-based v-sensors

selection algorithm for optimizing the data acquisition, we can save up to 80 % of energy for communication and sensing and provide inferred readings for uncovered positions matching an error-bound of 1°C up to 100 % of the time.

Finally, we evaluated the performance of the global optimization in a sparse network setting. The evaluations show that the round-based adaptive v-sensor selection strategy successfully increases the robustness of the system regarding a reduced number of participating devices. At the same time we can still save up to 77 % of energy for communication and sensing.

14. Related Work

Research interest in Public Sensing has grown over the last years. There are numerous applications such as detecting potholes in roads [EGH+08], tracking mobile objects [LLSC11, AML+10] and people [SMBL12], estimating the density of crowds [WL11], and sharing a participants' current activity and social context [MLEC07] or gathering information on social events happening all over a city [MPL+11]. However, as our focus is on environmental monitoring, we limit our analysis to related work from this area.

In the following, we begin by presenting a number of approaches addressing general systems challenges, before moving on to the problem of selecting a subset of participating devices and the application of data models in PS. Furthermore, we present other approaches for improving the energy efficiency of PS. Finally, we present related work concerning privacy-issues in PS as well as other optimizations for data acquisition systems geared towards static and actuated sensor networks.

Public Sensing Systems for Environmental Monitoring Several prototype systems demonstrated the feasibility of using PS for, e.g., monitoring of CO-levels [SM08], air pollution [MPLM11, BEH+06], noise pollution [MSN+09, SBS+11, DSJ13], and Seismology [Kra14]. Furthermore, end-user studies by Willett et al. [WAK+10] and Kanjo et al. [KBRL09] showed how people benefit from the information provided by PS. However, these systems are each built for a single specific task and require energy-intensive constant sampling by all participating devices.

To enable more flexible opportunistic PS systems, Campbell et al. [CEL+08] and Kansal et al. [KNLZ07] identified several systems challenges. These include the heterogeneity of sensor hardware on mobile devices, identifying appropriate sensing context, e.g., whether a sensor is currently exposed to the phenomenon to be observed, varying coverage of points of interest by mobile devices, and the

need for query languages to configure the system for heterogeneous sensing tasks.

To address the challenge of heterogeneous sensing hardware and device context, Eisenman et al. let neighboring devices share their sensing hardware, so that all devices gain access to data for which they do not possess the correct sensing hardware or for which they are in the wrong context to obtain a data reading [ELC08]. In contrast, MobGeoSen focuses on augmenting participating devices with additional external sensing hardware [KBP+08], giving all devices access to uniform hardware that is always exposed.

Several other systems consider the problem of how to submit heterogeneous tasks to the system. Campbell et al. [CEL+06] specified several types of v-sensor abstractions after which our v-sensors are modeled. To specify relevant data using simple spatial, temporal and contextual predicates, formal notations [DZP+14] and SQL-like query languages [HBZ+06, RB07] have been proposed. For a more flexible solution, other approaches present APIs for mobile code that can evaluate arbitrary predicates on mobile devices [DMP+10, BL12, RB07].

Selection of Participating Devices The systems presented so far offer predicates to obtain readings only from devices that will provide relevant data. Several other works deal with the problem of implementing such predicates. Reddy et. al. [RSB+08, RSB+09, RES10] developed a reputation system for selecting the most suitable devices for long-running tasks. The reputation is built over a long time (e.g., several days) from geographical and temporal availability, derived from mobility information and from past participation in sensing tasks. However, the purpose of the system is to aid a human operator in device selection. While the work of Zhao et al. [ZMLZ13] and Zhang et al. [ZXWC14] extended this idea to an automatic selection of devices for probabilistic coverage, they require the same lengthy training phase. Boutsis and Kalogeraki [BK14] pursue a similar approach of selecting participants for task execution in a participatory PS system, so that the chances of task execution are maximized and the cost for compensating participants is minimized. Kravets et al. [KAC+13] propose a probabilistic selection of devices similar to our local optimization, but with the goal of selecting devices uniformly distributed in space moving in a dense crowd of people, rather than close to specific locations. All of these approaches are device-centric, i.e., select devices for continuous sensing, potentially gathering a large amount

of data outside the area of interest. Furthermore, the lengthy training phases required for these approaches make them unsuitable to address the challenge energy consumption in flexible PS systems.

In our DrOPS system, device selection is location-centric. Closest to this, Lu et al. proposed a first method for the location-centric selection of devices at a single v-sensor [LLEC09]. However, their aim is to select devices with maximum dwell time in the area of a v-sensor rather than minimizing the distance of readings from the v-sensor. Kansal and Zhao [KZ07] propose to compensate for an inaccurate or unavailable positioning system by extracting proximity information from the data itself. Their approach is geared towards clustering images from smartphone cameras by location through content matching and is thus not easily applicable to the problem of environmental monitoring.

Applications of Data Models in Public Sensing The use of data-driven models in PS has been considered under several aspects. Rana et al. [RCK+10] and Wisniewski et al. [WDMCM13] use interpolation models specifically designed for noise measurements to compute inferred readings for times and locations where no device was available to take an effective reading. Focusing on other environmental variables, Mendez et al. and Sluiter showed that multivariate Gaussian distributions are well suited for computing inferred readings [MLR13, Slu08]. Furthermore, they use multivariate Gaussian distributions to detect outliers in measurements [ML12b] and propose an online-approach to optimize the placement a set of v-sensors to provide the most informative input data [ML12a, VLML14]. None of these approaches considers using the model for actively reducing the effort for data acquisition.

Little work has been done regarding online learning of models. The work of Lane et al. [LLEC08] and Miluzzo et al. [MCR+10] deals with the construction of models for event classification, e.g., speaker recognition or classification of social encounters. However, their goal is simply to maximize the amount of input data to build more accurate classifiers without regards for training duration.

Energy-Optimizations for Public Sensing Several works consider the energy consumption in PS regarding individual devices, effort for communication, and the system as a whole.

A common approach to reducing the energy consumption on individual devices is duty-cycling of sensors and processing units. Lu et al. [LYL+10] and Rachuri et al. [RMM12] adjust the sampling rate of sensors based on device context and the history of interesting observations. As a reduced sampling rate may lead to missed observations, Kim et al. [KKES10] and Shin et al. [STKC09] focus on continuous monitoring using energy-efficient sensors, e.g., to determine device context, and enable energy-intensive hardware only when more detailed data from other sensors is required. In a hardware-based solution, Priyantha et al. [PLL10, PLL11] propose to use an energy-efficient, dedicated coprocessor to continuously analyze all sensor data. However, this approach requires that smartphones are modified specially for PS tasks, which would exclude a significant part of the devices already deployed from participating in the system.

To save energy for the transmission of readings, Baier et al. [BDR13b] restrict communication to piggybacking PS messages on other data traffic, thus explicitly amortizing the energy-consumption for putting the communication interface in a transmit-state to other applications. Taking points of interest into account, Baier et al. also focused on offloading communication from energy-intensive mobile networks to more energy-efficient WiFi connections [BDR12b] and on limiting communication to devices that are in the correct location to execute a sensing task [BDR12a, BDR13a]. Devices are identified based on a simple mobility model assuming a maximum speed. Xiong et al. [XZWC14] propose a similar approach, additionally taking call times and call locations into account to build the mobility model and to predict when messages can be transmitted. However, their approach focuses on continuous sensing and thus does not respect potential points of interest. Furthermore, none of these approaches consider the potential for optimization by taking user-specified quality information into account.

Another challenge is the availability of the communication channel. Halo [ELC10] and CarTel [HBZ+06] lift the assumption of a continuously available internet connection by applying techniques from delay-tolerant networking (DTN). By substituting DTN for the cellular network connections, they improve both the amount of data delivered to the consumer when a device is in an area with low network coverage and the energy consumption for communication, even when a device has internet access.

Focusing on the overall energy consumption of the PS system rather than on

individual devices leads to approaches for scheduled device selection. Weinschrott et al. [WDR09, WDR10], Sheng et al. [STZ12], and Eberle et al. [EYA13] solve a problem similar to our local optimization, where each device is assigned a schedule when to take a reading. However, their goal is complete spatial k-coverage, without regard for specific points of interest. Weinschrott et al. also considered the case of obtaining sensor data at specific locations [WDR09]. In this case, however, locations were represented by fixed sensors that can be read via RFID. Thus, the position of devices was not taken into account.

Privacy concerns in Public Sensing A general concern in the use of large-scale data acquisition systems is privacy, as participating in PS should not reveal adverse information about participants. Thus, several approaches focus on hiding the identity of participants through mixing of readings either in-network [CKK+08] or by a trusted third party [DBF12]. As the presence of readings in an area may be sufficient to identify the source, other works focus on blurring the location of a reading or on providing k-anonymity, i.e., plausible deniability for each participant [KTC+08]. For cases where a participant can not trust any other entity of the system, Drosatos et al. propose the use of homomorphic encryption to compute aggregated results out of individual measurements without ever revealing individual information about measurements [DEA+12]. Pidcock et al. additionally considered the privacy requirements of subjects monitored using PS, e.g., non-participating people that are in an area of interest. They developed a notification system to warn bystanders of an impending data collection campaign [PSHG11].

Optimizations in Sensor Networks Using models for optimizing data acquisition has a long history in static sensor networks.

Device-centric approaches for optimization use time-series models. The data sink computes a time-series model of observations to predict future readings. These predictions are verified by sensor nodes by either performing the prediction on the sensor nodes in parallel or by transmitting predictions from the sink to the sensors. Sensor nodes then only need to spend energy for communication if the measured value deviates from the prediction [LGS06, SR06]. However, these models are location-centric and can only be applied when the location of

each device is fixed. This method transfers to the PS setting for truly device-centric models only. For example, Musolesi et al. used a Markov model to reduce the number of message transmissions required for keeping information about the current state of a participating device up to date on a gateway [MPF+10].

Other approaches for optimization focus on identifying points of interest, i.e., where sensors should be deployed, or, given an existing deployment, which subset of sensor nodes should perform the data acquisition work. Assuming a known sensing range, Abrams et al. [AGP04] propose an algorithm to select varying subsets of sensor nodes for full spatial coverage in a round-robin fashion, thus distributing the load of a sensing task. Assuming an infinite sensing range along the line-of-sight, Gonzalez-Banos et al. [GB01] maximize spatial coverage with a minimum number of devices. However, these approaches require control over the placement of sensor nodes (or at least knowledge thereof), and are thus not applicable to PS without a sensor network abstraction layer. Furthermore, they maximize spatial coverage with no regard for actual points of interest.

Most relevant to PS are approaches for identifying most interesting positions using spatial models. Das and Kempe present metrics for selecting sensors so that the error in predicted aggregate functions is minimized [DK08]. In the BBQ system, Deshpande et al. use a multivariate Gaussian distrubution, extended to a spatio-temporal model using markovian models, to compute inferred readings and reduce the number of effective readings taken in a sensor network [DGM+04]. However, they assume the multivariate Gaussian distribution to be available a priori and thus do not address the challenge of learning and monitoring said model. The GREEDY algorithm used in our approach was a follow-up result of this work [GKS05, Kra08].

Works in actuated sensing [BSBK09, SFRJ09, BKS13] focus on informative path planning for networks of mobile devices. The main difference to PS is that robots are used as mobile sensors. Thus, their movement is controlled by the system. These systems achieve the benefit of using fewer sensors to cover a large area at the additional cost of having to deploy specialized mobile sensor nodes.

15. Summary & Future Work

Public Sensing is a new paradigm for sensor data acquisition. In Public Sensing, data is gathered using commodity smartphones, containing a wealth of sensors, carried and maintained by participants, i.e., ordinary people willing to share their spare resources with the public. It improves over the state-of-the-art approach of using fixed sensor networks as data is provided without the administrative problem of having to deploy sensor nodes and the associated upfront-monetary cost. Furthermore, it allows for a much greater service area as participants move about during their daily life. Thus, Public Sensing allows for the construction of flexible, large-scale data acquisition systems that can be spontaneously queried. This easy access to data enables application developers to shift their focus from obtaining input data to the actual development of novel applications. Furthermore, it can improve peoples' everyday quality of life by increasing their awareness of environmental problems and thus enabling people to address these problems.

In building such systems, we face several challenges. As Public Sensing relies on volunteers, data collection should be performed opportunistically to avoid driving participants from the system. Furthermore, the system must conserve the shared resources, e.g., energy, as much as possible. Another challenge is the mobility of participants, which makes querying the system and using the resulting data more complicated. We addressed these challenges in the context of two applications: Creating indoor maps and monitoring of large-scale spatially distributed environmental phenomena.

In the area of indoor mapping, we presented the MapGENIE system that automatically infers indoor maps from pedestrian traces and structural building information. First, we presented a trace-based mapping approach, which is then improved by structural knowledge encoded in a formal indoor grammar. To analyze its performance, we conducted a large-scale experiment by collecting pedestrian indoor traces. The results show that MapGENIE enhances the indoor

mapping process significantly, i.e., producing detailed indoor maps from only a small set of traces.

Furthermore, we presented a quality model for indoor maps that classifies parts of the map into accurately mapped and inaccurately mapped, according to the amount of information that is available about these areas. Using a combination of an accurate but energy-intensive inertial positioning system (IMU) and a less accurate but more energy-efficient WiFi Position Tracking system (WPT), we showed that up to 15 % of energy can be saved when the trace recording is disabled in accurately mapped areas, i.e., WPT is used instead of IMU.

In the area of environmental monitoring, we presented the DrOPS system. To provide applications with a mobility-independent view on the system, we introduced our v-sensor concept as a mobility-independent abstraction and showed how the deployment of a static sensor network can be converted into queries for a Public Sensing system. In addition, we defined Request-Driven Execution, where participating devices acquire data only when it is actually requested by an application.

To further improve the efficiency of the data acquisition process, we presented our local optimization. To this effect we developed four coordination algorithms to select mobile devices for executing a query in the face of uncertain location information. The selection of devices is done in a way that the quality of the result is maximized while the necessary effort w.r.t. redundant readings, number of readings, and communication, is minimized. We analyzed the performance of our algorithms subject to metrics for result quality and efficiency. The evaluation showed that the proposed set of algorithms allows the user to trade an increase in quality for a decrease in efficiency by choosing a suitable algorithm.

Moreover, we showed how to transfer optimization techniques from static sensor networks to the Public Sensing domain. Our global optimization uses a model-driven sensing approach based on multivariate Gaussian distributions. On the one hand, we can compute inferred readings to compensate for missing readings due to unavailable v-sensors. On the other hand, we can select the most interesting v-sensors to observe. By not requesting effective data readings for all other v-sensors, we again reduce the effort required from participating devices, and thus reduce the energy consumption. Furthermore, we introduced OLA, an online learning algorithm to learn multivariate Gaussian distributions over short time

periods, and MOCHA, an online model validity check algorithm to determine whether a given multivariate Gaussian distribution fits current sensor readings. Our evaluations show that we obtain optimization models in a matter of minutes on average. Using the model-driven approach for optimizing the data acquisition, we can save up to 80 % of energy for communication and sensing and provide inferred readings for uncovered positions matching an error-bound of $1°C$ up to 100 % of the time.

We also extended the global optimization with an adaptive extension to enable operation in sparse networks, where most v-sensors are unavailable. In model-driven sensing, a minimum number of effective readings is required to provide sufficient result quality. With our round-based adaptive v-sensor selection algorithm, we can find the required readings even when the majority of v-sensors is unavailable. Our evaluations show that we can enable the system to work with a greatly reduced number of smartphones and that result quality is improved by up to 41 percentage points compared to the non-adaptive algorithm. Furthermore, we can save up to 81 % of energy for sensing and communication while providing inferred readings matching an error bound of 1 °C up to 96 % of the time.

15.1. Future Work

Apart from direct optimizations of the algorithms presented in this work, our concepts may be extended to include additional information in their decisions and to provide additional functionality as follows.

For our grammar-based mapping approach we require an input grammar provided by an expert to derive a grammar-based indoor model. When this system is widely used, manual specification of such grammars becomes infeasible. Thus, MapGenie could be extended to automatically derive and improve the grammar from trace information. The ultimate goal would be a closed-loop system in which the traces feed the grammar derivation, while the knowledge stored in the grammar supports the recording of traces.

As MapGenie relies on inertial sensors for trace data acquisition, it obtains information only about the exact positions that participants walked through. Through the use of image data from smartphone cameras or wearable devices, e.g., Google Glass, information about the immediate surroundings of a participant

can be included. On the one hand, using image data would enable MapGenie to distinguish between impassable areas due to obstacles such as furniture and actual walls. On the other hand, this would decrease the number of independent observations required to complete a floor plan, as each individual observation now covers a larger area.

Furthermore, with the availability of consumer devices with built-in 3D sensors, e.g., Google Tango Phone [pro15], a full 3D model of the interior can be provided. MapGenies' energy-efficient approach can be adapted to improve the efficiency of the 3D mapping process. First, the quality model needs to be modified for 3D observations. While a single observation may be sufficient to detect the presence of an object, this observation is incomplete with regard to information about the object itself. For instance, a single object contains information about only one side of the object, which may be insufficient to determine the type of the object. Thus, multiple observations from different directions may be required. Second, the Scheduler must consider additional information such as the field of view of the 3D sensor, which may be independent from the walking direction of the participant, and the resource-intensive task of processing image data and 3D data.

Using our quality model for floor plans, MapGenie can optimize the trace data acquisition process over time. MapGenie can be extended with algorithms for distributed coordination to also provide energy-efficient operation for crowds of participants moving together along a path, recording their movement all at the same time. As most people in a crowd will provide similar information, gathering observations from a subset of participating devices will likely be sufficient. The coordination algorithms would then limit the set of devices that record trace data, allowing all other devices to switch to the more efficient WPT.

In the area of environmental monitoring, our local optimization approach may be extended to consider additional information in its decision. One example of such information is resource information, e.g., the current battery level of participating devices. While we assume that mobility is high enough so that in every sampling period, another device fulfills the role of a v-sensor, devices may still have to sample several times for different v-sensors in subsequent sampling periods, when the density of selected (interesting) v-sensors is high. When taking resource information into account, the v-sensor selection can identify such cases

and temporarily select a larger set of less interesting v-sensors located elsewhere to reduce the load of devices in the vicinity of the most interesting v-sensors.

Furthermore, the selection of v-sensors may be optimized for better efficiency of positioning. For example, devices can use low-accuracy low-power positioning systems for a preliminary position fix [GLC+08]. If they are well out of the vicinity of any selected v-sensor, even given the uncertainty of their position, they do not need to sample a more energy-intensive positioning system, unless another smartphone app requires a position fix with higher accuracy. Thus, the overall energy consumption for positioning caused by Public Sensing may be reduced by, on the one hand, selecting v-sensors for effective readings so that the number of devices that can stop query execution after a low-power position fix is maximized. On the other hand, when areas emerge where participants already use other smartphone apps that require high-accuracy positions, v-sensors in these areas may be preferred, as in such areas, high-accuracy positions are available at no additional cost.

Privacy is an issue for all Public Sensing systems. As DrOPS is designed to be privacy-friendly, i.e., to require as little information about participants as possible, it may be extended with standard privacy techniques such as k-anonymity. In contrast, MapGenie requires full disclosure of trace information from participants. While participants might choose to improve their privacy through, e.g., mixing of trajectories, the question whether this has an impact on our trace correction algorithms, and thus the accuracy of the generated floor plan, remains an open problem.

Publications

The following publications of the author of this dissertation contribute major parts to this dissertation:

[PBD+14] Damian Philipp, Patrick Baier, Christoph Dibak, Frank Dürr, Kurt Rothermel, Susanne Becker, Michael Peter, and Dieter Fritsch. MapGENIE: Grammar-enhanced Indoor Map Construction from Crowdsourced Data. In *Proceedings of the 12th IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 139–147, Budapest, Hungary, March 2014. IEEE Computer Society Conference Publishing Services.

[PDR11] Damian Philipp, Frank Dürr, and Kurt Rothermel. A Sensor Network Abstraction for Flexible Public Sensing Systems. In *Proceedings of the 8th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 460–469, Valencia, Spain, October 2011.

[PSA+13] Damian Philipp, Jaroslaw Stachowiak, Patrick Alt, Frank Dürr, and Kurt Rothermel. DrOPS: Model-Driven Optimization for Public Sensing Systems. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 184–191, San Diego, CA, USA, March 2013. IEEE Computer Society.

[PSDR13] Damian Philipp, Jaroslaw Stachowiak, Frank Dürr, and Kurt Rothermel. Model-Driven Public Sensing in Sparse Networks. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, LNCS, pages 17–29, Tokyo, Japan, December 2013. Springer.

The following publications of the author of this dissertation have influenced parts of this dissertation:

[BPDR14] Patrick Baier, Damian Philipp, Frank Dürr, and Kurt Rothermel. Quality-based Adaptive Positioning for Energy-Efficient Indoor Mapping. Technischer Bericht Informatik 2014/06, Universität Stuttgart, Germany, Fakultät Informatik, Elektrotechnik und Informationstechnik, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, November 2014.

[BPF+13] Susanne Becker, Michael Peter, Dieter Fritsch, Damian Philipp, Patrick Baier, and Christoph Dibak. Combined grammar for the modeling of building interiors. In *Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, 2013.

[HFP12] Klaus Herrmann, Daniel Fischer, and Damian Philipp. Resource Management for Public Sensing. In *Activity Context Representation: Techniques and Languages*, pages 32–39, Toronto, Canada, July 2012. Online.

Supervised Student Theses

[Alt12] Patrick Alt. Integration modellbasierter Erfassungsmethoden in ein Public-Sensing-Testbett. Bachelorarbeit, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, 2012.

[BRW14] Alexej Bonet, Benjamin Rau, and Constantin Weißer. Leistungsverbesserung und Erweiterung eines Systems zur Innenraummodellierung. Projekt-INF, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, June 2014.

[Dib13] Christoph Dibak. Erfassung von Innenraummodellen mittels Smartphones. Diplomarbeit, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, November 2013.

[Fro11] Thorsten Frosch. UMTS-Implementierung für OMNeT++. Studienarbeit, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, April 2011.

[Neu13] Nikolai Neugebauer. Mechanismen zur Optimierung der Effizienz von Public-Sensing-Systemen. Bachelorarbeit, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, November 2013.

[Sta11] Jaroslaw Stachowiak. Optimized Acquisition of Spatially Distributed Phenomena in Public Sensing Systems. Masterarbeit, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, October 2011.

[Til12] Maren Tilk. Partitionierung von Modellen für räumlich verteilte Umgebungsphänomene. Bachelorarbeit, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, 2012.

[TMA12] Maren Tilk, Steffen Maaß, and Patrick Alt. Entwicklung eines Realwet-Testbeds für Public-Sensing-Systeme auf Basis von Android. Projekt-INF, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Verteilte Systeme, 2012.

Bibliography

[AAB+07] Tarek Abdelzaher, Yaw Anokwa, Peter Boda, Jeff Burke, Deborah Estrin, Leonidas Guibas, Aman Kansal, Samuel Madden, and Jim Reich. Mobiscopes for human spaces. *IEEE Pervasive Computing*, 6(2):20–29, 2007.

[ACMR11] C. Audras, A. Comport, M. Meilland, and P. Rives. Real-time dense appearance-based SLAM for RGB-D sensors. In *Proceedings of the Australasian Confonference on Robotics and Automation*, 2011.

[AGP04] Zoë Abrams, Ashish Goel, and Serge Plotkin. Set K-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 424–432, New York, NY, USA, 2004. ACM.

[AH96] Ian F. Akyildiz and Joseph S.M. Ho. On location management for personal communications networks. *IEEE Commun. Mag.*, 34(9):138–145, Sep 1996.

[AML+10] Gahng-Seop Ahn, Mirco Musolesi, Hong Lu, Reza Olfati-Saber, and Andrew Campbell. Metrotrack: Predictive tracking of mobile events using mobile phones. In Rajmohan Rajaraman, Thomas Moscibroda, Adam Dunkels, and Anna Scaglione, editors, *Distributed Computing in Sensor Systems*, volume 6131 of *Lecture Notes in Computer Science*, pages 230–243. Springer Berlin / Heidelberg, 2010.

[AR12] M. Angermann and P. Robertson. FootSLAM: Pedestrian simultaneous localization and mapping without exteroceptive sensors

– hitchhiking on human perception and cognition. *Proc. IEEE*, 100:1840–1848, May 2012.

[AY12] M. Alzantot and M. Youssef. CrowdInside: Automatic construction of indoor floorplans. In *Proceedings of the Conference for Advances in Geographic Information Systems (SIGSPATIAL)*, 2012.

[BBV09] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 280–293, New York, NY, USA, 2009. ACM.

[BDR12a] Patrick Baier, Frank Dürr, and Kurt Rothermel. PSense: Reducing Energy Consumption in Public Sensing Systems. In *Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 136–143. IEEE, March 2012.

[BDR12b] Patrick Baier, Frank Dürr, and Kurt Rothermel. TOMP: Opportunistic Traffic Offloading Using Movement Predictions. In *Proceedings of the 37th IEEE Conference on Local Computer Networks (LCN)*, pages 50–58, Clearwater, October 2012. IEEE Computer Society.

[BDR13a] Patrick Baier, Frank Dürr, and Kurt Rothermel. Efficient Distribution of Sensing Queries in Public Sensing Systems. In *Proceedings of the 10th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 272–280, Hangzhou, China, October 2013. IEEE Computer Society.

[BDR13b] Patrick Baier, Frank Dürr, and Kurt Rothermel. Opportunistic Position Update Protocols for Mobile Devices. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pages 787–796. ACM, 2013.

[BEH+06] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory Sensing. In *Work-*

shop on World-Sensor-Web: Mobile Device Centric Sensor Networks and Applications (WSW), pages 117–134, 2006.

[BH09] Susanne Becker and Norbert Haala. Grammar supported facade reconstruction from mobile LiDAR mapping. In *Proceedings of the Workshop on Object Extraction for 3D City Models, Road Databases and Traffic Monitoring*, 2009.

[BHF08] Susanne Becker, Norbert Haala, and Dieter Fritsch. Combined knowledge propagation for facade reconstruction. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII, Part B5, Commission 5*, page 423ff., 2008.

[BK14] I. Boutsis and V. Kalogeraki. On task assignment for real-time reliable crowdsourcing. In *34th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 1–10, June 2014.

[BKS13] Jonathan Binney, Andreas Krause, and Gaurav S. Sukhatme. Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research*, 32(8):873–888, 2013.

[BL12] Niels Brouwers and Koen Langendoen. Pogo, a middleware for mobile phone sensing. In *Proceedings of the 13th International Middleware Conference (Middleware)*, pages 21–40, New York, NY, USA, 2012. Springer-Verlag New York, Inc.

[BP99] Paramvir Bahl and Venkata N. Padmanabhan. User location and tracking in an in-building radio network. Technical Report MSR-TR-99-12, Microsoft Research, February 1999.

[BR11] L. Bruno and P. Robertson. WiSLAM: Improving FootSLAM with WiFi. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, 2011.

[Bro14] Broadcom. WICED Sense. `http://www.broadcom.com/products/wiced/sense/`, 2014. Accessed: 01.09.2014.

[BSBK09] D. Budzik, A. Singh, M.A. Batalin, and W.J. Kaiser. Multi-scale Sensing with Stochastic Modeling. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4637–4643, 2009.

[BWDR11] Patrick Baier, Harald Weinschrott, Frank Dürr, and Kurt Rothermel. MapCorrect: automatic correction and validation of road maps using public sensing. In *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks (LCN)*, Bonn, Germany, October 2011.

[BZF12] Michael Bosse, Robert Zlot, and P. Flick. Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE Trans. Robotics*, 28:1104–1119, 2012.

[CCR10] I. Constandache, R.R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *Proceedings of the 29th Conference on Information Communications (INFOCOM)*, pages 1–9, March 2010.

[CEL+06] Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, and Ronald A. Peterson. People-Centric Urban Sensing. In *Proceedings of the 2nd Annual International Workshop on Wireless Internet (WICON)*, New York, NY, USA, 2006. ACM.

[CEL+08] Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, Ronald A. Peterson, Hong Lu, Xiao Zheng, Mirco Musolesi, Kristóf Fodor, and Gahng-Seop Ahn. The Rise of People-Centric Sensing. *IEEE Internet Computing*, 12(4):12–21, 2008.

[CGGPC06] Jose Manuel Cano-Garcia, Eva Gonzalez-Parada, and Eduardo Casilari. Experimental Analysis and Characterization of Packet Delay in UMTS Networks. In *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, Lecture Notes in Computer Science, pages 396–407, 2006.

[CHK08] Dana Cuff, Mark Hansen, and Jerry Kang. Urban Sensing: Out of the Woods. *Commun. ACM*, 51(3):24–33, March 2008.

[CKK+08] Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, and Nikos Triandopoulos. Anonysense: Privacy-Aware People-Centric Sensing. In *Proceeding of the 6th International Conference on Mobile Systems, Applications and Services (MobiSys)*, pages 211–224, New York, NY, USA, 2008. ACM.

[CMGSS13] D. Camps-Mur, A Garcia-Saavedra, and P. Serrano. Device-to-device communications with wi-fi direct: overview and experimentation. *IEEE Wireless Commun. Mag.*, 20(3):96–104, June 2013.

[Com96] European Commission. The Green Paper on Future Noise Policy (COM(96) 540). November 1996.

[Com13] European Commission. Environment: New policy package to clean up europe's air. Press Release, December 2013.

[com14] ComNSense Project. `http://www.comnsense.de/`, 2014. Accessed: 15.1.2015.

[Cre93] Noel A. C. Cressie. *Statistics for Spatial Data*. Wiley-Interscience, October 1993. ISBN: 978-0-471-00255-0.

[DBF12] Akshay Dua, Nirupama Bulusu, and Wu-chang Feng. Privacy-preserving online mixing of high integrity mobile multi-user data. In Muttukrishnan Rajarajan, Fred Piper, Haining Wang, and George Kesidis, editors, *Security and Privacy in Communication Networks*, volume 96 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 470–479. Springer Berlin Heidelberg, 2012.

[DEA+12] G. Drosatos, P.S. Efraimidis, I.N. Athanasiadis, E. D'Hondt, and M. Stevens. A privacy-preserving cloud computing system for creating participatory noise maps. In *Proceedings of the 36th IEEE Annual Computer Software and Applications Conference (COMPSAC)*, pages 581–586, July 2012.

[DGM+04] Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. Model-Driven Data Acquisition in Sen-

sor Networks. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 588–599. VLDB Endowment, 2004.

[DK08] Abhimanyu Das and David Kempe. Sensor Selection for Minimizing Worst-Case Prediction Error. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 97–108, Washington, DC, USA, April 2008. IEEE Computer Society.

[DMP+10] Tathagata Das, Prashanth Mohan, Venkata N. Padmanabhan, Ramachandran Ramjee, and Asankhaya Sharma. Prism: platform for remote sensing using smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 63–76, New York, NY, USA, 2010. ACM.

[DNC+01] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Trans. Robot. Autom.*, 17(3):229–241, June 2001.

[DSJ13] Ellie D'Hondt, Matthias Stevens, and An Jacobs. Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. *Pervasive and Mobile Computing*, 9(5):681–694, 2013.

[DZP+14] Ellie D'Hondt, Jesse Zaman, Eline Philips, Elisa Gonzalez Boix, and Wolfgang De Meuter. Orchestration support for participatory sensing campaigns. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pages 727–738, New York, NY, USA, 2014. ACM.

[EGH+08] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceeding of the 6th International Conference on Mobile Systems, Applications,*

and Services (MobiSys), pages 29–39, New York, NY, USA, 2008. ACM.

[Eis08] Shane Brophy Eisenman. *People-centric mobile sensing networks.* PhD thesis, Columbia University, New York, NY, USA, 2008. AAI3333332.

[ELC08] Shane B. Eisenman, Nicholas D. Lane, and Andrew T. Campbell. Techniques for improving opportunistic sensor networking performance. In *Proceedings of the 4th IEEE international Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 157–175, Berlin, Heidelberg, 2008. Springer-Verlag.

[ELC10] Shane B. Eisenman, Hong Lu, and Andrew T. Campbell. Halo: Managing node rendezvous in opportunistic sensor networks. In Rajmohan Rajaraman, Thomas Moscibroda, Adam Dunkels, and Anna Scaglione, editors, *Distributed Computing in Sensor Systems*, volume 6131 of *Lecture Notes in Computer Science*, pages 273–287. Springer Berlin Heidelberg, 2010.

[EMP] Ericsson mobile positioning system. `http://www.ericsson.com/ourportfolio/products/mobile-positioning-system`. Accessed: 15.1.2015.

[EYA13] J. Eberle, Zhixian Yan, and K. Aberer. Energy-efficient opportunistic collaborative sensing. In *Proceedings of the 10th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, pages 374–378, October 2013.

[Fil12] Yevgeniya Filippovska. *Evaluierung generalisierter Gebäudegrundrisse in großen Maßstäben.* PhD thesis, Universität Stuttgart, 2012.

[FJVF10] A. Fietz, S. M Jackisch, B. A Visel, and D. Fritsch. Automated 2D measuring of interiors using a mobile platform. In *Proceedings of the Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2010.

[Fox05] E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Comp. Graph. and Applicat.*, 25:38–46, 2005.

[GB01] H. González-Banos. A Randomized Art-Gallery Algorithm for Sensor Placement. In *Proceedings of the 17th Annual Symposium on Computational Geometry (SCG)*, pages 232–240, New York, NY, USA, June 2001. ACM.

[GKS05] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 265–272, New York, NY, USA, 2005. ACM.

[GLC+08] Shravan Gaonkar, Jack Li, Romit Roy Choudhury, Landon Cox, and Al Schmidt. Micro-blog: Sharing and querying content through mobile phones and social participation. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 174–186, New York, NY, USA, 2008. ACM.

[Gmb] Navvis GmbH. Explore – navigate – interact. `http://www.navvis.com/`. Accesed: 15.1.2015.

[goo14] Google Maps/Google Earth APIs Terms of Service. `https://developers.google.com/maps/terms`, May 2014. Accessed: 5.8.2014.

[GP10] G. Gröger and L. Plümer. Derivation of 3D indoor models by grammars for route planning. *Photogrammetrie-Fernerkundung-Geoinformation*, 2010:193–210, 2010.

[Gur91] Klaus-Werner Gurgel. Erfahrungen mit dem Satelliten-Navigationssystem GPS – Genauigkeiten an Land und auf See. *Deutsche Hydrografische Zeitschrift*, 44(1):35–49, 1991.

[GZY+14] Ruipeng Gao, Mingmin Zhao, Tao Ye, Fan Ye, Yizhou Wang, Kaigui Bian, Tao Wang, and Xiaoming Li. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *Proceedings of*

the 20th Annual International Conference on Mobile Computing and Networking (MobiCom), pages 249–260, New York, NY, USA, 2014. ACM.

[Har13] R. Harle. A survey of indoor inertial positioning systems for pedestrians. *IEEE Communications Surveys Tutorials*, 15(3):1281–1293, March 2013.

[HBS11] Hai Huang, Claus Brenner, and Monika Sester. 3D Building Roof Reconstruction from Point Clouds via Generative Models. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, pages 16–24, New York, NY, USA, 2011. ACM.

[HBZ+06] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. CarTel: a distributed mobile sensor computing system. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 125–138, New York, NY, USA, 2006. ACM.

[HD09] Beverly Harrison and Anind Dey. What have you done with location-based services lately? *IEEE Pervasive Comput.*, 8(4):66–70, Oct.-Dec. 2009.

[HMC+13] Michael Hardegger, Sinziana Mazilu, Dario Caraci, Frederik Hess, Daniel Roggen, and Gerhard Tröster. ActionSLAM on a smartphone: At-home tracking with a fully wearable system. In *Proceedings of the 4th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2013.

[HW08] Mordechai Haklay and Patrick Weber. OpenStreetMap: User-generated street maps. *IEEE Pervasive Comput.*, 7(4):12–18, October 2008.

[ibe15] iBeacon for developers. `https://developer.apple.com/ibeacon/`, 2015. Accessed: 15.1.2015.

[ISO13] ISO. Geoinformation – Datenqualität. Technical Report 19157:2013, International Organization for Standardization, Geneva, Switzerland, 2013.

[KAC+13] Robin Kravets, Hilfi Alkaff, Andrew Campbell, Karrie Karahalios, and Klara Nahrstedt. CrowdWatch: Enabling in-network crowdsourcing. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC)*, pages 57–62, New York, NY, USA, 2013. ACM.

[KBG+10] Mikkel Baun Kjærgaard, Henrik Blunck, Torben Godsk, Thomas Toftkjær, Dan Lund Christensen, and Kaj Grønbæk. Indoor positioning using gps revisited. In *Proceedings of the Conference on Pervasive Computing (Pervasive)*, 2010.

[KBP+08] Eiman Kanjo, Steve Benford, Mark Paxton, Alan Chamberlain, Danae Stanton Fraser, Dawn Woodgate, David Crellin, and Adrain Woolard. MobGeoSen: Facilitating Personal Geosensor Data Collection and Visualization Using Mobile Phones. *Personal and Ubiquitous Computing*, 12:599–607, 2008.

[KBRL09] Eiman Kanjo, Jean Bacon, David Roberts, and Peter Landshoff. MobSens: Making smart phones smarter. *IEEE Pervasive Comput.*, 8(4):50–57, Oct.-Dec. 2009.

[KK00] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Steteless Routing for Wireless Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, Boston, MA, August 2000.

[KKES10] Donnie H. Kim, Younghun Kim, Deborah Estrin, and Mani B. Srivastava. SensLoc: sensing everyday places and paths using less energy. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 43–56, New York, NY, USA, 2010. ACM.

[KNLZ07] A. Kansal, S. Nath, Jie Liu, and Feng Zhao. SenseWeb: An infrastructure for shared sensing. *IEEE Multimedia*, 14(4):8–13, Oct.-Dec. 2007.

[KR08] B. Krach and P. Robertson. Integration of foot-mounted inertial sensors into a bayesian location estimation framework. In *Proceedings of the 5th Workshop on Positioning, Navigation and Communication (WPNC)*, pages 55–61, March 2008.

[Kra08] Andreas Krause. *Optimizing Sensing – Theory and Applications*. PhD thesis, Carnegie Mellon University, 2008.

[Kra14] A. Krause. Community sense-and-response systems: Your phone as seismometer. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 394–394, March 2014.

[KTC+08] Apu Kapadia, Nikos Tri, Cory Cornelius, Daniel Peebles, and David Kotz. AnonySense: Opportunistic and privacy-preserving context collection. In *Proceedings of the 6th International Conference on Pervasive Computing (Pervasive)*, 2008.

[KZ07] Aman Kansal and Feng Zhao. Location and mobility in a sensor network of mobile phones. In *Proceedings of the 17th International Workshop on Network and Operating Systems Support for Digital Audio & Video (NOSSDAV)*, June 2007.

[LBD+05] Benyuan Liu, Peter Brass, Olivier Dousse, Philippe Nain, and Don Towsley. Mobility Improves Coverage of Sensor Networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 300–308, New York, NY, USA, 2005. ACM.

[LEM+08] Nicholas D. Lane, Shane B. Eisenman, Mirco Musolesi, Emiliano Miluzzo, and Andrew T. Campbell. Urban Sensing Systems: Opportunistic or Participatory? In *Proceedings of the 9th Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 11–16, New York, NY, USA, 2008. ACM.

[LGS06] Ming Li, Deepak Ganesan, and Prashant Shenoy. PRESTO: feedback-driven data management in sensor networks. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation (NSDI)*, volume 3, pages 23–23, Berkeley, CA, USA, 2006. USENIX Association.

[LLEC08] Nicholas D. Lane, Hong Lu, Shane B. Eisenman, and Andrew T. Campbell. Cooperative techniques supporting sensor-based people-centric inferencing. In *Proceedings of the 6th International Conference on Pervasive Computing (Pervasive)*, pages 75–92, Berlin,Heidelberg, 2008. Springer-Verlag.

[LLEC09] Hong Lu, Nicholas D. Lane, Shane B. Eisenman, and Andrew T. Campbell. Bubble-sensing: Binding Sensing Tasks to the Physical World. *Pervasive and Mobile Computing*, 6(1):58 – 71, 2009.

[LLSC11] Ted Tsung-Te Lai, Chun-yi Lin, Ya-Yunn Su, and Hao-hua Chu. BikeTrack: Tracking stolen bikes through everyday mobile phones and participatory sensing. In *Proceedings of the 2nd International Workshop on Sensing Applications on Mobile Phones (PhoneSense)*, Seattle, USA, November 2011.

[LWG+09] Ralph Lange, Harald Weinschrott, Lars Geiger, Andre Blessing, Frank Dürr, Kurt Rothermel, and Hinrich Schütze. On a Generic Uncertainty Model for Position Information. In Kurt Rothermel, Dieter Fritsch, Wolfgang Blochinger, and Frank Dürr, editors, *Proceedings of the 1st International Workshop on Quality of Context (QuaCon)*, volume 5786 of *Lecture Notes in Computer Science*, pages 76–87. Springer, June 2009.

[LYL+10] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 71–84, New York, NY, USA, 2010. ACM.

[MCR+10] Emiliano Miluzzo, Cory T. Cornelius, Ashwin Ramaswamy, Tanzeem Choudhury, Zhigang Liu, and Andrew T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 5–20, New York, NY, USA, 2010. ACM.

[MDF+09] S. May, D. Droeschel, S. Fuchs, D. Holz, and A. Nuchter. Robust 3D-mapping with time-of-flight cameras. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2009.

[ML12a] D. Mendez and M.A. Labrador. Density maps: Determining where to sample in participatory sensing systems. In *Proceedings of the 3rd FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing (MUSIC)*, pages 35–40, June 2012.

[ML12b] D. Mendez and M.A. Labrador. Removing spatial outliers in ps applications. In *Proceedings of the International Conference on Selected Topics in Mobile and Wireless Networking (iCOST)*, pages 66–71, July 2012.

[MLEC07] Emiliano Miluzzo, Nicholas D. Lane, Shane B. Eisenman, and Andrew T. Campbell. CenceMe – Injecting Sensing Presence into Social Networking Applications. In Gerd Kortuem, Joe Finney, Rodger Lea, and Vasughi Sundramoorthy, editors, *Smart Sensing and Context*, volume 4793 of *Lecture Notes in Computer Science*, pages 1–28. Springer Berlin / Heidelberg, 2007.

[MLR13] D. Mendez, M. Labrador, and K. Ramachandran. Data interpolation for participatory sensing systems. *Pervasive and Mobile Computing*, 9(1):132–148, 2013.

[MPF+10] Mirco Musolesi, Mattia Piraccini, Kristof Fodor, Antonio Corradi, and Andrew T. Campbell. Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones. In *Proceedings of the 8th International Conference on Pervasive Com-*

puting (Pervasive), volume 6030 of *LCNS*, pages 355–372, Helsinki, Finland, May 2010. Springer.

[MPL+11] Emiliano Miluzzo, Michela Papandrea, Nicholas D. Lane, Andy M. Sarroff, Silvia Giordano, and Andrew T. Campbell. Tapping into the vibe of the city using VibN, a continuous sensing application for smartphones. In *Proceedings of 1st International Symposium on From Digital Footprints to Social and Community Intelligence (SCI)*, pages 13–18, New York, NY, USA, 2011. ACM.

[MPLM11] D. Mendez, A.J. Perez, M.A. Labrador, and J.J. Marron. P-Sense: A participatory sensing system for air pollution monitoring and control. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 344–347, March 2011.

[MSN+09] Nicolas Maisonneuve, Matthias Stevens, Maria E. Niessen, Peter Hanappe, and Luc Steels. Citizen Noise Pollution Monitoring. In *Proceedings of the 10th Annual International Conference on Digital Government Research*, dg.o '09, pages 96–103. Digital Government Society of North America, 2009.

[MTKW02] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 593–598, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[MWH+06] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25:614–623, 2006.

[MZWVG07] P. Müller, G. Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3):85, 2007.

[NBP+09] Daniel Nadeau, W. Brutsaert, M. Parlange, E. Bou-Zeid, G. Barrenetxea, O. Couach, M.-O. Boldi, J. Selker, and M. Vetterli. Estimation of urban sensible heat flux using a dense wireless network of observations. *Environmental Fluid Mechanics*, 9:635–653, 2009. 10.1007/s10652-009-9150-7.

[NDI+11] Richard A. Newcombe, Andrew J. Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. KinectFusion: real-time dense surface mapping and tracking. In *Proceedings of the Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[N.V14] Koninklijke Philips N.V. Where is the guacamole? Don't worry, Philips' supermarket lighting will tell you! Press Release, 2014.

[opea] Openstreetmap. `http://www.openstreetmap.org/`. Accessed: 15.1.2015.

[opeb] Openstreetmap — copyright and license. `http://www.openstreetmap.org/copyright`. Accessed: 5.8.2014.

[PBF13] Michael Peter, Susanne Becker, and Dieter Fritsch. Grammar supported indoor mapping. In *Proceedings of the 26th International Cartographic Conference (ICC)*, pages 1–18, Dresden, 2013.

[PHF11] Michael Peter, Norbert Haala, and Dieter Fritsch. Using photographed evacuation plans to support mems imu navigation. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Guimaraes, Portugal, 2011.

[PHSO10] Michael Peter, Norbert Haala, Markus Schenk, and Tim Otto. Indoor navigation and modeling using photographed evacuation plans and mems imu. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVIII, Part 4 Commission IV Symposium*, Orlando, USA, November 2010.

[PL90] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer New York, 1990.

[PLL10] Bodhi Priyantha, Dimitrios Lymberopoulos, and Jie Liu. EERS: Energy efficient responsive sleeping on mobile phones. In *Proceedings of the International Workshop on Sensing for App Phones (PhoneSense)*, Zurich, Switzerland, November 2010.

[PLL11] B. Priyantha, D. Lymberopoulos, and Jie Liu. LittleRock: Enabling energy-efficient continuous sensing on mobile phones. Technical report, Microsoft Research, 2011.

[PM01] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 301–308, New York, NY, USA, 2001. ACM.

[pro15] Project Tango. `http://www.google.com/atap/project-tango/`, 2015. Accessed: 7.6.2015.

[PSHG11] Sarah Pidcock, Rob Smits, Urs Hengartner, and Ian Goldberg. NotiSense: An urban sensing notification system to improve bystander privacy. In *Proceedings of the 2nd International Workshop on Sensing Applications on Mobile Phones (PhoneSense)*, Seattle, USA, November 2011.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RAK10] P. Robertson, M. Angermann, and M. Khider. Improving simultaneous localization and mapping for pedestrian navigation and automatic mapping of buildings by using online human-based feature labeling. In *Proceedings of the IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 365–374, May 2010.

[RB07] O. Riva and C. Borcea. The urbanet revolution: Sensor power to the people! *IEEE Pervasive Comput.*, 6(2):41–49, April–June 2007.

[RCK+10] Rajib Kumar Rana, Chun Tung Chou, Salil S. Kanhere, Nirupama Bulusu, and Wen Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 105–116, New York, NY, USA, 2010. ACM.

[RES10] Sasank Reddy, Deborah Estrin, and Mani B Srivastava. Recruitment Framework for Participatory Sensing Data Collections. In *Proceedings of the International Conference on Pervasive Computing (Pervasive)*, page 18, May 2010.

[RFA+13] P. Robertson, M. Frassl, M. Angermann, M. Doniec, B.J. Julian, M. Garcia Puyol, M. Khider, M. Lichtenstern, and L. Bruno. Simultaneous localization and mapping for pedestrians using distortions of the local magnetic field intensity in large indoor environments. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pages 1–10, Oct 2013.

[RKM13] Valentin Radu, Lito Kriara, and Mahesh K. Marina. Pazl: A mobile crowdsensing based indoor wifi monitoring system. In *CNSM*, pages 75–83, 2013.

[RMM12] KiranK. Rachuri, Cecilia Mascolo, and Mirco Musolesi. Energy-accuracy trade-offs of sensor sampling in smart phone based sensing systems. In Tom Lovett and Eamonn O'Neill, editors, *Mobile Context Awareness*, pages 65–76. Springer London, 2012.

[Rot02] Jörg Roth. *Mobile Computing: Grundlagen, Technik, Konzepte*. dpunkt-verlag, Heidelberg, 2002.

[RSB+08] Sasank Reddy, Katie Shilton, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani B Srivastava. Evaluating Participation and Performance in Participatory Sensing. In *International Workshop on*

Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense) at Sensys, page 5, November 2008.

[RSB+09] Sasank Reddy, Katie Shilton, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani B Srivastava. Using Context Annotated Mobility Profiles to Recruit Data Collectors in Participatory Sensing. In *Proceedings of the 4th International Symposium on Location and Context Awareness (LOCA)*, page 18. Springer-Verlag, May 2009.

[SBS+11] Immanuel Schweizer, Roman Bärtl, Axel Schulz, Florian Probst, and Max Mühlhäuser. NoiseMap – real-time participatory noise maps. In *Proceedings of the 2nd International Workshop on Sensing Applications on Mobile Phones (PhoneSense)*, Seattle, USA, November 2011.

[SC10] Hyojeong Shin and Hojung Cha. Wi-fi fingerprint-based topological map building for indoor user tracking. In *Proceedings of the 16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 105–113, August 2010.

[SCC12] Hyojeong Shin, Yohan Chon, and Hojung Cha. Unsupervised construction of an indoor floor plan using a smartphone. *IEEE Trans. Syst., Man, and Cybernetics, C: Applicat. and Reviews*, 42:889–898, 2012.

[SCZ+13] Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang. Walkie-Markie: Indoor pathway mapping made easy. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, pages 85–98, Berkeley, CA, USA, 2013. USENIX Association.

[SEKX98] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, June 1998.

[SFRJ09] Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nicholas R. Jennings. Decentralised Coordination of Mobile Sensors Using the Max-Sum Algorithm. In *Proceedings of the 21st International Joint Conference on Artifical intelligence (IJCAI)*, pages 299–304, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[Sky] Skyhook. Skyhook: How It Works. Electronic. [Online; 2010/11/12].

[Slu08] R. Sluiter. Interpolation methods for climate data. Technical report, KNMI, R&D Information and Observation Technology, November 2008.

[SM08] Anthony Steed and Richard Milton. Using Tracked Mobile Sensors to Make Maps of Environmental Effects. *Personal and Ubiquitous Computing*, 12(4):331–342, 2008.

[SMBL12] Sougata Sen, Archan Misra, Rajesh Balan, and Lipyeow Lim. The case for cloud-enabled mobile sensing services. In *Proceedings of the Workshop on Mobile Cloud Computing (MCC)*, pages 53–58, August 2012.

[SR06] Silvia Santini and Kay Römer. An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks. In *Proceedings of the 3rd International Conference on Networked Sensing Systems (INSS)*, pages 29–36, 2006.

[Ste] Illya Stepanov. CANU Mobility Simulation Environment (CanuMobiSim). `http://canu.informatik.uni-stuttgart.de/mobisim/index.html`. Accessed: 15.1.2015.

[STKC09] Minho Shin, P. Tsang, D. Kotz, and C. Cornelius. DEAMON: Energy-efficient sensor monitoring. In *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9, June 2009.

[STY05] Anton Schwaighofer, Volker Tresp, and Kai Yu. Learning gaussian process kernels via hierarchical bayes. In *Advances in Neural Information Processing Systems (NIPS)*, number 17, pages 1209–1216. MIT Press, 2005.

[STZ12] Xiang Sheng, Jian Tang, and Weiyi Zhang. Energy-efficient collaborative sensing with mobile phones. In *Proceedings of IEEE INFOCOM*, pages 1916–1924, March 2012.

[TLL+11] Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, 30:11:1–11:14, 2011.

[TNNL02] Juan D. Tardós, José Neira, Paul M. Newman, and John J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Int. J. Robotics Research*, 21:311–330, 2002.

[TRL+09] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 85–98, New York, NY, USA, 2009. ACM.

[Var] András Varga. The OMNeT++ Simulator. `http://www.omnetpp.org/`. Accessed: 15.1.2015.

[VLML14] I.J. Vergara-Laurens, D. Mendez, and M.A. Labrador. Privacy, quality of information, and energy consumption in participatory sensing systems. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 199–207, March 2014.

[WAK+10] Wesley Willett, Paul Aoki, Neil Kumar, Sushmita Subramanian, and Allison Woodruff. Common sense community: Scaffolding mobile sensing and analysis for novice users. In *Proceedings of the 8th International Conference on Pervasive Computing*, May 2010.

[WDMCM13] Mariusz Wisniewski, Gianluca Demartini, Apostolos Malatras, and Philippe Cudré-Mauroux. Noizcrowd: A crowd-based data gathering and management system for noise level data. In Florian Daniel, George A. Papadopoulos, and Philippe Thiran, editors, *Mobile Web Information Systems*, volume 8093 of *Lecture Notes in Computer Science*, pages 172–186. Springer Berlin Heidelberg, 2013.

[WDR09] Harald Weinschrott, Frank Dürr, and Kurt Rothermel. Efficient Capturing of Environmental Data with Mobile RFID Readers. In *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware (MAM)*, pages 41–51, Washington, DC, USA, 2009. IEEE Computer Society.

[WDR10] Harald Weinschrott, Frank Dürr, and Kurt Rothermel. StreamShaper: Coordination Algorithms for Participatory Mobile Urban Sensing. In *Proceedings of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 195–204, San Francisco, CA, USA, November 2010. IEEE.

[WGR13] Huayan Wang, Stephen Gould, and Daphne Roller. Discriminative learning with latent variables for cluttered indoor scene understanding. *Communications of the ACM*, 56(4):92–99, April 2013.

[WH08] Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. In *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp)*, pages 114–123, New York, NY, USA, 2008. ACM.

[WJH97] Andy Ward, A Jones, and A Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

[WKB08] Widyawan, M. Klepal, and S. Beauregard. A backtracking particle filter for fusing building plans with pdr displacement estimates. In *Proceedings of the 5th Workshop on Positioning, Navigation and Communication (WPNC)*, pages 207–212, March 2008.

[WL11] Jens Weppner and Paul Lukowicz. Collaborative crowd density estimation with mobile phones. In *Proceedings of the 2nd International Workshop on Sensing Applications on Mobile Phones (PhoneSense)*, Seattle, USA, November 2011.

[Woo07] O. Woodman. An introduction to inertial navigation. Technical Report 696, University of Cambridge, 2007.

[XSF10] Yiguang Xuan, Raja Sengupta, and Y. Fallah. Making indoor maps with portable accelerometer and magnetometer. In *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, pages 1–7, October 2010.

[XSK+10] Yu Xiao, Petri Savolainen, Arto Karppanen, Matti Siekkinen, and Antti Ylä-Jääski. Practical power modeling of data transmission over 802.11g for wireless applications. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (e-Energy)*, pages 75–84, New York, NY, USA, 2010. ACM.

[XZWC14] H. Xiong, D. Zhang, L. Wang, and H. Chaouchi. EMC3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint. *IEEE Trans. Mobile Comput.*, PP(99):1–14, 2014.

[YKG10] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. CrowdSearch: Exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 77–90, New York, NY, USA, 2010. ACM.

[ZJTS14] Xiuming Zhang, Yunye Jin, Hwee-Xian Tan, and Wee-Seng Soh. CIMLoc: A crowdsourcing indoor digital map construction system for localization. In *Proceedings of the 9th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, April 2014.

[ZMLZ13] Dong Zhao, Huadong Ma, Liang Liu, and Jing Zhao. On opportunistic coverage for urban sensing. In *Proceedings of the 10th*

IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS), pages 231–239, October 2013.

[ZXWC14] Daqing Zhang, Haoyi Xiong, Leye Wang, and Guanling Chen. CrowdRecruiter: Selecting participants for piggyback crowdsensing under probabilistic coverage constraint. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pages 703–714, New York, NY, USA, 2014. ACM.

Erklärung

Ich erkläre hiermit, dass ich, abgesehen von den ausdrücklich bezeichneten Hilfsmitteln und den Ratschlägen von jeweils namentlich aufgeführten Personen, die Dissertation selbstständig verfasst habe.

(Damian Cassius Philipp)