FGI PUBLICATIONS № 158

Context Awareness for Navigation Applications

by

Robert E. Guinness

Doctoral dissertation for the degree of Doctor of Science in Technology to be presented with due permission for public examination and debate in Tietotalo Building, Auditorium TB109, at Tampere University of Technology on the 21st of December 2015 at 12 noon.

KIRKKONUMMI 2015

Supervising professor
Professor Jarmo Takala, Department of Pervasive Computing, Tampere University of Technology, Finland

Thesis advisors
Professor Ruizhi Chen, Conrad Blucher Institute for Surveying and Science, School of Engineering and Computer Science, Texas A&M University – Corpus Christi, USA

Professor Heidi Kuusniemi, Department of Navigation and Positioning, Finnish Geospatial Research Institute, National Land Survey of Finland

Preliminary examiners
Dr. Martin Werner, Mobile and Distributed Systems Group, Institute for Informatics, Ludwig-Maximilians-Universität München, Germany

D.Sc.(Tech) Susanna Pirttikangas, Department of Computer Science and Engineering, University of Oulu, Finland

Opponents
Dr. Martin Werner, Mobile and Distributed Systems Group, Institute for Informatics, Ludwig-Maximilians-Universität München, Germany

D.Sc.(Tech) Jari Syrjärinne, HERE, a Nokia business, Finland

ISBN (printed): 978-951-48-0249-2
ISBN (pdf): 978-951-48-0250-8
ISSN (print): 2342-7345
ISSN (online): 2342-7353

Grano Oy, Vantaa 2015

ABSTRACT

This thesis examines the topic of context awareness for navigation applications and asks the question, “What are the benefits and constraints of introducing context awareness in navigation?” Context awareness can be defined as a computer’s ability to understand the situation or context in which it is operating. In particular, we are interested in how context awareness can be used to understand the navigation needs of people using mobile computers, such as smartphones, but context awareness can also benefit other types of navigation users, such as maritime navigators. There are countless other potential applications of context awareness, but this thesis focuses on applications related to navigation. For example, if a smartphone-based navigation system can understand when a user is walking, driving a car, or riding a train, then it can adapt its navigation algorithms to improve positioning performance.

We argue that the primary set of tools available for generating context awareness is machine learning. Machine learning is, in fact, a collection of many different algorithms and techniques for developing “computer systems that automatically improve their performance through experience” [1]. This thesis examines systematically the ability of existing algorithms from machine learning to endow computing systems with context awareness. Specifically, we apply machine learning techniques to tackle three different tasks related to context awareness and having applications in the field of navigation: (1) to recognize the activity of a smartphone user in an indoor office environment, (2) to recognize the mode of motion that a smartphone user is undergoing outdoors, and (3) to determine the optimal path of a ship traveling through ice-covered waters. The diversity of these tasks was chosen intentionally to demonstrate the breadth of problems encompassed by the topic of context awareness.

During the course of studying context awareness, we adopted two conceptual “frameworks,” which we find useful for the purpose of solidifying the abstract concepts of context and context awareness. The first such framework is based strongly on the writings of a rhetorician from Hellenistic Greece, Hermagoras of Temnos, who defined seven elements of “circumstance”. We adopt these seven elements to describe contextual information. The second framework, which we dub the “context pyramid” describes the processing of raw sensor data into contextual information in terms of six different levels. At the top of the pyramid is “rich context”, where the information is expressed in prose, and the goal for the computer is to mimic the way that a human would describe a situation.

We are still a long way off from computers being able to match a human’s ability to understand and describe context, but this thesis improves the state-of-the-art in context awareness for navigation applications. For some particular tasks, machine learning has succeeded in outperforming humans, and in the future there are likely to be tasks in navigation where computers outperform humans. One example might be the route optimization task described above. This is an example of a task where many different types of information must be fused in non-obvious ways, and it may be that computer algorithms can find better routes through ice-covered waters than even well-trained human navigators. This thesis provides only preliminary evidence of this possibility, and future work is needed to further develop the techniques outlined here. The same can be said of the other two navigation-related tasks examined in this thesis.

PREFACE

The research work presented in this thesis was carried out between November 2011 and May 2014. I have chosen to complete a “compendium-style” dissertation, in part because I have already had the pleasure of preparing a monograph when co-authoring a book with Prof. Ruizhi Chen, published in July 2014. I have no great desire to repeat such an experience yet. As many who have published such monographs can attest, it takes a lot out of you!

Due to other responsibilities, as well as a bad case of the “it’s-not-good-enough-yet” syndrome, it took me more than one year to finalize and publish some of the results of my doctoral research in article format. With the aid of gentle nudging from my colleagues and superiors, I prepared the summary content for this compendium mostly between January and July 2015.

There are two particular experiences I’d like to share that also motivated me for completing this dissertation. The first is when I was asked to be a reviewer for an article submitted to one highly-esteemed journal. When I realized that my work, in my own opinion, was superior to that which I was reviewing, I felt suddenly cured of the above-mentioned syndrome. This is one of the side benefits of peer review. The second was when I was participating in an interview of a now colleague (he got the job!). We asked him if he could describe one achievement of which he was most proud. Instead of pointing to one particular academic achievement, such as a highly-cited paper, he pointed out another kind of achievement: the fact that he can look back at his publications and realize that some of the early ones were poor but that there has been a steady improvement in the quality over the years. Since hearing that, this is what I aim for: Not to publish the perfect gem some day but to continually put out my work-in-progress for others to see and hopefully benefit from. Then, refine and repeat.

There are many I would like to thank for their contributions to this thesis and to my overall development. First and foremost, however, I thank God for the wonderful life and opportunities He has given me and for always being with me. Next, I’d like to thank my family for their many years of support, beginning with my parents but also including my siblings, Erin and Joe. Nowadays I have my own family, and when someone completes a dissertation in the midst of family life, there is often an unsung hero (or heroine) behind the work. In my case, it is my wife, Anne-Mari, who spent many long days and evenings taking care of our boys while I was completing this thesis. Thank you, my angel, for everything. Then, to my children Pyry, Kilian, and Aarre: Thank you for bringing joy to each day.

Next, I'd like to thank my thesis supervisors, Prof. Jarmo Takala, Prof. Ruizhi Chen, and Prof. Heidi Kuusniemi. Thank you for all the support and guidance, as well as the opportunity to complete this research under your supervision. Thank you also to Prof. Ling Pei, who served as an instructor during my early days at FGI. Thank you to my pre-examiners, Dr. Martin Werner (who will also serve as an opponent at my defense) and Dr. Susanna Pirttikangas. A special thanks goes to Dr. Jari Syrjärinne who agreed to be my second opponent on short notice. I'd also like to thank Dr. Valérie Renaudin, who also agreed to be an opponent, but due to a last minute change in the defense date was unable to attend.

Lastly, I'd like to thank my colleagues and co-authors, who have made the environment in which this work was completed very fun, interesting, and rewarding. There are so many of you nowadays that I won't attempt to name everyone individually, but please know that I value each and every one of you. I have learned so much from my colleagues, and you have made it a joy to come to work each day.

Kirkkonummi, 20.11.2015

Robert E. Guinness

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ACM	Association for Computing Machinery
AdaBoost	Adaptive Boosting
ADL	Activities of Daily Life
AESS	Aerospace and Electronics Systems Society
AI	Artificial Intelligence
AIS	Automatic Identification System
ANN	Artificial Neural Network
API	Application Programming Interface
BC	Before Christ
BN	Bayesian Network
CHMM	Coupled Hidden Markov Model
COTS	commercial-off-the-shelf
CPU	Central Processing Unit
CRF	Conditional Random Fields
DBSCAN	density-based spatial clustering of applications with noise
DT	Decision Tree
DTS	discriminative temporal smoothing
E	East

EM	expectation-maximization
Five Ws	Who, What, Where, When, and Why
FMS	Future Urban Mobility Survey
GIS	Geographic Information Systems
GMM	Guassian mixture model
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPSAR	Google Play Services Activity Recognition
HCI	Human-Computer Interaction
HDA	Hazard Detection and Avoidance
HF-SVM	Hardware-friendly Support Vector Machines
HIER	hierarchical agglomerative clustering
HMM	Hidden Markov Models
IBk	instance-based k-nearest neighbor algorithm
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
ION	Institute of Navigation
IRO-2	Ice Forecast and Route Optimization
kMC	k-means clustering
kNN	k-Nearest Neighbor

KStar	K* algorithm
lat	latitude
LBS	Location-Based Services
LDA	Linear Discriminant Analysis
LoMoCo	Location-Motion-Context
long	longitude
LR	logistic regression
LS-SVM	Least Squares-Support Vector Machine
LSA	latent semantic analysis
LWL	Locally weighted learning
MAP	maximum a posteriori
MCMC	Markov chain Monte Carlo
MEMS	microelectromechanical systems
MLE	maximum likelihood estimate
MLP	Multilayer Perceptron
N	North
NB	Naïve Bayes
OPTICS	ordering points to identify the clustering structure
OS	Operating System
PCA	principal component analysis
PDA	Personal Digital Assistant
PLANS	Position Location and Navigation Symposium

RF	RandomForest
RSSI	Received Signal Strength Indicator
SAR	Synthetic Aperture Radar
SLAM	Simultanous Localization and Mapping
SVM	Support Vector Machines
UAVs	Unmanned Aerial Vehicles
UCI	University of California, Irvine
WGS84	World Geodetic System 1984
WLAN	wireless local area network

LIST OF PUBLICATIONS

This thesis contains a compilation of five previously published papers, referred to as [P#] throughout the text. The following publications are included:

[P1] R. Chen, **R. E. Guinness**, “Context Awareness” in *Geospatial computing in mobile devices*. Boston: Artech House, ch. 8, pp. 150-170, 2014.

[P2] R. Chen, **R. E. Guinness**, “Contextual Reasoning” in *Geospatial computing in mobile devices*. Boston: Artech House, ch. 9, pp. 171-197, 2014.

[P3] L. Pei, **R. E. Guinness**, R. Chen, J. Liu, H. Kuusniemi, Y. Chen, L. Chen, and J. Kaistinen, “Human behavior cognition using smartphone sensors,” *Sensors*, vol. 13, no. 2, pp. 1402–1424, 2013.

[P4] **R. E. Guinness**, “Beyond where to how: A machine learning approach for sensing mobility contexts using smartphone sensors,” *Sensors*, vol. 15, no. 5, pp. 9962–9985, 2015.

[P5] **R. E. Guinness**, J. Saarimäki, L. Ruotsalainen, H. Kuusniemi, F. Goerlandt, J. Montewka, R. Berglund, and V. Kotovirta, "A method for ice-aware maritime route optimization,” in *Position, Location and Navigation Symposium–PLANS 2014*, IEEE/ION, pp. 1371–1378, 2014.

1. INTRODUCTION

1.1 Background and Motivation

We are currently witnessing an era of technological convergence that rivals some of the great technological upheavals of modern history[1]. The steam engine, the electric lamp, the transistor, the jetliner, the artificial satellite—it is in this same revered company that we can place the technological revolution we are now undergoing. According to authors Erik Brynjolfsson and Andrew McAfee, we are living in a "second machine age" (where the first machine age began with James Watt's steam engine), which they describe as "an inflection point in the history of our economies and societies because of digitization" [2, p. 11]. They define digitization as "converting things into bits that can be stored on a computer and sent over a network" [2, p. 10]. The resulting digital information has remarkably different properties from the industrial products of the first machine age, a topic which Brynjolfsson and McAfee explore in detail in their book. They define "digital technologies" as "those that have computer hardware, software, and networks at their core" [2, p. 9]. It is within this wider context of digital technologies and the second machine age that this thesis is best understood.

Digital technology is a broad category; therefore, it is useful to narrow the focus to a few key technologies that are driving the development of the second machine age. There are four specific technologies that have particular relevance to this thesis: (1) mobile telecommunication devices, (2) the Internet, (3) positioning technologies, and (4) a wide range of inexpensive yet highly capable sensors, namely microelectromechanical systems (MEMS). We note that these four technologies have converged over the course of a few decades, so that the changes are clearly evident within one human generation (i.e. 20-30 years). All of these technologies came to

[1] By "technological convergence", we mean that a set of technologies has undergone rapid advances simultaneously and thus have become available for technological uptake in combinatorial ways.

a technological crossroads in the late 20th century and early 21st century, so that a child born and raised in the 21st century will have vastly different technological possibilities, compared to one born and raised in the 20th century.

The applications arising from this technological convergence span many different areas, and we have no intention to cover these applications exhaustively in this thesis. Instead we focus on one application area—navigation. The research scope of this thesis will be defined more precisely in Section 1.2.

The first major manifestation of this technological convergence, especially with respect to consumer markets, is the so-called "smartphone," which incorporates or supports all four of the above-mentioned technologies. Looking at the history of mobile devices, it is difficult to say which mobile phone can be considered the first smartphone. The first commercially-available phone with a Global Positioning System (GPS) receiver, the *Benefon Esc!*, was released in 1999. In terms of marketing, the Ericsson R380, released in 2000, was the first mobile phone to be called a smartphone. In terms of the four technologies listed above, the Samsung SCH-S310, introduced in 2005, was probably the first to exhibit all four. The first iPhone was released in 2007, and the first Android phone was released in 2008.

About 64 million smartphones were sold globally in 2006 [3], and by 2008 this number exceeded 139 million [4]. By 2012, there were already more than one billion smartphones in use worldwide [5]. This number is forecast to reach nearly 2.5 billion in 2015 [6]. These devices allow their users to stay "connected" virtually everywhere they go, and consequently anyone can connect to these billion plus users from any networked device, including desktop computers and "land-line" phones—no matter where the user is located or traveling to. Ironically, in many technologically advanced societies, it is now considered a societal and/or behavioral challenge for one to go "off the grid" or "disconnected" for any extended period of time.

It is our view that the smartphone is only the first manifestation of this technological revolution. Many other so-called "smart" devices are soon to follow: "smartwatches" and the use of various wearable sensors may soon become a mainstay consumer habit. In addition, the same technologies that have made smartphones possible and popular are quickly making their way into existing everyday devices, including cars, home appliances, and even toothbrushes. Furthermore, it is not just consumer markets that are being transformed but also many industrial markets, ranging from manufacturing to commercial shipping. It would be naïve to speculate exactly how this revolution will play out in the coming decades, but it is clear that the developments are already changing the lifestyles, habits, and possibilities of people living in the early 21st century, especially those who can afford these (currently) "high-end" consumer devices.

Aside from being a convergence of new digital technologies, is there any unifying concept or principle that is underlying this technological revolution? Some would argue that it is the increased levels of *mobility* that these technologies provide. Others have rallied under the banner of *ubiquitous computing* or *pervasive computing*, which describes the fact that computing devices can now be found nearly everywhere one looks. Certainly these are two important characteristics giving wind to this revolution, but we argue in this thesis towards another underlying principle that provides a common thread and deep insight into how our relationship to these computing devices is changing.

One common development, of course, is the increasing ability of computing devices to fulfill various user desires, e.g. to download large amounts of data at high speeds, to capture or render various high-quality multimedia content, to store and edit content in various ways, etc. What is not advancing or expanding—at least, not at any considerable rate—is the patience or attention span of the users themselves. Therefore, users are expecting (consciously or not) that their devices will "do more" with essentially the same total quantity and quality of human input. Fortunately, however, these devices are rapidly advancing in their ability to know what their users want or need without the user having to explicitly formulate and express these desires to the computer.

It is our view that we are not even close to unleashing the full potential of computing devices to understand their users. In many ways, smartphones and other so-called *smart devices* are not yet "smart". They have the "brawn" and not the brains, in the sense that they are powerful and capable but deficient in understanding the user's needs.

The field of study related to how humans communicate with computers and vice versa is known as Human-Computer Interaction (HCI), and this thesis finds relevance in HCI. Essentially, we aim to reduce the need for explicit human-to-computer communication by increasing the ability of computers to understand humans. This is the goal under which this thesis is motivated and focused—to improve our understanding of how computing devices can better understand us and our needs.

The primary method by which this thesis aims to achieve this goal is through *machine learning*. According to Tom Mitchell and co-authors, "machine learning research seeks to develop computer systems that automatically improve their performance through experience" [1]. This is our favorite definition of machine learning among the many found in the literature, but we note that achieving such a system is incredibly difficult. Most methods that go by the name of "machine learning" fail to meet this definition in terms of *automatically* improving performance. Nonetheless, the discipline of machine learning has grown in recent decades, and the set of techniques going by the name of machine learning is indeed very powerful. In many ways, machine learning has become the preferred framework for building up systems that understand users' needs. Some observers may note that such systems exhibit—or at least attempt to exhibit—*artificial intelligence*.

Artificial Intelligence (AI) has been an elusive goal of computer science researchers ever since the term was coined in 1955[2]. Although computers have not yet replicated human intelligence in a general sense, there are many tasks of increasing complexity that computers can already perform equally well or even better than the most gifted, well-trained humans. As detailed in [2], computers have been programmed to beat even the best human players of the game-show *Jeopardy!*, to write corporate earnings previews for *Forbes.com* that are indistinguishable from ones written by humans, and to diagnose breast cancer from images of tissue as good as or even better than pathologists can[3]. Such examples demonstrate the increasing practicality of artificial intelligence, but what about understanding users' needs? Is it possible for a computer or computing system (including various sensors) to know what its user needs or wants before he or she makes any keystroke or swipes any touchscreen? Such a system would be considered by many to exhibit a high level of artificial intelligence.

[2] Although McCarthy is usually credited with coining the term artificial intelligence, we note that its first usage in the literature was a paper co-authored by McCarthy, Minsky, Rochester, and Shannon [7]. Therefore, it is not entirely clear who first came up with this term, and in an interview even McCarthy himself could not recall [8].

[3] To be precise, what Brynjolfsson and McAfee describe is a system, known as C-Path, that helped to diagnose breast cancer and also identified new features of breast cancer tissue that were shown to be good features for predicting survival.

1.2 Research Questions and Scope

The goal stated above is ambitious and open-ended. To narrow it slightly, this thesis aims to improve the state-of-the-art in a computer's ability to understand situations or contexts that humans find themselves in. Mobile computing researchers have adopted the term *context awareness* to refer to this ability. In other domains, such as aviation, maritime, and military domains, the term used is *situational awareness* (or *situation awareness*)[4]. The science related to context awareness (or situation awareness) is vast. Therefore, we have focused on one particular application area, navigation, where context awareness may be applied.

In particular, this thesis will be organized around a central research question:

- What are the benefits and constraints of introducing context awareness in navigation?

Highly-respected navigation researcher Dr. Paul Groves has called context one of the key challenges for the next generation of navigation technologies [9], and we share this view.

A secondary research question addressed in this thesis is:

- How can machine learning be used to build context or situation awareness, in order to solve problems in navigation?

Given that machine learning has been successful in many other related application areas, our hypothesis is that machine learning will prove to be an effective tool for building context awareness for navigation applications.

By focusing on navigation, we have limited the scope of research to a reasonably-sized domain. That being said, improvements in the state-of-the-art in context awareness have wide-ranging applications, and it is our hope that the few applications described in this thesis are seen only as examples and not as end goals in themselves. There are a wide range of rich mobile applications that can be enabled with the aid of context awareness, and researchers have identified context awareness as one of the key open issues in mobile computing research [10].

[4] For consistency, in this thesis we primarily use the term context awareness, but it can be considered synonymous with the term situation(al) awareness.

1.3 Key Issues in Navigation Research

Before we attempt to answer the above research questions, it would be prudent to define what we mean by *navigation* and discuss some of the key issues in navigation research. General principles in navigation will be discussed in more detail in Chapter 2.

No universally agreed definition of navigation exists [11]. In general, the definition of navigation varies by industry. For example, the maritime, aerospace, and road transport industries each have their own views of what is meant by navigation. We mostly follow the delineation of navigation described in [11], which divides navigation into two distinct concepts:

1. the determination of the position and velocity of a moving body with respect to a known reference point.

2. the planning and maintenance of a course from one location to another, avoiding obstacles and collisions.

The first concept is relatively straight-forward compared to the second. Over the years, many techniques have been developed for determining the position and velocity of moving bodies. Today, however, the “gold standard” method for determining position and velocity is the use of Global Navigation Satellite System (GNSS). GNSS receivers, which have in recent years become relatively inexpensive and small, are capable of achieving position accuracy of below 10 m under most circumstances worldwide. Velocity measurement accuracy, depending on the technique used, can be on the order of a few cm/s [12]. The drawback of GNSS is that it requires the acquisition and tracking of weak radio signals broadcast from space. If the view to the sky (from the receiver’s perspective) is significantly obstructed by, e.g. structures or vegetation, then the accuracy will degrade, and in the worst circumstances no position or velocity solution will be obtained. For example, in most indoor environments, unless the receiver is near a window or other glass structure, it will be unable to perform positioning or velocity determination.

Therefore, one of the key research issues in navigation research with respect to the first item above is developing methods for position and velocity determination in indoor and highly urban environments. There is not a single positioning technique that works well in *all* environments, and thus solving the problem of ubiquitous positioning requires a hybrid approach. Due to this fact, a ubiquitous positioning system must also be capable of knowing when to utilize GNSS vs. some other method. This is one of the areas where context awareness can benefit navigation.

The latter concept of "planning and maintenance of a course" includes topics such as route optimization, navigation visualization (e.g. turn-by-turn navigation), and hazard detection and avoidance. In particular route optimization is an important topic in navigation, and there are a number of active research topics in this area, e.g. dynamic optimization of road journeys to avoid traffic, safety optimization, fleet management, etc. In certain contexts, "planning and maintenance of a course" may include determining the required maneuvers and associated vehicle parameters for achieving those maneuvers (e.g. in maritime navigation, the ship's rudder positions and engine power, etc.). Lastly, in the context of modern Location-Based Services (LBS), this latter concept has come to include many ancillary functions that were not part of traditional navigation systems. One example is the search and retrieval of information concerning possible destinations. Especially in the case of navigation being integrated into mobile devices, the dividing lines between "navigation" and various other LBS functions is becoming increasingly blurred. For example, location "check-in" services are being integrated into navigation applications, although a check-in does not strictly fit into the above definition.

Finally, one active research area which combines the above two concepts is Simultanous Localization and Mapping (SLAM), where the goal is to determine a vehicle or pedestrian's position (and velocity) while at the same time produce a map of an unknown environment.

Given the wide gamut of functions that navigation systems must perform, depending on the application, one of the other key issues in navigation is how can a navigation system gather and maintain a "picture of the world" such that it has complete and up-to-date information about how to best perform all of these functions. In this sense, context awareness is ideally suited for navigation because it can provide this picture of the world.

1.4 Research Methodology

The research methodology adopted for this thesis varied according to several distinct phases. In the first phase, a wide literature review was conducted on the subject of context awareness. The results of this literature review are presented primarily in Section 4.2 but also in the included publications. In the second phase of our research, we aimed to synthesize the results of our literature review and formulate a general theoretical framework for contextual information and contextual reasoning. This work is reflected in Chapter 4 and publications [P1] and [P2]. Finally, the third phase of our research focused on tackling three different research tasks related to introducing context awareness into navigation applications. Determining how to best accomplish these tasks required different research methods, depending on the task, although the first two tasks were more closely linked compared to the third. The tasks were: (1) to recognize the activity of a smartphone user in an indoor office environment, (2) to recognize the mode of motion that a smartphone user is undergoing outdoors, and (3) to determine the optimal path of a ship traveling through ice-covered waters. These tasks are very different from one another, especially the third task with respect to the first two, demonstrating the breadth of problems encompassed by the topic of context awareness in the field of navigation. They also demonstrate wildly different aspects of "understanding users' needs" for different types of users.

The first task provides possible enhancements for a navigation or position tracking system that must work also indoors. As discussed briefly above, reliable and everywhere-available indoor positioning is one of the biggest unsolved problems in navigation. Context awareness is one way to help solve this problem. For example, if the system detects that a user is sitting and working in a static position (e.g. seated at a desk), then it can apply a positioning filter that assumes little or no changes in user position (and perhaps go into a low-power-consumption mode), but when it detects that the user has stood up, it can change the filter to one that assumes greater possibilities for movement. If the system later detects that the user has done some routine activity, e.g. fetched a fresh cup of coffee, it can apply a post-processing filter to refine the position tracking history, perhaps removing outliers or some other desired refinement.

Similarly, the second task is important for navigation because a navigation system designed to work outdoors can adapt and improve its performance based on the motion mode in which it is used, but it would be easier if the user did not have to manually change the modes of the navigation system when he or she transitions, e.g. from walking to driving. In other words, a context-aware navigation system would automatically know that a pedestrian user needs a pedestrian navigation system and a driving user needs a car navigation system; it would adapt itself automatically according to these different needs.

The third task is a rather classic problem in maritime navigation, but surprisingly this function has been and continues to be performed in a manual way (i.e. the ship captain or navigator manually choosing the route based on ice charts, local observations, and experience). It is also becoming increasingly important to find efficient paths through ice-covered waters due to the opening up of northern sea routes, as well as increased wintertime maritime transport in general (e.g. in the Baltic Sea). In terms of understanding the users' needs, this capability means that if maritime conditions change, such that the captain or navigator needs to alter the ship's route (based on changing ice conditions or other factors), an "ice-aware" navigation system could automatically inform the ship's crew that a new route is recommended and even suggest the optimal route to the crew.

For the first and second tasks, we utilized a method from machine learning called supervised learning, which is a quantitative research method. Supervised learning will be discussed in more detail in Section 3.3, but in brief this method uses labeled "training data" to measure the performance of a learning process. In the context these two tasks, the learning task was to recognize activities and motion modes of smartphone users. Thus, the performance on this learning task was measured by comparing the inferred activity/motion with the actual activity/motion using labeled data.

The third and final task was investigated using feasibility assessment, in which we investigated the viability of developing an ice-aware maritime route optimization system. In this research work, we studied the relevant state-of-the-art and proposed a method for performing the desired route optimization. In this sense, our work can be considered as constructive research. The chosen method is based on graph theory and a breadth-first search algorithm. Finally, we implemented the method in software and validated it by comparing the computed routes with historical routes. More details on the methodology for this task are found in [P5].

1.5 Main Contributions

This thesis explores an important link between machine learning and context awareness and exploits this link to demonstrate possible applications in the field of navigation. Firstly, the author has developed a generic conceptual framework for the multi-step processing of raw sensor data into contextual information, which had been largely lacking in the literature. Also, many studies on context awareness either focus on a narrow area of context (e.g. [13] [14] [15] [16]) or do not provide any clear framework or mechanism of how to encode a situation or context in a systematic and comprehensive way (e.g. [17] [18] [19] [20] [21]). This thesis proposes and describes a simple but powerful framework for describing a context in terms of seven key questions, covered further in Chapter 4 and [P1]. Together, these two conceptual frameworks benefit the research community by making the abstract and ambiguous concepts "context" and "context awareness" more concrete and clearly defined, as well as providing a methodological skeleton on which to build context-aware systems.

In addition, this thesis examines three separate use case scenarios or applications of context awareness related to the field of navigation. These correspond to the three tasks described in Section 1.2 above. The remainder of this section describes the key contributions related to these use case scenarios.

Firstly, the thesis presents a probabilistic Location-Motion-Context (LoMoCo) model, combining location and motion context, used to detect human behavior (i.e. activities) in an indoor office environment. The sensors used to detect the human behavior include only sensors available in commercial-off-the-shelf (COTS) smartphones, as well as access points in a wireless local area network (WLAN) used for the positioning component. To our knowledge, this is the first study focused on detecting office-environment activities that utilizes only smartphone-based sensors and standard WLAN access points. This is significant because earlier studies mostly relied on the installation of custom-designed sensors in the office environment or wearable sensors that are not in common use in offices. For example, [22] relies on sensors installed in an office chair and multiple cameras installed in an office room to infer activity. As smartphones and WLAN access points are already widely present in office environments around the world, the results of this research have more potential for widespread application. Our method can be used anywhere within an office building where WLAN signals are present, provided the user has a smartphone.

A problem related to the above topic is the determination of whether a smartphone user is indoors or outdoors. This is important contextual information because the optimal positioning system differs depending on whether the user is indoors or outdoors. Another important benefit of this contextual information is that it can be used to conserve smartphone battery usage. Outdoor positioning systems, namely those based on Global Navigation Satellite Systems (GNSS), are power intensive and should be turned off automatically when the user is indoors. The method described in this thesis for indoor-outdoor determination is, according to our knowledge, the first smartphone-based probabilistic indoor-outdoor method described in the literature. A similar method was published later as a patent application [23].

Next, this thesis includes a systematic evaluation of a large number of machine learning algorithms applied to the problem of detecting "mobility contexts", including consideration of the computational cost of the resulting classifiers, due to their intended use in resource-limited mobile devices. The number of algorithms investigated and applied to this problem is larger than any other previous study, according to our knowledge. Also, most existing studies dealing with mobility context do not consider or evaluate the computational cost of classifiers, so our study is novel in this aspect.

Furthermore, our study is the first research on mobility context to utilize GNSS, accelerometers, and information from Geographic Information Systems (GIS) for the purposes of detecting mobility context[5]. A similar study utilized GNSS and GIS information but not accelerometers [13]. In particular, GIS is an important source of information for detecting mobility context because it can be used to determine proximity to relevant landmarks, such as train stations and bus stops. Most earlier studies do not consider this important source of semantic information, and our research provides strong evidence, as a result of feature selection, that such information improves the context recognition result.

The main contribution of this study was to measure the relative performance of many different types of classification algorithms applied to this particular machine learning problem. Using default parameter values, the best performance was achieved using

[5] This research was first published as a conference paper in 2013 (see [24]). The publication included in this thesis is an extended version of this earlier study.

the RandomForest algorithm. We also studied the influence of parameter tuning for the RandomForest algorithm. After parameter tuning, we achieved an average recall rate of >97.5% for our test data. We are not aware of any other study achieving this level of performance for a comparable classification problem. One limitation of this study is that exhaustive parameter tuning was not performed for every algorithm type. Therefore, it is possible that other algorithms, after tuning, may achieve similar or even superior performance.

Lastly, for the purposes of developing an "ice-aware" maritime navigation system, we developed a novel method for route optimization. Compared to earlier works in this area, our method is only the second graph-based approach to the problem of route optimization through ice-covered waters. Compared to the earlier graph-based approach, described in [25], our method is more computationally efficient, since it uses the A* algorithm rather than Dijkstra's algorithm. We note that in [25] only a few tens of nodes were considered in the route optimization examples given, so computational complexity was perhaps not an apparent issue. To find truly optimal routes over large distances, however, it is necessary to consider thousands of nodes or more. [26], published later than [P5], also used the A* algorithm. Compared to [26], we incorporated into our method an operational constraint related to ice breaker assistance, whereas the cost function used in [26] did not consider this issue. Another advantage of our cost function is that the cost, expressed in the unit of time, is easier to interpret. The cost function employed in [26] is a linear combination of four different variables, and its physical meaning is difficult to interpret.

An earlier study tackling ice-aware route optimization expressed the problem as a differential equation and used numerical methods to solve it, such as Powell's method [27]. Such methods, however, do not guarantee a global optimum. Due to the complex nature of ice fields, local minima can be significantly worse than the global optimum. The benefit of a graph-based approach is that shortest-path algorithms exist that can guarantee an optimal solution. The main novelties in our method are the design of a suitable graph structure that provides a reasonable trade-off between realistic modeling of ship motion and computational complexity, as well as the incorporation of ice breaker assistance into the cost function used in optimization.

1.6 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 provides background on the most important principles in navigation relevant to this thesis. Chapter 3 provides an overview of the principles of machine learning relevant to this thesis. Chapter 4 provides a background on context awareness and summarizes our approach to introduce context awareness to navigation research. Chapter 5 provides an overview of the included publications and summarizes the results. Finally, Chapter 6 offers some conclusions that can be drawn from this thesis work and provides some suggestions for future areas of research and development related to context awareness for navigation applications.

2. PRINCIPLES OF NAVIGATION

This chapter contains a concise overview of the principles of navigation. As stated in Section 1.3, no universally agreed definition of navigation exists [11]. We have adopted the definition of navigation described in [11], which divides navigation into two distinct concepts:

1. the determination of the position and velocity of a moving body with respect to a known reference point.
2. the planning and maintenance of a course from one location to another, avoiding obstacles and collisions.

We have divided this chapter into three sections. Section 2.1 describes the navigation system in general. Section 2.2 describes the first part of navigation, i.e. methods for determining position and velocity. Finally, Section 2.3 describes the functions of navigation falling under the second category of "planning and maintenance of a course..."

2.1 The Navigation System

A *navigation system* is a system used to perform or assist in the functions described above. In modern terms, it is a digital system consisting of one or more computational units, memory units, navigation sensors, and in many cases, a user input and display unit. A navigation system may communicate with other systems in a vehicle through clearly-defined interfaces. In many navigation systems, particular those integrated into mobile devices, the system communicates over a wireless network to obtain assistance data, map data, or other ancillary data. Figure 2.1 shows a block diagram of a typical navigation system.

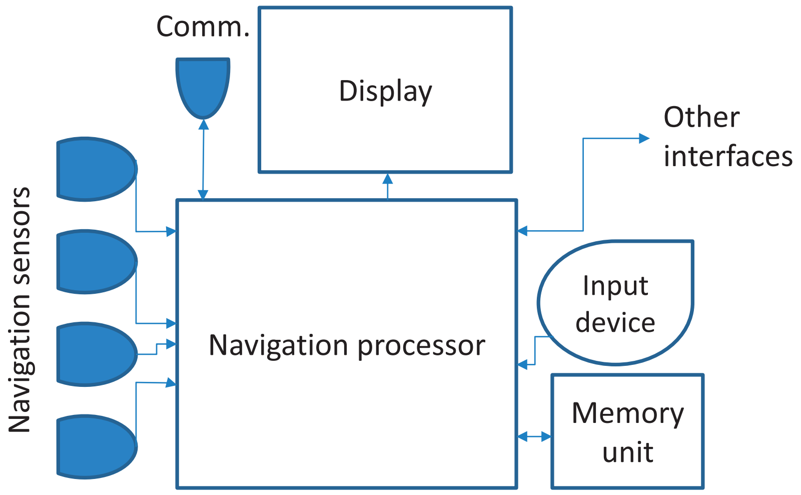


Fig. 2.1: *Block diagram of a navigation system.*

The primary output of the navigation system is known as the *navigation solution*, which consists of (1) the position and (2) velocity of the vehicle or object which is being tracked. A third output, often considered part of the navigation solution, is time. This is due mainly to the fact that GNSS receivers compute time as an intermediate step to determining position. Since a time solution is also needed in many navigation systems, it is natural to include time when speaking of the navigation solution. Finally, some navigation systems also output the vehicle's attitude (e.g. roll, pitch, and yaw), especially in cases where the navigation system includes an Inertial Measurement Unit (IMU).

In some cases, navigation systems may be designed to track objects remotely, but in most cases the navigation system is physically attached to the vehicle or object being tracked. In the case of mobile device positioning, the navigation system is integrated with the mobile device, so there is no separate display, input device, or communications channel. In fact, most navigation systems in mobile devices utilize the host processor and memory of the mobile device, so the integration is indeed very tight. In this context, the mobile device is the object being tracked, but since the mobile device is normally close to the user, we often consider the user to be the object being tracked.

The system requirements for a navigation system vary greatly depending on the application. Different requirements may include accuracy, rate of navigation solution, mass, size, cost, reliability, and various functional requirements (route planning, display, etc.). For example, in aviation and maritime applications, the emphasis is on *integrity*, i.e. ensuring the navigation solution is within the stated error bounds and rapidly informing the user whenever the desired accuracy cannot be ensured. In consumer applications, the emphasis may be on delivering acceptable performance at the lowest possible cost. Due to these varying requirements, there is a great variety of products and solutions in the navigation industry, ranging from navigation systems where almost all the functions are integrated into a single Integrated Circuit (IC) to large, complex, and expensive systems such as those found on submarines.

2.2 Methods for Determining Position and Velocity

In this section we will briefly describe the major methods available for determining position and velocity. Due to space limitations, we cannot cover all such methods, but we will highlight the most important ones.

2.2.1 Trilateration

Today the most commonly used method for determining position is known as *trilateration*. This method, although in use since the late 1940s, was made famous by GPS and other GNSS, which employ this technique. Trilateration uses a set of distance measurements from an unknown position to reference objects to determine the unknown position[1]. The distance measurements are often referred to as *ranges*. In two-dimensional space, a range to a single reference object constrains the unknown position to a disk (or annulus). The width of the disk is determined by the amount of error in the range measurement. Ranges to two reference objects constrain the position to within one region or at most two mirrored contiguous regions (see upper-right of Figure 2.2). By making range measurements to three or more reference objects, the unknown position can be constrained to a small two-dimensional contiguous space, as illustrated in the lower-right of Figure 2.2.

Extending this concept to three-dimensional space, each range measurement constrains an unknown position to within a spherical shell. Two spherical shells

[1] The positions of the reference objects are known to a high level of accuracy.

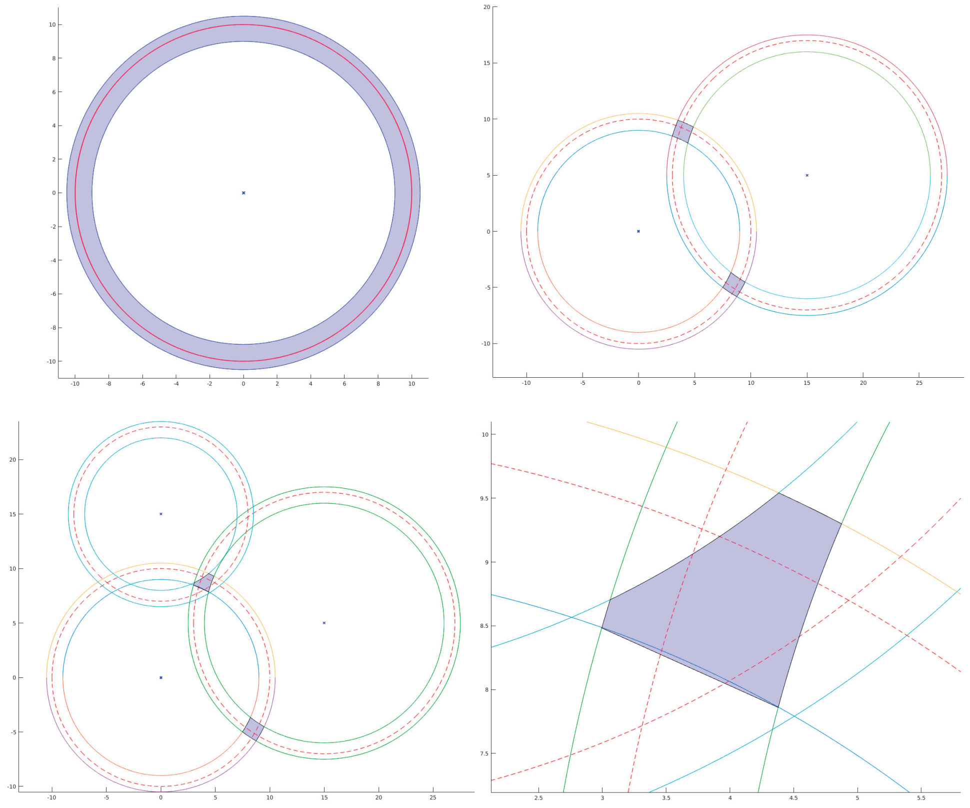


Fig. 2.2: *Example of trilateration using ranging in two dimensions. The final constrained position space is shown in the lower-right image.*

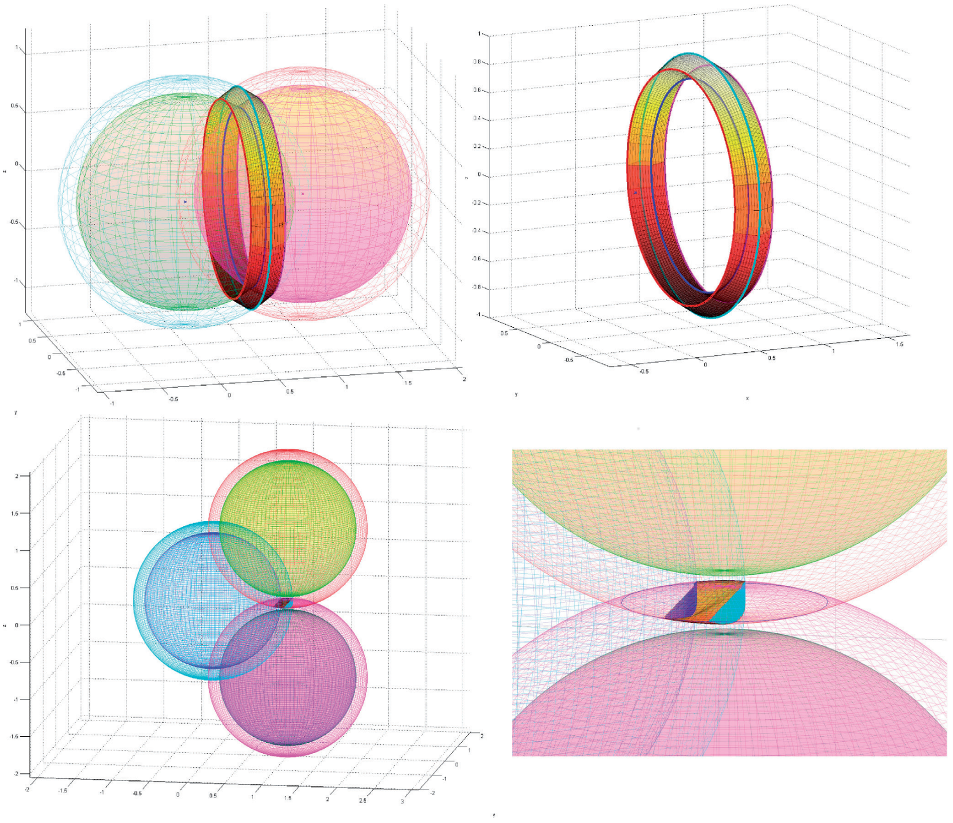


Fig. 2.3: *Example of trilateration using ranging in three dimensions. The final constrained position space is shown in the lower-right image.*

intersect in a three-dimensional shape known as a toroid (see upper part of Figure 2.3, and two toroids, formed from a set of three range measurements, can intersect at most in two mirrored contiguous volumes. In Figure 2.3 for simplicity, we illustrate the case where the two toroids intersect to form one contiguous volume. In practice, if any such “mirror” ambiguities exist, it is usually possible to disregard one of the two volumes because it is, e.g. far from the Earth’s surface.

This is the principle upon which GPS and other GNSS are based, although the reality of how these systems are implemented is a bit more complicated. A GNSS receiver measures radio signals from satellites whose position can be determined using orbital parameters. The radio signals contain time signals that can be used to determine the time it took for the signals to propagate from the satellite to the receiver, and because these signals travel at a known, nearly constant velocity (the speed of light), the travel time measurements can be converted to distance. The time measurements, however, contain various biases, the most important of which is the time bias between the GNSS system time and the receiver clock, known as the receiver clock bias. For this reason, GNSS receivers require measurements from four or more satellites to determine a position. The reason that the required number is four and not three, as was depicted above, is because the receiver must also estimate the receiver clock bias.

Because it is well-known that this bias (and other biases) are contained in the measurements taken from the satellites, these measurements are distinguished from the true range and referred to as *pseudoranges*, defined as:

$$\rho \equiv r + ct_u \tag{1}$$

with

$$r = \sqrt{(x_s - x_u)^2 + (y_s - y_u)^2 + (z_s - z_u)^2} \tag{2}$$

where r is the true range, (x_s, y_s, z_s) are the coordinates of the satellite, (x_u, y_u, z_u) are the coordinates of the unknown position, t_u is the receiver clock bias, and c is the speed of light.

Because t_u is common in all pseudorange measurements, its value $\big($along with $(x_u, y_u, z_u)\big)$ can be solved using the following system of equations:

$$\begin{aligned}
\rho_1 &= \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} + ct_u \\
\rho_2 &= \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} + ct_u \\
\rho_3 &= \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} + ct_u \\
\rho_4 &= \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} + ct_u
\end{aligned} \tag{3}$$

where $\{\rho_i\}_{i=1}^{4}$ are the four pseudorange measurements and $\{(x_i, y_i, z_i)\}_{i=1}^{4}$ are the coordinates of the four reference objects.

Neglected from the above equations are other sources of error which are not necessarily common to all pseudorange measurements, such as ionospheric and tropospheric delays. Although there exist closed-form solutions to the pseudorange equations that take into account these additional error sources, these generally require more than four satellites in order to perform a linear regression and also to estimate the error covariance matrix [28]. For this reason, it is more common to use iterative techniques based on linearization or another estimation algorithm, such as the Kalman filter. More details on how to compute the navigation solution can be found in [29] or [30].

Regarding velocity determination, recall that velocity is the time derivative of position; if the position at two epochs can be estimated and the time between the two epochs is known, then the velocity can be estimated as $d/\Delta T$, where d is the distance between the two positions and ΔT is the time difference. As GNSS receivers also produce a time solution, it is possible to estimate the velocity in this way. This is a simple and occasionally employed approach. Much better velocity accuracy can be obtained, however, by utilizing the *Doppler shift* measurements that a GNSS receiver must inherently measure.

The *Doppler effect* is a change in perceived frequency of a wave due to the relative velocity between the object generating the wave and an observer. Because the satellites are virtually always moving with respect to a GNSS receiver, the receiver must measure this shift in frequency, known as the Doppler shift, in order to maintain track of the signal. The Doppler shift can be converted into a pseudorange rate, and given the pseudorange rates from four or more satellites, the receiver's velocity can be computed. The equations to obtain pseudorange rates can be obtained by differentiating Eq. 3. The details of Doppler-based velocity estimation, however, are beyond the scope of this chapter, and we refer the interested reader to [31]. It will suffice to say that when velocity is estimated using this approach, the accuracy improves several orders of magnitude compared to estimating velocity using simple differencing.

2.2.2 Dead Reckoning

Perhaps the second most common method for position determination is known as *dead reckoning*. Dead reckoning is a form of *relative positioning*, where the current position is estimated iteratively from the known or estimated position of the previous epoch. The basic equations of dead reckoning in two dimensions are the following:

$$\begin{aligned} N_{i+1} &= N_i + s_i \Delta t \cdot \cos \alpha_i \\ E_{i+1} &= E_i + s_i \Delta t \cdot \sin \alpha_i \end{aligned} \tag{4}$$

where (N_i, E_i) are the coordinates in a local geodetic coordinate system at epoch i, s_i is the speed at epoch i, Δt is the time difference between epochs i and $i + 1$,

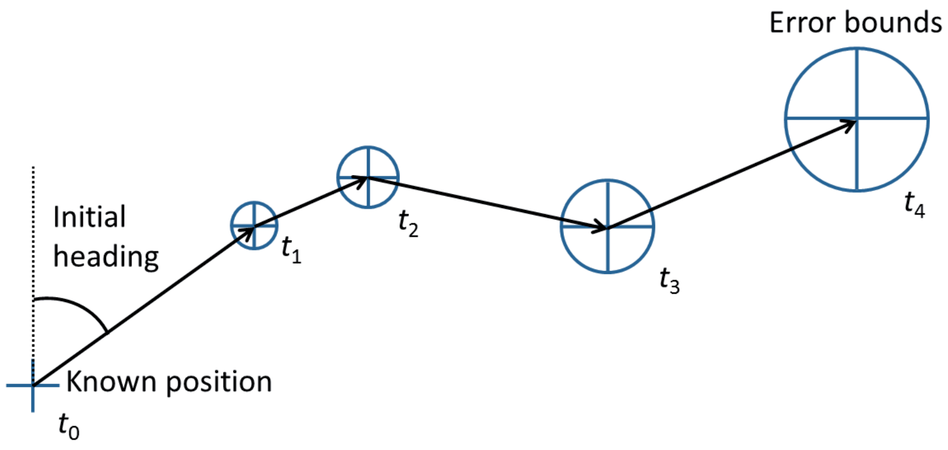


Fig. 2.4: *Illustration of the principle of dead reckoning after [11]. The error bounds of the estimated position increase with time.*

and α_i is the azimuth (or heading) with respect to North. In three-dimensions, the equations are similar, but one must measure also the elevation with respect to the Earth's surface.

One of the main drawbacks of dead reckoning is that the error in the estimated position will increase with respect to time and distance traveled. This principle is illustrated in Figure 2.4.

Nonetheless, dead reckoning is a commonly employed positioning method whenever an absolute positioning method is not available. It is also often used when a positioning solution is needed at a higher rate than that which the absolute positioning system can achieve. For example, it is common that low-end commercial GNSS receivers can only compute a navigation solution at 1 Hz. If higher solutions rates are needed, a dead reckoning method with a Δt less than 1 second may be desired.

There are many measurement techniques available to provide the velocity and heading measurements needed to perform dead reckoning. One of the most commonly used measurement devices is an *Inertial Measurement Unit (IMU)*, which consists of a set of *inertial sensors*, usually consisting of three orthogonal accelerometers and three orthogonal gyroscopes. The use of IMUs for navigation is known as *inertial navigation* and the complete system consisting of an IMU, a navigation processor, and associated hardware and software is known as an Inertial Navigation System (INS). The details of inertial navigation are beyond the scope of this chapter, but we refer the interested reader to [11]. INSs range in performance and cost. The most accurate but expensive INSs may cost in excess of $100,000 and can achieve positioning accuracy of about 1 km after 1 hour of operation[2]. Less expensive INSs based on MEMS technology may cost as little as $100, but they have much worse performance.

Because dead reckoning requires measurements of speed and heading, this method of positioning inherently includes velocity determination.

2.2.3 Positioning Based on Pattern Matching

The final method for positioning that we will discuss in this chapter is positioning based on pattern matching. This method is often known as "fingerprinting" for reasons that will soon become apparent. The basic principle in positioning based on pattern matching is that *a priori* spatially-correlated information about the environment is used to estimate the position based from an observed signal at an unknown location. This is done by matching the observed signal against a database of previously observed signals at known locations or a pre-computed signal-spatial model, thus exploiting the spatial-correlation of the signals. There are many possible methods for performing pattern matching against a database of signals, but a common approach is to find the closest match in "signal space" according to some *distance metric*. For example, consider the following two sets of measured signals:

$$s_1 = \left\{ \begin{array}{c} \text{a=24} \\ \text{b=36} \\ \text{c=14} \\ \text{d=56} \end{array} \right\} \quad s_2 = \left\{ \begin{array}{c} \text{a=13} \\ \text{b=26} \\ \text{c=34} \\ \text{d=55} \end{array} \right\}$$

Each of these sets of measurements represents a signature or "fingerprint" of the location where they were obtained. If the measurements are similar (i.e. distance in signal space is small), then the two fingerprints are considered a match. A commonly-used distance metric is the Euclidean distance, which for this example is given by:

[2] The positioning performance will depend significantly on the type of motion and speeds experienced.

$$d = \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2 + (c_i - c_j)^2 + (d_i - d_j)^2} \tag{5}$$

where $\{a_i, b_i, c_i, d_i\}$ and $\{a_j, b_j, c_j, d_j\}$ are the four signal measurements in two measurement sets s_i ands_j, respectively. Such signals can be queried against a database, and the best match according to this metric can be retrieved, or a more elaborate algorithm such as k-Nearest Neighbor (kNN)-regression can be used.

In addition to the above approach based on a signal measurement at a single epoch, some techniques utilize a sequence of measurements. For example, [32] uses a time-series of magnetic field measurements to approximate the unknown starting position using a Monte Carlo Localization (MCL) technique. The authors found that in many indoor environments, positioning based on magnetic field could be obtained with sub-meter accuracy after moving about 10 m in a straight line within the environment.

Many possible signals can be used to perform this type of positioning, but some of the more commonly used ones are WLAN received signal strength, other radio signals, magnetic signals, or even images. WLAN-based fingerprinting is perhaps the most common approach in indoor environments.

A drawback to positioning based on pattern matching is velocity determination can only be performed using the simple differencing method described in Section 2.2.2. In cases where the target being tracked is a pedestrian, an alternative approach for velocity determination is *step detection*, where the pedestrian's steps are detected using, e.g. inertial sensors. In this case, the user velocity can be determined within a few 10s of cm/s, provided a good estimate of the user's step-length is available. See [30] for more details on step detection.

2.3 Functions Related to Course Planning and Maintenance

As discussed in Section 1.3, there are a wide range of functions related to "the planning and maintenance of a course from one location to another, avoiding obstacles and collisions." In this section we will describe only a few of the most common and important ones relevant to this thesis.

2.3.1 Route Optimization

Route optimization, also referred to as course planning, path planning, route planning, or simply routing, is the process of determining the optimal route or path that a vehicle or person (or set of vehicles or persons) should take to arrive at a desired destination or otherwise complete some desired task. The definition of optimal can vary greatly depending on the application. For example, one may wish to minimize the travel time, minimize fuel or other costs, maximize safety (i.e. minimize risk), or perhaps optimize other characteristics of the journey such as comfort of the passengers or "providing the most scenic route." Fleet optimization is another topic that is intrinsically linked to route optimization.

Historically route optimization has been primarily a manual task. Navigators or other experienced persons would plan a route based on paper maps and detailed knowledge of the environment and the vehicle. This has been one of the primary roles of "navigators" in maritime shipping and aviation, for example. Algorithmic route optimization, nonetheless, has a rather long history. For example, the "truck dispatching problem" was studied by Dantzig and Ramser more than 50 years ago [33]. Going even further back, the "traveling salesman problem," introduced in its early form in the 1800s, can be seen as a route optimization problem. It is only within the past few decades, however, that computerized route optimization has been realized in practice for navigation applications.

Many different approaches to route optimization have been developed. To tackle route optimization, many techniques from the general field of mathematical optimization can be applied, and the main difference between route optimization methods lies in how the problem is formulated. One approach is to model the environment using graph theory, where different locations are represented as nodes in a graph and the edges or vertices in the graph are associated with a cost to travel between the nodes. This is the approach we took in [P5]. The interpretation of "cost" can vary according to the optimization criteria, e.g. it may defined according to travel time, fuel consumption, risk, or even some multi-modal criteria. Another approach comes from optimal control theory, where the cost is modeled as a function of initial conditions, speed, geometry, vehicle controls, and environmental factors. The problem is then to minimize a continuous-time cost function subject a set of constraints and boundary conditions. Yet another approach is to consider a route as a particular point in a space of $3n$ dimensions, where n is the number of waypoints in the route [27]. Each waypoint has two spatial coordinates and one temporal coordinate[3]. The goal is then to find a point in this $3n$-space that minimizes a cost function. Analytic or numeric methods can be used, but in many cases the cost function is complex and its derivative cannot be solved analytically. The drawback of numeric approaches is that, in most cases, they cannot guarantee global optimality.

2.3.2 Visualization for Navigation

In order to "plan and maintain a course," it may be necessary to provide a visualization of the navigation situation. This is especially the case when there is a human-in-the-loop, but visualization is of relevance even in the case of autonomous robotics, since usually there is a desire for humans to check the computed route or analyze the performance afterwards. Visualizing the navigation situation touches on many topics in HCI, such as cartography, interface design, and geospatial visualization. One topic that is trending in navigation is 3D navigation visualization. Virtual reality and augmented reality are related research areas that have strong ties to navigation.

We mentioned SLAM already in Section 1.3, another trending topic in navigation that is linked to visualization. The goal in SLAM is not only to map and visualize an unknown environment but to use the continuously refined map to constrain the vehicle or person's position. The visualization may consist of a simple 2D map, e.g. the floor plan of a building, but the interesting part from a visualization perspective is how to best represent uncertainty in the map.

Another interesting topic related to navigation visualization and relevant to this thesis is visualization of context or situation awareness. This is a rather new research topic, although some commercial solutions for visualizing situation awareness already exist in the marketplace (e.g. [34]).

[3] For aircraft or spacecraft, this model can be extended to three spatial coordinates, and the problem is therefore solved in $4n$-dimensional space.

2.3.3 Hazard Detection and Avoidance

The last phrase in the latter concept from navigation presented above is “avoiding obstacles and collisions.” This is the role of Hazard Detection and Avoidance (HDA). HDA is most prevalent in aviation and spaceflight, but it is also an important system function in maritime transportation and robotics in general. With the increasing research and development in autonomous cars, HDA will continue to grow in importance. Context awareness has clear relevance with respect to HDA, since the existence of hazards can be considered one aspect of context.

Perhaps the most developed application of HDA is in the aviation industry, where such systems are also known as Airborne Collision Avoidance Systems (ACAS). One particular implementation of ACAS standards is the Traffic Alert and Collision Avoidance System (TCAS). Modern aircraft are equipped with several ACAS modules, including Ground Collision Avoidance Technology (GCAT), which warns pilots when they are danger of ground collisions and in some cases may take control of the aircraft to avoid a collision, and also a Midair Collision Avoidance System (MCAS). The International Civil Aviation Organization (ICAO) mandates that larger aircraft are equipped with a TCAS, which consists of a transponder that broadcasts the aircraft’s position, speed, and heading. The transponder can also receive and decode radio signals from other aircraft, and these signals are used to warn the pilots if another aircraft comes within a protected volume around their own aircraft. The size of the protected volume depends on the altitude, speed, and heading of the aircraft. If a potential collision is detected, the TCAS of two aircraft can communicate and automatically negotiate the best collision avoidance maneuvers. A related technology, known as Automatic Dependent Surveillance - Broadcast (ADS-B) broadcasts similar information, but it is mainly used for surveillance and tracking of aircraft by the air traffic control system.

Finally, an even more challenging area of HDA research is MCASes specifically designed for Unmanned Aerial Vehicles (UAVs). In the case where there is no human-in-the-loop, an effective design for MCAS for UAVs must be entirely autonomous, and reliable systems are required in order to integrate UAVs into non-segregated airspace. This remains an active area of research and development.

In terms of spacecraft, the main role of HDA is to detect and avoid orbital debris, which may be come from either natural sources, such as micrometeroids or other space material, or human-made sources such as debris from satellites and spent rocket stages. Particularly in Low-Earth Orbit (LEO) orbital debris is increasingly a huge danger to spacecraft. Due to several notable collisions between defunct satellites (or even intentional destruction of satellites using ballistic missiles), LEO is scattered with hundreds of thousands of pieces of debris. For example, the International Space Station (ISS) has had to make more than a dozen evasive maneuvers to avoid orbital debris in its history. Occasionally the crew are required to enter and seal a "lifeboat" when a imminent potential collision is detected, in order to minimize the risk to life. Orbital debris detection is conducted using both ground-based and space-based methods. The US Department of Defense (DoD) tracks more than 500,000 pieces of debris orbiting the Earth using mostly ground-based radar.

Finally, maritime transport is another application where HDA is applicable. In general, large ships are equipped with radar and sonar (i.e. echo sounders) to detect potential hazards, such as other ships, icebergs, ocean debris (such as floating shipping containers), and shallow waters. Since the 1980s, the International Maritime Organization (IMO) began requiring larger commercial ships to carry Automatic Radar Plotting Aids (ARPA), a type of computer system which interface with the ship's radar to aid in avoiding collisions with other ships. A newer technique more recently required on large commercial ships is the Automatic Identification System (AIS), which works in a similar way to ADS-B. Position, speed, and heading are broadcast to the vessel traffic services (VTS) and to other ships, and the received AIS data are typically integrated into the ship's Electronic Chart Display and Information System (ECDIS) and radar displays. AIS data also being increasingly integrated into ARPA.

3. PRINCIPLES OF MACHINE LEARNING

This chapter will provide background on the topic of machine learning, whose role in this thesis was outlined in the introduction chapter. We illustrate the main concepts using examples relevant to context awareness.

Our preferred definition of "machine learning research" was also given in the introduction chapter, but it is worth repeating here:

> Machine learning research seeks to develop computer systems that automatically improve their performance through experience [1].

Stated slightly differently, *machine learning* is concerned with developing and analyzing algorithms used by computer systems that automatically improve their performance through experience. An earlier definition, widely attributed to Arthur Samuel, is that machine learning is "the field of study that gives computers the ability to learn without being explicitly programmed"[1]. This definition also implies *automatic* learning, but it suffers from the problem that the meaning of "learn" is not precisely defined.

As is the case in some fields, the discipline known as "machine learning" has drifted somewhat from its original defining aims. This will become more evident later on in this chapter when we describe the major types of machine learning problems that have developed over the past 30+ years.

The chapter is organized as follows. Section 3.1 presents a historical perspective of machine learning. Section 3.2 provides an overview of the modern notion of machine learning. Section 3.3 describes supervised learning, and Section 3.4 describes unsupervised learning. Finally, Section 3.5 provides a few concluding remarks on the topic of machine learning.

[1] We have been unable to recover the original source of this quote. Some references cite [35], but the quote is not found in reprints of this article.

3.1 Roots of Machine Learning

Some of the earliest works in machine learning (and artificial intelligence in general) involved computer-based games, such as chess and checkers [36]. Although he did not use the term explicitly, many of the early ideas in machine learning were developed by Claude Shannon in a 1950 paper on computer chess [37][2]. Arthur Samuel further developed these ideas during the 1950s, but his preferred game was checkers [35]. Perhaps the most famous game-playing computer program was initiated at Carnegie Mellon University in 1985 for the game of chess. This project was later transitioned to IBM, culminating in the computer Deep Blue® beating the chess master Garry Kasparov in a six-game match-up in 1997[3]. Nowadays, similarly advanced chess programs can run on a personal computer or even a smartphone or tablet. A full literature review of computer chess is beyond the scope of this thesis, but we refer the interested reader to [38] [39], and [40]. Our intention in this section is use computer chess to illustrate that the early pioneers in machine learning aimed to develop computers that could *automatically* learn to perform a task through experience.

The strategy in developing Deep Blue was to encode into a program both the rules of chess, as well as the tactics and strategies of great chess masters, attempting to cover as many possible situations in chess, also known as *chess positions*, as possible. After a long period of development, Deep Blue eventually succeeded in beating the best human chess players. The main reason for this success was the ability of the computer to evaluate hundreds of millions of chess positions per second, whereas a human chess player relies less on computational power and more on intuition and experience to evaluate the strengths and weaknesses of different chess positions.

Examining the strategy by which Deep Blue was developed, this approach hardly fits the above definition of machine learning. To create Deep Blue required many years of highly "manual" work of programmers refining the set of strategy rules, testing the program against human players, and repeating this process. Thus, it falls short of the aim of *automatically* improving performance. Although this strategy

[2] According to colleagues' accounts, computer visionary Alan Turing also began considering computer chess during the 1940s.

[3] We note that Kasparov has accused IBM of cheating by letting human players intervene in one of the matches.

was ultimately successful in creating a master chess player, when reading Shannon or Samuel's papers on such subjects, it becomes clear that their aim was not simply solving the direct problem of playing chess or checkers, but rather they used these games as a means of demonstrating a completely revolutionary idea in computer science–that computers might be able to *learn* by themselves.

3.2 Modern Machine Learning

In this section, we describe the modern notion of machine learning, which, as we have already alluded to, has developed into something a bit different from what the early pioneers in machine learning had envisioned. That is, today there is a well-established community of machine learning researchers and practitioners whose focus is not entirely the same as what Shannon, Samuel, Mitchell, or other machine learning pioneers had in mind. Our intention in pointing this out is not to denigrate the discipline of machine learning as a whole but rather to emphasize those aspects of the discipline which fall short of the original goals of machine learning.

Let us first present a few other definitions of machine learning found in recent textbooks on the subject:

> Machine learning is programming computers to optimize a performance criterion using example data or past experience [41].

This definition appears quite close to that of [1], if we assume that "example data" can be generated automatically. This may be true in some cases, but in most methods described in [41], the example data are data that have been manually labeled with the "correct" value relative to the performance criterion that is to be optimized. Although programs using such methods can improve their performance by obtaining more example data, if the example data cannot be generated automatically, then the method would fall short of the definition of [1].

Another recent definition is given by [42], which defines machine learning as:

> a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (such as planning how to collect more data!).

This definition includes the "automatic" aspect, similar to [1], although we prefer the definition of [1] due to its simplicity. Also, note that there is no reference to improving performance with experience.

We note that some books on machine learning (e.g. [43]) omit to precisely define the concept, perhaps because it has come to encompass many diverse methods. One book goes so far as to explicitly refuse to define machine learning in any principled way:

> The kind of learning techniques explained in this book...are called machine learning without really presupposing any particular philosophical stance about what learning actually is [44].

Mitchell, on the other hand, also provides a precise definition of the concept of learning in the context of machine learning:

> A computer program is said to *learn* from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E* [45].

Continuing with formalisms, many learning tasks can be expressed in terms of learning a mathematical function between the inputs to the task and the desired outputs. In other words, the learning task is to find some optimal mapping between the inputs and the possible outputs. This can be expressed as follows:

$$f : \mathbf{x} \rightarrow y \tag{6}$$

where f is a function, $\mathbf{x}$ is a vector of inputs of arbitrary dimension, and y is an output with $y \in Y = \{y_1, y_2, ...y_m\}$, corresponding to the set of all possible outputs (which may or may not be finite).

Machine learning techniques differ mainly in how they express and learn this unknown function $f(\mathbf{x})$, also known as a *model*. They also differ in the form in which y (and therefore the set Y) are expressed. For example, Y may be a continuous range or a finite, discrete set. When the learning task involves a continuous-valued output value, it is called *regression*.

When the output variable is discrete, we call it *classification*, since the possible values generally represent different *classes* or categories. In this thesis, our goal was to output *context*, and most aspects of context (e.g. motion modes and activities) were represented as discrete sets of classes, but in the case of [P5], the ice environment was described mostly with continuous variables.

In some cases, the output of the model may be best represented in probabilistic form. In such cases, the machine learning algorithm actually estimates the conditional probability $p(y|\mathbf{x})$. This distribution, $p(y|\mathbf{x})$, may be intrinsically important to the application at hand, or it may be an intermediate step towards determining the most likely value of y according to:

$$y = \arg\max_{y \in Y} p(y|\mathbf{x}) \tag{7}$$

This is known as the maximum a posteriori (MAP) estimate of y. One of the benefits of estimating $p(y|\mathbf{x})$ is that it provides a measure of the confidence of the output y.

Algorithms designed to learn $p(y|\mathbf{x})$ are known as *discriminative* approaches. An alternative approach is to first learn a model of the joint probability $p(y, \mathbf{x})$ and then condition on $\mathbf{x}$ to derive $p(y|\mathbf{x})$. These are known as *generative* approaches.

Apart from the distinctions regression vs. classification and discriminative vs. generative, there are two main categories of machine learning techniques, based on how the unknown function f is learned or approximated. The first category is known as *supervised learning*. In supervised learning, a "trainer" supervises the learning process. The goal is essentially then to transfer the knowledge of the trainer or supervisor in the form of a mathematical or computerized model. More details on supervised learning will be covered in Section 3.3.

The other main category is known as *unsupervised learning*. In unsupervised learning, the learning process is not guided in any significant way. The goal is essentially to uncover patterns that are implicit in the data but unobvious. Unsupervised learning can be considered automatic, but what can be very challenging in unsupervised learning is to define a notion of performance (recall the definition of machine learning given at the start of the chapter). In this thesis, we have focused primarily on supervised learning, but some research on context awareness also uses unsupervised learning, so we present one example from unsupervised learning in Section 3.4 to illustrate its potential role in context awareness.

3.3 Supervised Learning

As stated above, supervised learning uses a “trainer” to supervise the learning process. In most cases, the trainer has encoded his or her knowledge in the form of *labeled data*, also known as *training data*. In terms of the function f expressed above, the training data consist of input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i$ is an input of arbitrary dimension, y_i is a “labeled” output, and N is the number of training samples, such that $\mathcal{D}$ provides *examples* of values of the function $y = f(\mathbf{x})$. In simple terms, the training data provide sample input data that are *labeled* with the correct or desired output.

It is usually the case that the training data does not exhaustively define the unknown function f. If, however, certain assumptions can be made about the function, then the function might be fully specified by a finite set of training data. In the simplest case, where f is linear and $\mathbf{x}$ is one-dimensional, then only two training samples are needed to specify the relationship between $\mathbf{x}$ and y[4]. Most practical examples of machine learning algorithms, however, are more complicated due to (1) higher dimensionality, (2) non-linearity, and (3) error present in the training data.

Let us consider a simple example from the domain of context awareness. Suppose we would like to develop a smartphone application that needs to know whether the user is walking, running, or standing still (i.e. static). We refer to these as *mobility contexts*. The smartphone has a GPS receiver that can record the user’s position and speed, and it also has a three-axis accelerometer that can measure acceleration. Instead of using the raw accelerometer signal, we define a feature from the accelerometer data, known as *dynamic acceleration*:

$$a_d = var(\{\sqrt{a_{xi}^2 + a_{yi}^2 + a_{zi}^2}\}_{i=1}^{N}) \tag{8}$$

where $var(\cdot)$ is an operator that computes the variance over some time-series of data (e.g. one second of acceleration data); a_{xi}, a_{yi}, a_{zi} are the accelerations in the x, y, and z directions, respectively, for some given time epoch i; and N is the number of samples in the time-series.

A researcher, Mary, has painstakingly collected a dataset for developing this context-aware application and labeled whether she was walking, running, or standing

[4] Recall that two points define a straight line.

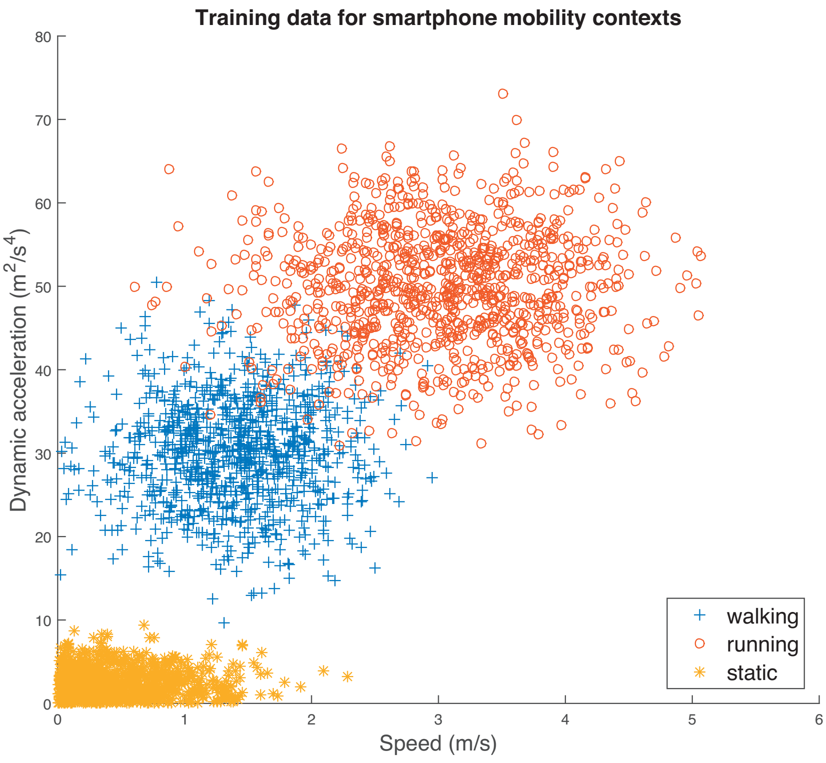


Fig. 3.1: *Example training data for supervised learning. The data are similar to those shown in Table 3.1. Note the partial overlap between the "walking" and "running" classes.*

still. Some sample data are shown in Table 3.1 below, consisting of two dimensions of input data, $(speed^i, a_d^i)$, and the labeled output. In order to keep the size reasonable, only 35 data samples are shown in the table. In Figure 3.1, similar data are plotted, but now we include 1000 samples from each class.

With these data in mind, the goal of supervised learning is to find the function or model f that maps the input data $\mathbf{x}_i = (speed^i, a_d^i)$ to the correct output class $y_i \in \{`walking', `running', `static'\}$, such that the number of errors are minimized. In this context, errors are defined as input data that are mapped to the wrong output class, also known as misclassifications.

Table 3.1: *Example data for supervised learning. The data consist of two-dimensional input data from smartphone sensors and a labeled output class.*

ID	Speed (m/s)	Dyn. accel. (m^2/s^4)	Label
1	2.56	21.10	walking
2	0.94	28.78	walking
3	1.24	31.22	walking
4	2.99	36.66	walking
5	1.24	36.43	walking
6	0.64	29.88	walking
7	0.73	34.13	walking
8	1.68	28.56	walking
9	2.72	32.96	walking
10	1.82	38.57	walking
11	2.10	30.70	walking
12	2.80	49.59	running
13	4.01	47.41	running
14	3.10	61.96	running
15	1.98	54.44	running
16	2.33	53.92	running
17	5.48	44.49	running
18	4.14	52.38	running
19	2.69	52.85	running
20	4.73	44.02	running
21	1.22	48.76	running
22	4.88	47.78	running
23	0.40	2.89	static
24	0.92	0.92	static
25	0.36	1.48	static
26	1.16	3.37	static
27	0.00	5.76	static
28	0.28	3.27	static
29	0.60	0.70	static
30	0.45	2.97	static
31	1.44	1.79	static
32	0.11	1.45	static
33	1.36	1.51	static
34	1.06	0.03	static
35	0.81	1.28	static

Based on the above figure, before employing any machine learning, several observations can be made. We clearly see three clusters of data, corresponding to the three mobility contexts. The cluster corresponding to the "static" context is well separated from the other two, but in the case of the "walking" and "running" contexts, there is some overlap. Another important observation is that in the "walking" data, some of the values for speed are very close to 0 m/s. This could be due to errors in the data (i.e. the data from the GPS receiver might have some error) or labeling errors made by Mary. Similarly, the static data contain many points where the speed is non-zero. It is very common with this type of data that some labeling errors are present in the training set. For example, at the transition points between the walking and static contexts, it is difficult to accurately label which data corresponds to "walking" and which corresponds to "static"[5].

Once the labeled data are collected and the desired input features are generated, the next step is *model selection*, in which the optimal model is determined through quantitative performance measures. Correctly measuring the performance requires dividing the labeled data into three distinct sets: (1) the training set, (2) the validation set, and (3) the test set. To aid in model selection, one often performs precursory *data exploration*, in which different features are plotted to study their distribution and how well the different classes are separated. For example, the observations made in the previous paragraph can be considered a type of data exploration (for another example, see [P4]). For example, if it is clear that the classes are linearly separated, then a simple linear classifier [e.g. Linear Discriminant Analysis (LDA)] may perform well.

Three important interrelated concepts should now be introduced: *generalization*, *underfitting*, and *overfitting*. Generalization refers to the idea that supervised learning should "generalize" beyond the specific examples given in the training data. In other words, the goal is not simply to map the inputs to the outputs for the given training data but rather to find a mapping function or model that works well on some yet unseen data. If the goal were simply to fit a function to the training set, then it would be trivial to write a function that performs with zero errors (e.g. a simple lookup table would do the job).

Overfitting refers to the situation where the supervised learning algorithm has

[5] One technique to avoid such labeling errors is to remove these transition points entirely from the training data.

produced a mapping function that follows the training data in too much detail. Keep in mind that every labeled dataset is somehow incomplete and imperfect. If the training results in a function that does not properly take into account the gaps and the noise in the training data, then it will *overfit* the training data and will not generalize well.

It is also possible that a model *underfits* the training data. This usually means that the mapping function is overly simple, for example, using a linear model for data that are inherently non-linear. Therefore, good generalization lies in between the two extremes of underfitting and overfitting.

The goal of learning is more precisely defined as minimizing the *generalization error*, which is the average error rate that will be produced by any future data, and this means finding a model that neither overfits nor underfits. Of course, it is difficult to estimate the true generalization error. This is the reason for dividing the labeled data into three distinct sets. The test set is not used at all in the learning process but is reserved for estimating the generalization error after learning has already taken place[6].

A test set provides a way to measure the generalization error *after* the learning process and to see whether any overfitting or underfitting is occurring, but the question remains: How does one determine the right type of function or model to fit to the training data, i.e. perform model selection? The answer in short is that we use the *validation set* to measure the relative performance of different models and choose the best one for final testing. In detail, the model selection process proceeds as follows:

1. Choose a hypothesis set $\mathcal{H}$ containing different hypothesis function types to be used in model selection. This hypothesis set can be of one particular function class, such as the set of all linear functions or can be of several different classes. The goal is to include within the hypothesis set a class of functions that match well with the underlying data under investigation. This is, however, non-trivial and may require some precursory *data exploration*.

2. Given the hypothesis set $\mathcal{H}$, for each hypothesis class $\mathcal{H}_i \in \mathcal{H}$, use the training set $\mathcal{D}$ to find the best function $h_i \in \mathcal{H}_i$. For example, if $\mathcal{H}_i$ is the set of all linear functions of the form $h(x) = a * x + b$, then this step is equivalent to

[6] The purpose of the validation set will become clear further below.

finding the parameters a and b that best match the training data, according to some linear regression estimator, e.g. the least squares estimator.

3. Now we have a set of fitted functions, each from a different hypothesis class. That is, for each $\mathcal{H}_i$, we have a corresponding fitted function h_i. Let us denote these as $\mathcal{H}_{best-in-class} = \{h_i\}_1^N$, where N is the number of hypothesis classes. The next step is to choose the best h_i from this set. For this, we use the validation set to measure the error rate and choose the function with the lowest error, which we denote h_{best}, and its hypothesis class is denoted by $\mathcal{H}_{best}$.

4. Finally, fit a new function $h_i \in \mathcal{H}_{best}$ using the training set plus the validation set, and measure its error using the test set. Since the test set was not used in the learning process, the resulting error rate can be considered an estimate of the generalization error.

Depending on the amount of labeled data available, and the complexity of the underlying structure in the data, it may be necessary to repeat this process with different divisions of the labeled data into the respective training set, validation set, and test set. The standard technique for this repetition process is known as *cross-validation*. Due to space limitations, we will not cover cross-validation in detail, but it was employed in [P3] and [P4].

So far we have discussed general concepts in supervised learning but not any specific algorithms. Several examples of supervised learning algorithms will be described in [P2].

3.4 Unsupervised Learning

Unsupervised learning is, in many ways, quite similar to supervised learning, except that there are no labeled data. In other words, there are only input data, and the goal is to learn something about the structure or patterns in the input data. In this way, unsupervised learning is very similar to traditional statistical methods, where the goal is to infer a statistical model from a set of data. Many unsupervised learning methods, such as density estimation, come straight from statistics. Others differ only in the name or some other superficial characteristics. Especially in recent years, there are large overlaps between statistics research and unsupervised learning research[7].

Consider again the data presented in Table 3.1 and Figure 3.1. Suppose Mary had not gone to the trouble of labeling the data with the actual mobility context associated with each data sample. We would have then only a two-dimensional dataset of input data, and we could make a similar plot as Figure 3.1, except the legend would be missing and we would also not have the information necessary to label the samples with different colors as in Figure 3.1. The top part of Figure 3.2 shows such a plot.

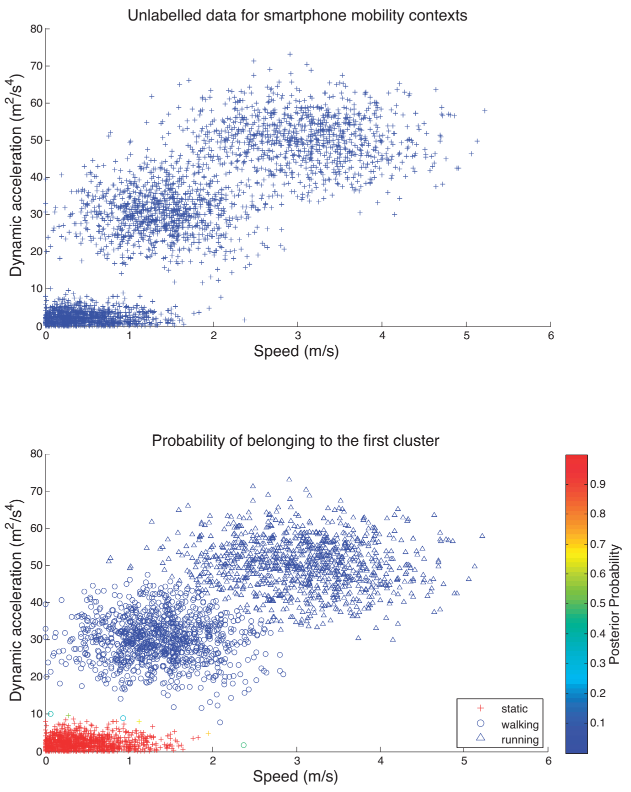


Fig. 3.2: *The top plot shows example input data for unsupervised learning. The bottom plot shows one result from the EM-based clustering. The color of the points shows the posterior probability that the points belong to the first component in a GMM. Data labels are shown strictly for demonstration purposes. In a real situation, no such label would be available to interpret the unsupervised learning result.*

[7] This is also true to a certain extent in supervised learning, but the similarity is more striking in unsupervised learning.

One unsupervised learning task would be to identify different clusters or groups present in the data. Depending on the data and the application, it may or may not be apparent how many clusters are inherently present in the data, so the number of clusters may also be a parameter to determine as part of the unsupervised learning task. There are a plethora of different unsupervised learning algorithms available in the literature that perform clustering. Possibilities include k-means clustering [46], OPTICS [47], and the expectation-maximization (EM) algorithm [48]. In particular, the EM algorithm has its roots in statistics and can fit observed data to an arbitrary statistical model.

To provide an example of clustering, we used the EM algorithm to fit a Guassian mixture model (GMM) to the data that we have previously seen in the top half of Figure 3.2. A GMM is of the form:

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^{K} \pi_k \phi_k(\mathbf{x}; \boldsymbol{\theta}_k) \tag{9}$$

where $\mathbf{x}$ is a random vector, K is the number of components in the mixture model, $\phi_k(\mathbf{x}; \boldsymbol{\theta}_k)$ are normal distributions with parameters $\boldsymbol{\theta}_k = (\boldsymbol{\mu_k}, \mathbf{\Sigma_k})$, π_k are mixing weights satisfying $\pi_1 + ... + \pi_K = 1, \pi_k \geq 0$, and $\Theta = \{\pi_1, ..., \pi_K, \theta_1, ..., \theta_K\}$ is the complete set of model parameters[8].

The EM algorithm itself is a widely-used iterative algorithm used to find the maximum likelihood estimate (MLE) of the model parameters (which we denote with Θ as above) for an underlying distribution $p(\mathbf{x}|\Theta)$ used to model a given dataset, which we denote as $\mathcal{D} = (\mathbf{x}_1, ..., \mathbf{x}_N)$ [49]. The MLE is obtained by maximizing a function Q equal to the expected value of the log-likelihood $\mathcal{L}(\Theta|\mathcal{D}, \mathcal{Y})$, given the observed data $\mathcal{D}$ and the current parameter estimates $\Theta^{(i-1)}$:

$$Q(\Theta, \Theta^{(i-1)}) = E[\log \mathcal{L}(\Theta|\mathcal{D}, \mathcal{Y})|\mathcal{D}, \Theta^{(i-1)})] = E[\log p(\mathcal{D}, \mathcal{Y}|\Theta)|\mathcal{D}, \Theta^{(i-1)})] \tag{10}$$

where $\mathcal{Y} = (y_1, ..., y_N)$ is a vector of latent variables that indicate to which component of the GMM a given data sample $\mathbf{x}_j$ belongs. The latent variables can be expressed in various ways, but perhaps the simplest expression is that $y_j = k$ when $\mathbf{x}_j$ belongs to component k. In the above equation i indexes the current

[8] The notation used for the GMM is similar but not identical to that given in [49].

iteration interval of the algorithm, so $\Theta^{(i-1)}$ represents the parameter estimate from the previous iteration (or the initial estimate, if $i = 1$).

Before applying the EM algorithm to find the parameters Θ of a GMM, one must decide on the number of components K to incorporate into the GMM. As we shall see, each component k in the model will correspond to a cluster in the final clustering result; thus, this step is, in practice, the same as determining the number of clusters, and we can consider K to be a hyperparameter in the estimation problem.

Various methods can be used to determine the best value for K. For low-dimensional data, a practical method is to simply plot the data (as we did in the top half of Figure 3.2) and try to visualize the inherent number of clusters. For high-dimensional data ($D > 3$), this simple approach is not necessarily adequate, nor does it support the goal of automation described earlier. Therefore, a more sophisticated, systematic approach is preferred, such as the one described in [50]. In the interest of space, we assume in this example that the choice of K is already clear, and for these data $K = 3$ seems to be a reasonable choice.

The next step is simply to apply the EM algorithm to determine the parameters Θ of our three-component GMM. A detailed description of the EM algorithm is beyond the scope of this thesis, but here we provide a brief overview.

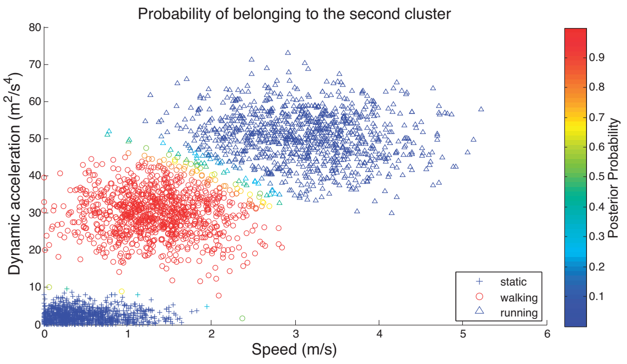
First, EM requires an initial estimate of Θ, and various initialization techniques to provide sensible initial estimates can be found in the literature. A simple approach is to use the given dataset $\mathcal{D}$: e.g. select K random samples to initialize $\boldsymbol{\mu_k}$ and use the covariance matrix of $\mathcal{D}$ for each of the initial K covariance matrices $\boldsymbol{\Sigma_k}$ [51].

After initialization, the algorithm then alternates between computing an expectation function (known as the E-step) and finding the parameters Θ that maximize this function (known as the M-step). At each E-step, the algorithm calculates a new $Q(\Theta, \Theta^{(i-1)})$. In the M-step, an updated estimate $\Theta^{(i)}$ of the parameter set is obtained by maximizing $Q(\Theta, \Theta^{(i-1)})$, according to:

$$\Theta^{(i)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(i-1)}) \tag{11}$$

The algorithm terminates when $Q(\Theta, \Theta^{(i-1)})$, evaluated at $\Theta = \Theta^{(i)}$, converges towards a maximum value (i.e. improvement is below some threshold value ϵ).

Finally, once the parameters Θ are estimated, we can determine the posterior probability that a data sample $\mathbf{x}_j$ belongs to a particular component k of the GMM,



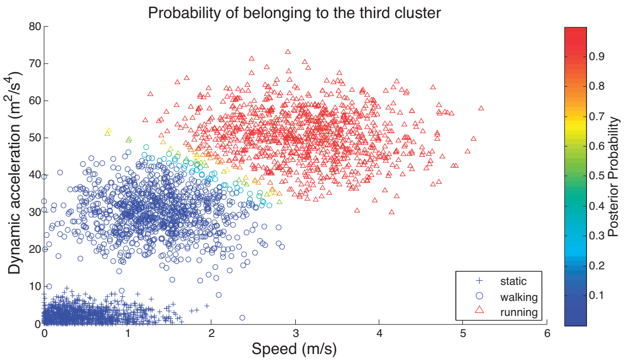


Fig. 3.3: *These two plots continue the results presented in Figure 3.2. The coloring used in the top plot shows the posterior probabilities that the points belong to the second component in a GMM, whereas the bottom plot shows the same for the third component. As in Figure 3.2, the data labels are shown strictly for demonstration purposes.*

according to its so-called "membership weight" [51]:

$$w_j^k = p(y_j = k|\mathbf{x}_j, \Theta) = \frac{p_k(\mathbf{x}_j|\theta_k)\pi_k}{\sum_{m=1}^{K} p_m(\mathbf{x}_j|\theta_m)\pi_m} \tag{12}$$

Recall that each component of the GMM corresponds to a cluster, and therefore the membership weight for a given k is the posterior probability that the data sample belongs to cluster k. The bottom half of Figure 3.2 and Figure 3.3 show the posterior probabilities for our example data, corresponding to membership in each of the three clusters. Note that a dividing line between each cluster can be drawn where the posterior probability reaches 0.5.

Relating this example back to context awareness, we can see from the clustering results that the data can be grouped into three distinct classes, although we have no clear interpretation for these classes in terms of contexts. Nonetheless, we can deduce that two of the classes are somewhat similar relative to the third class (in terms of the two features investigated). Such a result can be useful for context awareness purposes. One purpose is for studying and visualizing the possibility of separating the data into different contexts or situations; if no clusters are evident, it may be very difficult to perform classification with the given features.

Another way to use clustering is as a guide for how to collect training data. A large amount of unlabeled data can be collected and clustered, and then more targeted data collection campaigns can be planned in order to "label" each cluster. Essentially, the clustering results can help one to decide how much labeled data to collect from different segments of the feature space.

Yet another application of clustering is detecting abnormal situations or behaviour. For example, if a particular data sample falls well outside of known clusters of data, then even if the cause of abnormality is unknown, it can be flagged for further investigation. Such techniques are common in applications such as failure detection and security monitoring. Finally, in the context of navigation, clustering and other unsupervised learning techniques can be useful for understanding how different signals might be used for positioning using pattern matching techniques (as discussed in Section 2.2.3).

3.5 Concluding Remarks

Machine learning is clearly a wide topic covering many different concepts and techniques. Our purpose in this chapter was to introduce the most important principles and to elucidate how machine learning can be used to endow computing systems with context awareness. Further examples and details will be given in the included publications. We have emphasized the necessary role of labeled data in supervised learning. We have also demonstrated how a similar result can be achieved through the use of unsupervised learning, although measuring the performance of the result is somewhat problematic.

4. CONTEXT AWARENESS IN NAVIGATION RESEARCH

As stated in the introduction, *context awareness* is the term adopted by mobile computing researchers to describe a computer's ability to understand (i.e. be aware of) the situation or context in which it is operating. In navigation research, of particular emphasis are the *human* context (i.e. the computer *user's* situation) and the *environment* context, but device-specific or vehicle-specific context can also be important to the extent that it can affect the user and his or her goals and ability to achieve them. Examples include: (1) low battery of a mobile device may affect how the user uses the device and even cause him or her to alter plans based on this situation, (2) low fuel level in a car can cause the driver to stop for more fuel, (3) equipment failure in an aircraft can cause the pilot to initiate an emergency landing, etc.

Many definitions of context and context awareness have been proposed, usually reflecting different discipline-specific perspectives. The word context figures prominently in diverse fields including linguistics, psychology, neuroscience, law, and computer science. Due to the great number of definitions, some researchers have used techniques such as latent semantic analysis (LSA) and principal component analysis (PCA) to find the relationships between the many definitions of context [52] [53]. Others (e.g. [54]) have attempted to formalize the concept mathematically.

Let us start by seeing how context is defined in a dictionary. In the Merriam-Webster Dictionary, the word context has two definitions [55]:

1. the parts of a discourse that surround a word or passage and can throw light on its meaning

2. the interrelated conditions in which something exists or occurs : ENVIRONMENT, SETTING

In this thesis, we adopt the second definition. This is because we are not directly concerned with human discourse but rather with conditions of an environment or setting that can be "understood" by computers. Clearly, these two definitions are interrelated—discourse is the way that humans articulate their understanding of an environment or setting. Put in another way, natural language is how humans encode contextual information. In this thesis, we focus on techniques that computers can use to sense, represent, and process context without human intervention. When we refer to context, we refer directly to the conditions in the environment/setting rather than representations of context, such as discourse. As noted in Section 1.2, *situation* can be used as a synonym for context. We see no reason to distinguish between the two terms, although we note that some formalisms make a distinction (see [56]).

With this working definition of context established, we proceed to the remainder of the chapter, which is organized as follows. First, in Section 4.1 we provide two simple frameworks for specifying a context (i.e. the "interrelated conditions") and for contextual reasoning. Next, Section 4.2 provides a review of relevant context awareness literature, paying particular attention to studies relevant to the three tasks described earlier in Section 1.2. Section 4.3 analyzes the differences between our proposed context frameworks and one popular representation of context found in the literature. Finally, Section 4.4 describes the processing chain used to build up context-aware navigation services, beginning with sensing, proceeding up to context recognition and higher-level reasoning capabilities, and finally integrating these capabilities into services.

4.1 Frameworks for Context and Contextual Reasoning

In this section we describe two separate but related frameworks for working with context and contextual reasoning. These frameworks are also covered later in [P1] and [P2], so here we provide only a brief introduction, in order to summarize the ideas and emphasize the author's contributions within the context of the whole thesis.

4.1.1 A Framework for Contextual Information

Because context is such an abstract concept, it is useful to choose some techniques for describing a particular context. These techniques can be used to build a framework for expressing contextual information. The goal of this section is to describe one such technique. We make no claim that this technique or framework is an authoritative one, nor that it is complete in the sense of exhaustively covering the concept of context.

In our view, the goal of context-aware systems is essentially to mimic the way that humans understand and describe situations, contexts, conditions, or events (we use all these terms almost interchangeably, although they may emphasize different aspects, such as fixed versus dynamic elements). According to this goal, we might employ the classic technique of journalism (since journalism is an age-old craft for describing conditions and events), known as the Five Ws: Who, What, Where, When, and Why [57]. This technique can be traced back to the late 2nd century BC when Hermagoras of Temnos defined seven elements of circumstance, which includes (in addition to the Five Ws) “in what manner” and “by what means” [58].

Using these questions as a framework (with a slightly different order), the following provides an example of elements of a particular context:

What: A small gathering of colleagues for lunch

Who: Present are Mary, Philip, George, and Anita

Where: 60.1609°N, 24.5460°E (WGS84); inside the lunch-room of the Finnish Geospatial Research Institute (FGI) in Masala, Finland

When: Tuesday, 10 March 2015 at 11:03AM

Why: Because it is lunchtime, and it is the custom for this group of colleagues to eat lunch together.

In What Manner: Mary’s smartphone is experiencing small, sporadic movements, but it mostly remains in a constant orientation. Mary’s smartwatch is experiencing more dramatic but also sporadic movements. Both sources of motion data are consistent with a user who is sitting and having a casual conversation and/or eating lunch. Multiple human voices are engaged in conversation of an informal and lively manner.

By What Means: All of the above information has been sensed or reasoned by the sensors and software existing in a smartphone and a smartwatch, plus some additional sensor data recorded by a networked node installed in the lunch-room. In this case, the smartphone is a Samsung Galaxy S5 with Android 4.4.2 Operating System (OS), which includes a GPS receiver, WLAN-based positioning engine, Bluetooth connectivity, microphone

> and audio analyzer, ambient light sensor, accelerometers, gyroscopes, compass, and magnetometers. The smartwatch is an LG G Watch with accelerometers, gyroscopes, Bluetooth connectivity, microphone, and an audio analyzer. It runs Android Wear 5.0.1.

This depiction of the situation is not likely to win a Pulitzer Prize in Journalism, probably because the situation is not particularly interesting. Also, note that it has not been formulated completely into prose but rather is more like a set of notes that a journalist might jot down for later use (except maybe for the latitude and longitude coordinates and the motion description). The "By What Means" section can also be thought of as notes as to the "source" that the journalist might record along with the other information (especially if the account is second-hand).

The seven elements of circumstance can be interpreted as different elements of context, according to the following guidelines:

- "What" usually refers to the *activity context*, that is, what is actually happening. In some contexts, there might be little "action" taking place, but it may also be of interest for some purposes that "nothing is happening".
- "Who" refers to the human characters in the context. When speaking about context-aware mobile devices, the user of the mobile device in question is usually the main character, whereas others in the environment can be thought of as supporting characters. The "who" portion can also be summarized as the *user and social context*.
- "Where" refers to the *location context*. The most important point to note is that location can be expressed in many different ways: geographic coordinates, an address, or some semantic representation such as "the Finnish Geospatial Research Institute" or perhaps more personalized, such as "my workplace". "Where" can also refer to the *environment context*, which includes information about the location where the action takes place. For example, does the action take place indoors or outdoors? Is the location a home, business, etc.?
- "When" is the *time* and *date context*. We need only to be careful about specifying things like time zones. In addition, it may be important to encode some common sense or semantic knowledge about meaningful aspects, such

as "this is after work hours" or "today is a holiday". Time can be specified either as a specific moment, such as in the example above, or as a time or date segment (e.g. 10–20 March 2015).

- "Why" can be thought of as the *motivational context*, e.g. Why is the user doing that? Why is this event taking place? Etc. It could also be appropriate to encode information about whether the context is normal or unusual, as well as an explanation for the unusual events. For example, if a person normally commutes to work along $routeA$, and in the present he or she is driving along $routeB$, then "why" would be a good place to capture the fact that "there was an accident along $routeA$, so the alternate $routeB$ was chosen".

- "In What Manner" is a bit of a "catch all" category. It is used to provide additional details that do not fit nicely into any of the other categories. This is less than ideal for any formal system of context, but rather than attempting to list out all the possible categories of context (which is probably impossible), we believe it is more practical to have an "other" category. One way that we use this category is to capture the *motion context*. This is similar to the activity context, but it is more focused on detailed attributes of the motion. For example, if the current activity of a user is "dancing", then "in what manner" might be used to capture the type of dance and the tempo in which the user is dancing.

- "By What Means", as mentioned above, is to capture the source of the contextual information. It includes information about the devices and sensors used in the context-aware system, as well as the reasoning methods employed.

Further details about this framework are covered in [P1]. As context-aware systems develop, we suspect that our framework may change slightly. It is difficult to anticipate what types of contextual information will become important in the future, but this framework should be broad enough to encompass most types of contextual information. Also, our intention is not to have a rigid framework that constrains all future context-aware systems, but rather it is to provide a rough skeleton upon which to experiment and build more elaborate and detailed context ontologies.

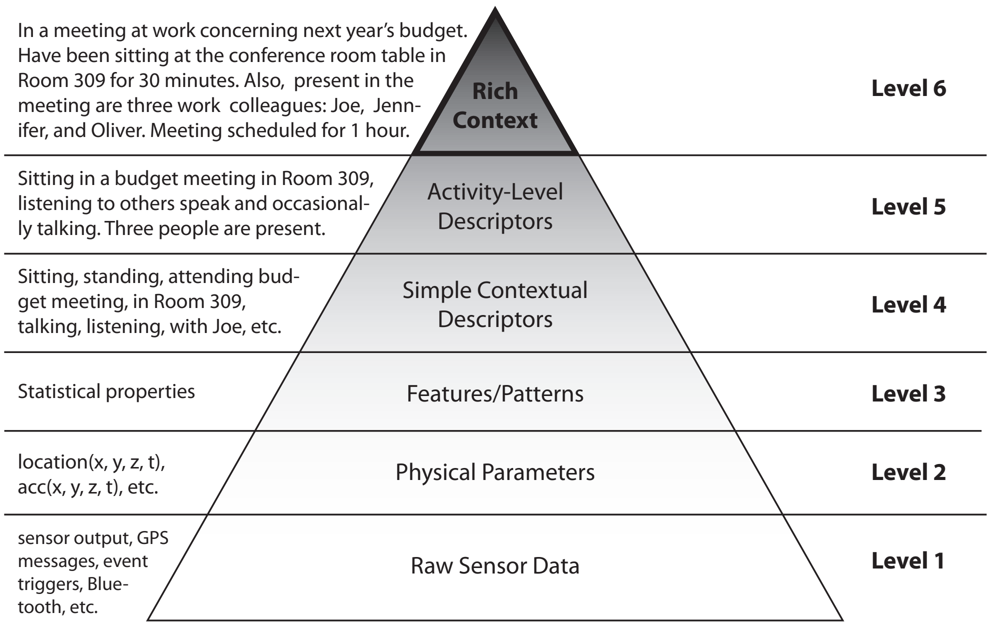

In a meeting at work concerning next year's budget. Have been sitting at the conference room table in Room 309 for 30 minutes. Also, present in the meeting are three work colleagues: Joe, Jennifer, and Oliver. Meeting scheduled for 1 hour.
Rich Context
Level 6
Sitting in a budget meeting in Room 309, listening to others speak and occasionally talking. Three people are present.
Activity-Level Descriptors
Level 5
Sitting, standing, attending budget meeting, in Room 309, talking, listening, with Joe, etc.
Simple Contextual Descriptors
Level 4
Statistical properties
Features/Patterns
Level 3
location(x, y, z, t), acc(x, y, z, t), etc.
Physical Parameters
Level 2
sensor output, GPS messages, event triggers, Bluetooth, etc.
Raw Sensor Data
Level 1

Fig. 4.1: *The "context pyramid" shows the different levels of processing to build context-aware systems, starting from raw sensor data at the bottom and working up to "rich context" at the highest level.*

4.1.2 A Framework for Contextual Reasoning

In this section, we present a framework for *contextual reasoning*, which we define as the process of forming higher level inferences about context from lower-level information. This topic and the associated framework are covered in greater detail in [P2] and partly also in [P3]. We conceptualize the process of contextual reasoning as a pyramid, dubbed the "context pyramid". This context pyramid is shown in Figure 4.1 below. On the left side is an example of contextual information. Inside the pyramid and forming separate layers are different types of data or information related to context. Between each level a step in the contextual reasoning processing chain is inherent. We have not given specific labels to the processing steps, due to the difficulty in generalizing about the processing steps, but we will attempt to describe some common processing steps below.

At the bottom of the pyramid lies the raw input to the context-sensing system, such as sensor data. The difference between the first and second level in the pyramid is that in Level 2, some "pre-processing" of the data may have been performed, such as reference frame transformation or filtering out noise. In the next level of processing, statistical features are extracted from the data, such as mean values or frequency domain features computed from time-series data. The distinction between "pre-processing" and statistical feature extraction can be a bit blurry in some cases, but generally-speaking Level 2 data usually have a clear physical meaning, whereas Level 3 data might have only a mathematical or statistical meaning.

Next, Level 4 is achieved after the Level 3 data are subjected to a function or algorithm that performs contextual classification or in some cases regression. We have proposed that this function or model can be built using machine learning techniques, as described in Chapter 3. This step in the process will be covered further in Section 4.4.2. Level 4 data are in the form of *simple contextual descriptors*, which can be thought of as "atomic" elements of context. They each should belong to one of the seven categories described in Section 4.1.1 above.

Then, Level 5 is achieved by combining multiple simple contextual descriptors into an activity-level description of the context, including the main pertinent contextual details. This process is described in more detail in Section 4.4.3 and again machine learning is the primary tool used to achieve this level in the context pyramid.

Finally, Level 6 combines all available contextual information into a *rich context*. The aim at this level is to approach a description of the context that is indistinguishable from human-written prose. As was the case between Levels 2 and 3, the difference between Levels 5 and 6 can be sometimes blurry. In some context-aware systems there may be fewer processing steps or perhaps more, so even the number of levels should not be taken as dogma. We believe, however, that the general process will always follow the overall trend illustrated in the context pyramid.

As a further example, take the case of an ice-aware maritime route optimization system (i.e. Task 3), which is very different from the example contextual information given in Figure 4.1. At the bottom of the pyramid would lie raw data concerning the ice field, e.g. data from Synthetic Aperture Radar (SAR) or other sources. The first step is to extract physical parameters from the raw data, such as a grid of values for ice thickness and ice concentration, or other system parameters such as the location of an ice breaker. Next, one might perform statistical analysis on the ice data, e.g. applying Gaussian process regression for the purposes of interpolation. Then, one uses features from Level 3 and a ship performance model, in order to estimate the ship's theoretical speed at coordinate $(lat, long)$. Machine learning is one of the possible tools for building the ship performance model.

This type of contextual information is analogous to a Level 4 simple contextual descriptor (i.e. the *in what manner* context). Next, the planned route, computed using the Level 4 contextual information and the route optimization algorithm, is roughly analogous to an Activity-Level Descriptor at Level 5 because it is a higher-level context inferred from lower-level contextual information. Finally, combining the optimal route with other navigational information, such as weather information, maritime traffic information, etc., produces a rich contextual description describing in detail the ship's current situation, i.e. Level 6 at the top of the pyramid.

4.2 Related Studies

There is a vast corpus of context awareness literature. A comprehensive review of recent context awareness literature would require covering hundreds, if not thousands, of different studies. The review found in [59], which only looks at journal articles published between 2000 and 2007, covered well over 200 different studies. Searches of several major databases of scientific publications (e.g. IEEE Xplore Digital Library, ACM Digital Library, SpringerLink) using the keyword "context-aware" each yielded thousands of results. Clearly, these publications will vary greatly in their relevance to this thesis. To help focus on only the most relevant literature, we only include publications related to the three context awareness tasks described in Section 1.4. We also provide a brief overview of the early context awareness research.

According to our review of the literature, the first explicit reference to context awareness was in a 1994 paper by Schilit and Theimer, where they use the term *context-aware computing* to describe software that can "adapt according to its location of use, the collection of nearby people and objects, as well as the changes to those objects over time." [60]. Earlier, however, we can find strong but implicit references to the concept of context awareness. For example, in the 1991 article, titled "The Computer for the 21st Century," Weiser provides a fictional account of a number of different automated or computer-assisted functions made possible by "ubiquitous computing" [61]. Although not specifically highlighted by Weiser, the necessity of these computers to understand context is clearly evident.

Another article, published in 1992 by Want et al., may be the first implementation of a context-aware device described in the literature, even though the authors did not use the term “context-aware” [62]. By the mid-1990s, many different implementations of context-aware devices can be found, including the ParcTab, stick-e notes, CyberGuide, and CyberDesk. By 2001, the research field was active enough to support a special issue of the journal Human-Computer Interaction, which provides an excellent review of the state-of-the-art in context awareness for that time period [63].

Going further back, however, the concept of context has been studied in computer science research for many years. As early as 1963, John McCarthy, one of the “fathers of AI”, began developing *situation calculus* as a “formal system in which facts about situations, goals and actions can be expressed” [64]. A situation is defined as “the complete state of affairs at some instant of time”, thus, it is roughly equivalent to our definition of context. Beginning in 1987, McCarthy began to consider the concept of context explicitly and attempted to formalize it [65].

Formalisms of context, however, do not appear to have led directly to the realization of any context-aware software or devices, except for perhaps one example, Cyc [66]. The context-aware devices and applications of the 1990s mostly consisted of location-aware devices, and in our opinion, they do not require an elaborate formalism of context. Nonetheless, the work of McCarthy and others pioneers who offered formalisms of context are worthy of mention in the history of context awareness. In particular, we refer the interested reader to [54] [67] [68] [56], and [69]. In addition, [70] and [71] provide excellent reviews of context in artificial intelligence.

* * *

We now review literature specifically related to the three tasks described in Section 1.4.

In terms of the first task, recognizing the activity of a smartphone user in an indoor office environment, there are only a few studies basing their results on smartphone sensors, and they mainly rely on location awareness. For example, one early study described user tests of a context-aware Personal Digital Assistant (PDA) application, which can be considered a proto-smartphone-like mobile device, designed to be used in an office environment [72]. The main source of contextual information was location, but the application also used contextual information from calendar events that the user had previously recorded in the application. For example, if a user scheduled an event at a particular time and marked it as a meeting, then the system would automatically suggest meeting-related services when the meeting time approached and the user had entered the meeting room. The described system, however, was not capable of automatically inferring office activities independently from user-supplied information.

More recently, [73] described a pilot study related to smartphone-based context awareness of "mobile knowledge work", which may partly take place in office environments. The article, however, mainly discusses the research problem in general, as well as describing the pilot data collection campaign. The article does not go into any detail on the results of analyzing the collected data, nor the planned analysis methods.

If we include studies using non-smartphone sensors to detect office-related activities, we encounter several other relevant studies. For example, [74] used a microphone and a USB camera to detect different office activities and motion patterns, including "rest," "moving near door," "conversation," "nobody around," etc. They used two machine learning techniques to detect motion and activity: Incremental Hierarchical Discriminant Regression (IHDR) trees and Hidden Markov Models (HMM). [18] similarly used a set of cameras to detect whether an office worker is working alone or in a meeting, or whether the office is empty. For activity detection, they used adaptive background modeling, which makes use of Gaussian mixture densities [75].

Another example is [76], which used a richer set of sensors, including two USB cameras, a pressure sensor, and an acceleration sensor (a Wii remote controller) built into an office chair. This study focused on detecting location and posture context, such as leaning back in the chair, leaning on the desk, or upright sitting posture. They primarily used k-means and kNN clustering to perform context recognition. Similarly, [77] presented a so-called "smart office chair" designed to measure an office worker's mental and physiological states, such as sleepiness and concentration. The details of the recognition algorithm, however, are not given in the paper.

One study used a combination of mobile phone sensors and room occupancy sensors installed in an office environment [78]. The resulting application, called *WorkSense*, was able to detect when and where meetings and conversations took place. By sensing social interactions, the authors were also able to identify project groups automatically.

Other relevant studies have focused on using wearable sensors to detect different activities, some of which are common in office environments. For example, [79] used accelerometers attached to various parts of a person's body. Although they did not specifically focus on an office environment, some of the activities they addressed are very relevant. These include sitting, standing, walking, writing on a whiteboard, typing on a keyboard, and shaking hands. The authors used a naïve Bayes classifier for activity recognition. Similarly, [80] used wearable sensors to detect several different office-related activities (typing, cleaning a whiteboard, using an elevator, etc.). The authors evaluated the use of kNN and Multilayer Perceptron (MLP) classifiers for activity recognition[1].

Finally, [81] studied context recognition in a meeting room environment, using a suite of sensors including: a microphone array, passive infrared sensors, and an illumination (light) sensor. They detected different states of activity in the meeting room, such as: presentation, monologue, discussion, room idle, people entering room, and room in use. They do not describe their context recognition algorithm in any great detail, but it appears to be a heuristically-defined rule-based algorithm.

The relatively small number of studies related to context awareness in an office environment suggests this is a fairly immature research topic. We can conclude from our literature review that the research in this thesis is rather novel, especially with regards to smartphone-based context awareness for office environments.

* * *

With regards to the second task, recognizing modes of motion that a smartphone user is undergoing outdoors, there are comparatively many relevant studies. Despite this fact, to our knowledge no systematic review of this research area has been published, but a few related reviews are available. For example, [82] reviews literature on the use of GPS to study health-related physical activity. The focus of this review, however, is mostly on assessing health behavior, rather than on methods to detect different mobility contexts. [83] reviews smartphone-based "opportunistic user

[1] MLP is a type of Artificial Neural Network (ANN).

context recognition”, which includes some literature on mobility context recognition, and similarly [84] reviews “anticipatory mobile computing”. While useful in our review of the literature, none of these reviews focus specifically on mobility context awareness, and they exhibit significant gaps in this regard.

Due to the large number of relevant studies, we will not describe them individually, but pertinent facts from these studies can be found in Table 2.1. We make no claim that this compilation of publications on the subject is exhaustive, but it is, according to our knowledge, the most comprehensive compared to existing literature. We decided to include also studies that utilized wearable sensor modules, since this research is closely linked to smartphone-based research. All of the listed studies used supervised machine learning techniques, except for [85], which evaluated several different unsupervised learning techniques.

Note also that many of the studies investigate not solely mobility contexts but also other contexts, such as Activities of Daily Life (ADL) or various posture contexts (sitting, lying down, etc.). In Table 2.1, we have only identified the relevant motion-related modes. Some other motion-related contexts, such as using an elevator or walking upstairs/downstairs, were investigated by a few studies, but we listed only those modes common to a significant number of studies. For those studies incorporating additional contexts, we have identified these contexts collectively using the symbol “+”. Lastly, we point out that some of the included studies investigated indoor motion modes, even though our research task was to study outdoor mobility contexts.

Table 4.1: *Publications related to mobility context. The definitions for the abbreviations used for algorithms/techniques are given in the Abbreviations section provided above. The abbreviations used for the motion modes are as follows: S = static (including standing, sitting, etc.), W = walking, R = running/jogging, B = riding a bicycle, D = driving a motor vehicle, MT = any kind of motorized transport (including car, bus, train, etc.), RB = riding a bus, RT = riding a train/tram/light-rail, and RS = riding a subway/metro. "+" indicates that other motion modes are also covered by the publication.*

Year	Author(s) & Citation	Device(s) Used	Algorithm(s) / Technique(s) Evaluated	Motion Modes Studied
2000	Foerster, Smeja & Fahrenberg [86]	accelerometers	rule-based	S, W, B, +
2002	Lee & Mase [87]	sensor module	fuzzy-rule-based	S, W, +
2005	Lester et al. [88]	sensor module	AdaBoost, NB	S, W, R, B, +
2005	Ravi et al. [15]	sensor module	DT, kNN, SVM, NB, meta-classifiers	S, W, R
2006	Pärkkä et al. [89]	various sensors	rule-based, DT, ANN	S, W, R, B, RB, +
2006	Pirttikangas, Fujinami, & Nakajima [80]	sensor modules	ANN, kNN	S, W, R, B, +
2007	Suutala, Pirttikangas & Röning [90]	sensor modules	SVM, HMM, SVM-HMM, DTS	S, W, R, B, +
2008	Kunze & Lukowicz [91]	sensor modules	DT, kNN, BN	S, W, R, B, +
2008	Jin et al. [92]	sensor modules	fuzzy-rule-based	S, W, R, +
2009	Yang [93]	mobile phone	DT, NB, kNN, SVM, HMM	S, W, R, B, D, +
2010	Reddy et al. [94]	mobile phone	DT, kMC, NB, ANN, SVM, CHMM, DT+DHMM	S, W, R, B, MT,
2010	Pei et al. [95]	mobile phone	rule-based	S, W, +

Table 4.1 – continued from previous page

Year	Author(s) & Citation	Device(s) Used	Algorithm(s) / Technique(s) Evaluated	Motion Modes Studied
2010	Frank et al. [96]	sensor module	NB, BN, HMM	S, W, R, +
2011	Stenneth et al. [13]	mobile phone	DT, NB, BN, RF, ANN	S, W, B, D, RB, RT
2011	Pei et al. [97]	mobile phone	DT, BN, SVM	S, W, +
2011	Susi, Borio, & Lachapelle [98]	sensor module	DT, NB, kNN	S, W, R, +
2012	Bancroft et al. [99]	sensor modules	NB, rule-based	S, W, R, B, MT, +
2012	Anguita et al. [100]	mobile phone	SVM, HF-SVM	S, W, +
2013	Guinness [24]	mobile phone	20 different algorithms	S, W, R, D, RB, RT, +
2013	Susi, Renaudin, & Lachapelle [101]	sensor modules	DT	S, W, +
2013	Feng & Timmermans [102]	sensor module	BN	W, R, B, D, RB, RT
2013	Hemminki, Nurmi & Tarkoma [103]	mobile phone	HMM, Adaboost, meta classifier	S, W, RB, RT, RS
2014	Stenneth [104]	mobile phone	NB, BN, DT, RF, ANN, meta-classifiers	S, W, B, D, RB, RT
2014	Xia et al. [105]	mobile phone	SVM	S, W, B, D
2014	Elhoushi et al. [106]	mobile phone	DT	W, R, B, MT
2014	Parvainen et al. [107]	mobile phone	DT, SVM, MAP	S, W, R, B, MT
2014	Yu et al. [14]	sensor module + mobile phone	DT, AdaBoost, SVM	S, W, R, B, MT
2014	Kwon, Kang, & Bae [85]	mobile phone	GMM, kMC, HIER, DBSCAN	S, W, R, +

Table 4.1 – continued from previous page

Year	Author(s) & Citation	Device(s) Used	Algorithm(s) / Technique(s) Evaluated	Motion Modes Studied
2014	Sankaran et al. [16]	mobile phone	rule-based, GPSAR, FMS	S, W, MT
2014	Chiang, Yang & Tu [108]	mobile phone	DT, kNN, NB, SVM	S, W, R, B, D, +
2015	Yu & Cho [109]	mobile phone	DT, SVM, ANN	S, W, R, D, RB, RT, RS

* * *

Regarding the third task, determining the optimal path of a ship traveling through ice-covered waters, only a few highly-relevant studies are available in the literature. The earliest is [27]. As discussed briefly in Section 1.5, this work expressed the route optimization problem as a differential equation and used numerical methods to solve it, namely Powell's method. As a result, their system cannot guarantee that the computed route corresponds to a global optimum. Similarly, [110] uses a genetic algorithm for route optimization in ice-covered waters. Genetic algorithms have better capabilities to escape from local minima, but they still do not guarantee a global optimum.

The first study to adopt a graph-based approach is [25][2]. The authors present a method for ice-aware route optimization that uses Dijkstra's algorithm to find the optimal route. In the examples given in the paper, only a few tens of nodes were shown for the sea area under consideration. For such small graphs, Dijkstra's algorithm is tractable, but for larger graphs with many edges, Dijkstra's algorithm does not scale well. To overcome this challenge, [26] uses the A* algorithm, which uses a heuristic to guide the search process. Our method, published prior to [26], also uses the A* algorithm. One difference between [P5] and [26] is that the cost

[2] As reported in [110], [111] also used a graph-based approach, but it is only available in Korean. [25] appears to be an extension to [111].

function we developed takes into account possible ice breaker assistance. [26] does not consider this case.

Another related study is [112]. While it does not consider route optimization explicitly, the author investigates ship performance in varying ship conditions. The results are thus applicable to route optimization. Other examples of research in this area include [113], [114], and [115].

Lastly, the project Ice Forecast and Route Optimization (IRO-2), discussed in [116], is very relevant to this topic, but scientific results are not yet available publicly. A preliminary publication reports on tests of the project's prototype ice-aware ship routing system, but the authors only describe performance of the ice forecast model, not the routing system itself [117].

4.3 Analysis of Proposed Frameworks for Navigation Research

In this section, we analyze the proposed frameworks described in Section 4.1 in comparison to an existing definition of context found in the literature and describe why the proposed frameworks are particularly well-suited for navigation research. One of the most similar categorical frameworks of contextual information, compared to ours, can be found in [118]:

> Context-aware applications look at the who's, where's, when's and what's (that is, what the user is doing) of entities and use this information to determine why the situation is occurring.

This framework for context overlaps ours in terms of the first four questions (what, who, where, and when), however, it does not treat *why* as a "first-class citizen" of context but rather as an output of a context-aware application. Our view is that *why*, i.e. the motivational context, is an important aspect of context. It is true, however, that it is often more challenging to determine the motivational context compared to other aspects of context, but *why* can also be an input to higher-level context reasoning processes, so we see no need to treat it differently compared to other aspects of context. Earlier we gave an example of the why context from navigation: that "there was an accident along $routeA$, so the alternate $routeB$ was chosen".

Another example could be taken from an autonomous driving application, where a car owner might in some cases like to drive above the speed limit. Under normal circumstances, perhaps the autonomous driving system would not allow this, but, for example, in an emergency this limitation might be removed. The *why* context would be the natural category in which to encode this kind of information regarding the reasons for operating in a particular mode.

In addition to this difference, the framework of [118] lacks two other categories from our proposed framework: "In What Manner" and "By What Means." Earlier we described "in what manner" as a kind of "catch all" category. The justification for having such a category is that there often arise cases where a piece of contextual information does not fit nicely into any of the other categories, but these examples are scattered such that it would not be desirable to have many distinct categories to cover each of these cases. An example from navigation that comes to mind is:

What: Aircraft in flight between JFK and LAX
Where: 38.6272°N, 90.1978°W, FL310 (i.e. about 31,000 feet altitude)
In What Manner: Flying at 805 km/h and experiencing heavy turbulence

One could argue that particular context categories could be added for specific applications, such as aviation, but our goal in enunciating this framework is to produce a general, flexible framework that can be used consistently in many, if not all, applications.

Finally, the "By What Means" category captures the sources of the contextual information. In this way, it somewhat of a "meta-context" category, since it contains information about the context information and not about the context itself. In a certain sense, however, the sources of contextual information are important aspects of context as well. For example, we propose to include in this category information about the devices and sensors used in the context-aware system, which we argue are an important part of context. This is certainly the case in navigation research, but we believe it would also be the case in other areas of context awareness research. It is not clear in which category this type of information would be included in the definition of [118].

4.4 How to Sense and Use Context for Navigation Research

In Section 4.1 we mainly discussed context and context awareness at a general theoretical level. In this section, we will describe how we approached the problem of sensing and using context in navigation research at a more detailed, practical level. The sub-section structure follows roughly the levels in the previously described "context pyramid", although we have combined some levels in the interest of space. In the last sub-section, we further enforce the motivation for introducing context awareness to navigation research.

4.4.1 Sensing for Context Awareness in Navigation

As stated in the introduction to this chapter, this thesis focuses on techniques that computers can use to sense, represent, and process context without human intervention. Thus, the goal is to "sense" context using automated means. This means utilizing various sensors that can measure the environment and other aspects about the situation in which a device and the user of the device find themselves.

When applying these techniques to navigation, there are a few obvious choices of sensors that can help determine the context. For example, sensors that can determine the position and velocity of the user or vehicle in question are natural choices (e.g. GNSS receivers). Other sensor choices, however, may be less obvious. For example, we stated in the introduction that a navigation system can perform better by recognizing the mode of motion a user or vehicle is undergoing. It may have one mode with a navigation algorithm optimized for when a user is driving and another mode optimized for when the user is walking. The choice of which sensors to use to recognize these different motion modes is non-trivial. In this thesis, we have adopted a practical approach of choosing the sensors that are already available in smartphones (i.e. [P3] and [P4]). In general, however, the choice requires a careful understanding and consideration of the requirements of the navigation application, and the researcher must investigate the unique attributes of each context that must be recognized. A period of experimentation with different possible sensors may be required.

Also, specific sensors often have different options in terms of sampling rate, sensitivity, resolution, or other parameters. Each of these parameters may have an impact on the final context-aware system. In resource-constrained applications, such as in mobile devices, one must not only consider the context recognition performance of the system but also other factors, such as power consumption, memory requirements, etc. In this thesis (namely [P3] and [P4]), we were mainly constrained by the rate at which sensor data could be recorded on the smartphones used in the research. When we attempted to record data from all of the available sensors at the highest possible data rates, the data collection software would often become unresponsive and occasionally also crash. Thus, through experimentation we obtained a balance between data recording rates and stability of the software.

In addition to the choice of sensors and sensor parameters, one must determine what types of features to generate or "extract" from the raw sensor data (recall Level 2 and Level 3 of the context pyramid). In this regard, existing literature dealing with similar context awareness areas is one of the best guides, but of course we aim also to find novel features not found in the state-of-the-art. In this thesis, for example, we use features such as the distance of the user to the nearest train station (computed using the user's current location and the locations of train stations available in a GIS database), capitalizing on the fact that train journeys virtually always start and end at train stations.

Discovering novel features to use in context awareness requires ingenuity on the part of the researcher and also experimentation. It is often not possible to determine the most useful features for context awareness *a priori*. Only after a process of feature selection can the value of different features be evaluated quantitatively (see [P4]). For example, some features that we envisioned would be useful for distinguishing between motion modes turned out to improve the performance negligibly.

4.4.2 Motion, Environment, and Activity Recognition

In navigation applications, the motion ("in what manner"), environment ("where"), and activity ("what") contexts are three particularly important aspects of context awareness. In most cases, these cannot be determined directly from sensors. Instead some type of model must be adopted in order to interpret the sensor data and arrive at the desired contextual result. Continuing the example from the section above, a navigation system that has different modes for driving, walking, etc. must adopt a model for each of these motion modes, where sensor data (or features from the sensor data) are the inputs to the model, and the output is the most probable motion mode. Determining the best model to use for recognizing different contexts is, in our view, one of the most challenging aspects of context awareness research. We have investigated the use of machine learning techniques for building and optimizing models of this type.

Recalling Chapter 3, the process of model selection must be performed to find the best model for inferring context from sensor data. This process was employed in [P3] and especially in [P4]. Different types of models perform differently in different domains. The "no free lunch theorem," discussed in [P2] states that no single machine learning algorithm performs better than any other across all problems. Thus, model selection is unavoidable if one wants to optimize the performance of the model.

We again emphasize that thorough data exploration can help guide the selection of model hypotheses and also to a certain extent in feature extraction and selection. By plotting different pairs of features, one can visualize how the context classes are separated. When the feature dimensionality is high, however, it is difficult to visualize class separation in this way and techniques from unsupervised learning may be more effective. For example, using algorithms like DBSCAN or OPTICS, one can generate two-dimensional plots representing clusters of data of arbitrary dimension.

Another issue that must be considered at this level in the processing chain is how to divide up the context space into distinct classes. We will return to this subject in Section 6.4.3, but for example, in the case of mobility context, there are many possible sets of mobility modes that can be used to divide up this aspect of context (e.g. motorized transport can be grouped together as one context, or different forms of motorized transport can be treated as separate mobility contexts). We argue in Section 6.4.3 that some level of standardization in this regard is sorely needed in context awareness research. The correct choice, however, really depends on the application, i.e. how the context results will be used. Clustering or other unsupervised learning techniques may also help guide the choice because they can reveal how feasible it is to separate different class sets with the available data.

4.4.3 Higher-level Contextual Reasoning

In addition to the type of modeling approach described above, it may be the case that higher-levels of contextual reasoning are needed, where the inputs to the contextual reasoning model consist of outputs from a lower-level model. For example, in [P3] we used the output of motion mode classification, together with location, in order to reason about higher-level activity contexts from a workplace environment, such as “fetching coffee,” “having lunch,” “working,” etc. In some context awareness research, activity recognition is not performed in hierarchical fashion, but our research has shown this to be an effective strategy. In [P3] it not only produced promising classification performance, but this hierarchical approach has the added benefit that the results at the separate layers are more easily interpreted and can be used independently for different purposes.

The methodology used for higher-level contextual reasoning is much the same as presented above, but the major difference is that the labeled data consist of outputs from a lower-level process of context inference together with the desired final output. It will often be the case that some of the input data to this higher-level process contain errors propogating from the lower-level processes, but these errors can be treated in the same way that noise from sensor data is treated. Given enough training data, machine learning algorithms are generally able to minimize the effects of such errors.

4.4.4 Using Context in Navigation Services

It is important to note that context awareness is not an end goal in itself, but rather we would like to use context awareness to improve various navigation services. In order to demonstrate the usefulness of context awareness in a navigation service, [P5] integrates context awareness into a maritime route optimization system. The specific aspects of context that are utilized in [P5] are the environment context, in terms of information about the sea and ice environment, the location context, the speed of the ship, and contextual information about the icebreakers (i.e. icebreaker waypoint locations). The resulting route optimization system is said to be “ice aware,” since it especially takes into consideration the current ice conditions and icebreaker information.

The main benefit of introducing context awareness in this particular case is that route planning can be performed in a more automated manner. Currently, maritime routes are mostly planned manually by experienced navigators. Especially in ice conditons, the navigator must integrate many different sources of information in various formats. This is a difficult task for a human to perform and one that is particularly well-suited for computers. The main challenge is incorporating into the route optimization system many different operational considerations that affect the route planning functions, such as safety, economic factors, or maritime traffic conditions. By virtue of the general nature of our context framework, all of these aspects can be considered as part of context in one way or another.

The results described in this thesis are merely a starting point in integrating context awareness into an ice-aware route optimization system, and much further work is needed before the system can be implemented operationally or commercially. Future work in this area is discussed briefly in Section 6.4.1.

The details of how to integrate context into navigation services will vary greatly depending on the requirements of the service, but in navigation there are several aspects of context that are particularly important. A few, such as motion, environment, and activity, have been mentioned earlier, and this thesis emphasizes those aspects. Other obvious examples include location, speed, and conditions of the vehicle (if applicable). In road transportation, traffic conditions are increasingly being incorporated into navigation services. Traffic can be considered as part of the environment context. Another important type of contextual information is semantic information about the location, such as whether a particular area is a commercial, residential, or recreational space. Many databases containing such semantic information have been built in recent years, and these are increasingly available as Application Programming Interfaces (APIs) for use in, e.g. mobile applications.

5. OVERVIEW OF PUBLICATIONS

This chapter provides an introduction and overview of the five publications included in this compendium thesis. Two of the publications, [P1] and [P2], are excerpted from the book *Geospatial Computing in Mobile Devices*, published by Artech House in 2014. Two others, [P3] and [P4], were published in the peer-reviewed open access journal *Sensors*. The final publication [P5] was published in the 2014 Proceedings of the Position Location and Navigation Symposium (PLANS), jointly organized by the Aerospace and Electronics Systems Society (AESS) and the Institute of Navigation (ION). AESS is an affiliate society of the Institute of Electrical and Electronics Engineers (IEEE).

The remainder of this chapter is organized as follows: Section 5.1 briefly summarizes the publications. Section 5.2 maps the included publications to different research areas. Section 5.3 describes the author's contributions to each publication.

5.1 Summary of Publications

Now, we briefly summarize the included publications.

[P1] presents the concept of context awareness at a conceptual level, as well as tracing its historical development. It presents a conceptual framework for describing context, adapted from the seven elements of circumstance, first introduced by Hermagoras of Temnos in the 2nd Century BC. Our main motivation was that we were not satisfied with any of the existing frameworks in the literature in terms of being comprehensive, simple, yet flexible. The publication also aims to show at a practical level how different aspects of context awareness can be implemented in a mobile device. Examples are given for the Android OS.

[P2] presents the concept of contextual reasoning, which is defined as the process of forming higher-level inferences about context from lower-level information. From this definition, we elaborate a conceptual model of contextual reasoning, which we call the “context pyramid”. The context pyramid describes contextual reasoning as a series of processing steps at different levels, starting with raw sensor data at the base of the pyramid and working up to the peak of the pyramid, where rich context is realized. We argue that machine learning is an ideal technique available for contextual reasoning, and we provide several examples of different machine learning techniques, such as naïve Bayes’ classifiers, Hidden Markov Models (HMM), Bayesian Networks, and Support Vector Machines (SVM).

[P3] combines indoor positioning technologies and smartphone sensors to detect different human activities in an office environment. We provide a real-world implementation of the context pyramid on a smartphone, resulting in a contextual reasoning capability, which we call the “cognitive phone”. The key technologies we utilize include ubiquitous positioning, motion recognition, and human behavior modeling. We combine these technologies into a single probabilistic model, which we call the LoMoCo (Location-Motion-Context) model. In this paper, we demonstrate the feasibility of the fifth level in the context pyramid—Activity-Level Descriptors.

The location accuracy we achieved using WLAN-based indoor positioning was about 2–5 meters, depending on the type of space. The positioning performance was better in corridor areas and the worst in lobby areas. We used Received Signal Strength Indicator (RSSI) as the observable and used the pattern matching or fingerprinting approach for positioning. Training data were collected to estimate a Weibull function representing the WLAN signals in the environment. Finally, for position estimation we used the Histogram Maximum Likelihood algorithm.

We also describe our implementation of a probabilistic method for indoor-outdoor detection based on GPS and WLAN signals. For motion recognition, we used the techniques of supervised learning, described in Chapter 3. We performed extensive feature selection, using the sequential forward selection (SFS) algorithm, from thirteen different features derived from the smartphone sensors. Finally, we evaluated several different supervised learning algorithms such as decision trees and LDA, but the best performance was achieved using a Least Squares-Support Vector Machine (LS-SVM) classifier, which produced an average recall rate of 92.9%. The most common errors were confusion between “sharp turning” and “gradient turning,” which is not surprising because these motions are very similar.

Next, we used Bayesian inference to determine the most probable activity class, using the location and motion modes as inputs. Locations were represented discretely at the room-level. The detected activities included “fetching a coffee,” “fetching water,” “taking a break,” “having lunch,” “working”, and “undefined context,” which included any activity not otherwise defined. Average recall rate for all contexts was 90.3%. The most common misclassifications were when “undefined contexts” were misclassified as one of the other defined activities. This is not surprising given that the training data collected for “undefined context” consisted of various activities, such as fetching paper from a printer or using a toilet. Our results suggest that it is a challenge to build a strong classifier for a “none of the above” type context class.

Finally, we measured the battery drain of the smartphone that polled sensor data at a high rate, similar to the rates used in the above research. We varied the set of sensors that were turned on to see their relative influence. Less than 5% of the battery was used after 40 minutes with only the inertial sensors turned on. The most significant battery drain was when GPS was turned on, in which case around 20% of the batter was drained during 40 minutes. These results support our view that GPS should only be used when necessary in outdoor environments. From these results, the importance of “indoor-outdoor” context awareness is also supported.

[P4] investigates the use of smartphone sensors, geospatial information, and machine learning to sense mobility contexts, including walking, running, driving and using a bus or train. Our aim was to evaluate techniques that could be used in real-time or near-real-time (<5 s). We also measured the computational complexity of the resulting classifiers because this impacts smartphone battery usage.

It is important to point out that for this application, we are not interested in the time complexity of the training phase but rather the testing phase. This is because training can be done offline where time and power resources are not as constrained. Although it is insightful to consider the time complexity of the testing phase analytically, it is not possible to compare the computation time of different classifiers using pure analysis. This is because the time complexity of different classifiers depends on parameters specific to each classifier, and in some cases complexity is dependent of the training data. For example, in decision trees, the time complexity is linear with the number of non-leaf nodes in the tree. Since the size of the tree is dependent on the training data, it is not possible to analyze the time complexity without first building the tree with training data. For this reason, we measured time complexity computationally for our particular data set.

We investigate a wide range of supervised learning techniques for classification, including decision trees (DT), support vector machines (SVM), naïve Bayes classifiers (NB), Bayesian networks (BN), logistic regression (LR), artificial neural networks (ANN) and several instance-based classifiers (KStar, LWL and IBk). A total of seven features were extracted from two different smartphone sensors (GPS and accelerometers). One of the most novel features used was the distance of the user from the nearest train station or bus stop. This was computed using locations from a GIS database. We performed feature selection to identify the most important features from our dataset for detecting mobility context. These results showed that all features except "speedChange" were useful for detecting mobility contexts. In particular, the GIS features appear to be very useful for detecting mobility context, e.g. performance using one classifier improved from 93.8% to 97.1% when the GIS features were added. Individually, the best performing feature was speed, which is perhaps not surprising in the case of mobility context.

In terms of parameter tuning, we focus on the best performing classifier, RandomForest, which is a type of ensemble decision tree algorithm. Using a hold-out set, we tune its parameters to find the optimal performance. RandomForest requires the setting of two parameters, F representing the number of features used in random selection for building the decision trees and K representing the number of trees to grow. Using grid search, we found that the best choice for F is two, especially in the cases where $K > 10$. As K increases, the performance asymptotically improves, but for values above 30, the improvements are very minor. After tuning, average recall rate above 97.5% were achieved[1].

Finally, we measured computational complexity in terms of Central Processing Unit (CPU) time needed for classification, in order to provide a relative comparison between the algorithms in terms of battery usage requirements. As a result of our measurements, we are able to rank the classifiers from lowest to highest complexity as follows: SVM, ANN, LR, BN, DT, NB, IBk, LWL and KStar. CPU times were measured on a desktop PC not a smartphone, so the results only provide information about the relative complexity of the classifiers. The RandomForest algorithm, although it does not generate the simplest classifier in terms of computational cost,

[1] Note that the performance results for the model selection and parameter tuning portions of this paper are not directly comparable because the performances was measured with different test sets.

provides the best performance with reasonable complexity. SVM and ANN are very good in terms of testing complexity. SVM time complexity scales linearly with the number of features and support vectors and also depends on the type of kernel used. ANN scales with the number of neurons and the complexity of the network. RandomForest time complexity is affected mostly by the choice of parameter K. Our CPU measurement times were made with $K = 10$, and at this setting, RandomForest took about 3 times longer for testing compared to SVM. Lastly, according to our results, IBk, LWL, and KStar are very expensive at the testing phase (i.e. the inference phase). Their time complexity scales linearly with the amount of training data used. They are often referred to as "lazy classifiers" because they do the bulk of the computation, not during training, but during testing.

[P5] examines the feasibility of an ice-aware maritime route optimization algorithm and presents a novel method for such purposes. Our aim is to increase the safety and efficiency of maritime transport under icy conditions. Earlier works in this area mainly used numerical methods that could not guarantee global optimum solutions. Our proposed method combines several elements, including (1) a sea spatial model, (2) ship maneuverability model, (3) sea ice model, and (4) ship performance model. The sea spatial model is a rectilinear grid of points, masked by a boolean criteria—whether the depth of the sea is greater than a chosen threshold (normally the draught of the ship). The ship maneuverability model consisted of a set of neighbor grid points with respect to each grid point, defining edges between nodes in a graph. The purpose of this model is to discretize the number of directions in which the ship can travel, making the route optimization algorithm more computationally tractable. The sea ice model used, called HELMI, was developed by the Finnish Meteorological Institute. Lastly, the ship performance model was previously developed by co-authors as originally presented in [27].

Route optimization is performed using the A* algorithm, which is a graph-based search algorithm that uses a heuristic to guide (i.e. speed up) the search process. The heuristic we used was the Euclidean distance between the current search point and the desired destination, divided by the maximum speed of the ship over open water.

The main novelty in this research is the development of an intuitive cost function that takes into account ice conditions and available icebreaker assistance. From a context awareness perspective, the ice conditions and status of the icebreaker system are the main aspects of context utilized in this application. The focus of this paper, however, was not on detecting context but on applying context awareness in order to build up a context-aware navigation service. This research is also the first application of the A* algorithm to long-range maritime route optimization. We present example results based on the method, using the Baltic Sea as a case study. Generated routes are compared with historical routes under the same ice conditions to provide preliminary validation of the method. Further validation will require more extensive research, including field trials using the generated routes in real ice conditions.

5.2 Mapping of Publications to Research Areas

Figure 5.1 presents a mapping between the included publications [P1]-[P5] and different areas of research within the topic of context awareness. These areas can be divided into three broad areas according to the subject matter and methodology used: (1) background and literature review, (2) concepts and theory, and (3) different use case scenarios. Publications [P1] and [P2] fall primarily within the first and second areas, respectively. Publications [P3]-[P5], although containing some elements from the first two areas, mainly deal with different use case scenarios where context awareness can be applied.

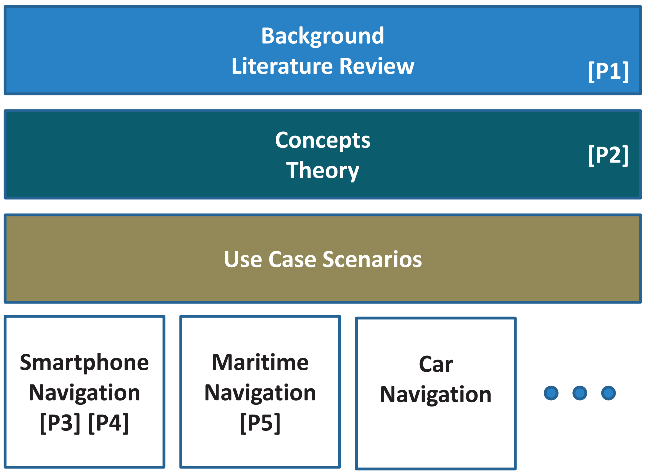


Fig. 5.1: *Mapping of included publications to research areas. Many use case scenarios have yet to be explored. Further discussion on future use case scenarios can be found in Section 6.4.2.*

As Figure 5.1 indicates, there are many potential use case scenarios that are not addressed in this thesis. Our original research plan was to cover as many different use case scenarios as possible, but due to time constraints and project limitations, only two separate use case scenarios could be investigated. Our plans to cover additional use case scenarios will be discussed briefly in Chapter 6.

5.3 Author’s Contributions to the Publications

This section outlines the main contributions of the author of this thesis to the included publications.

[P1]: The thesis author was the main author of this chapter, whereas the co-author provided only editorial comments to a near-final draft. The thesis author conducted all the necessary background literature review and independently decided on the detailed contents of the chapter. The author also came up with the idea to use Hermagoras’s “seven elements of circumstance” to organize and describe the different elements of context. The author also independently identified and collated the various Android Application Programming Interfaces (APIs) relevant to context awareness.

[P2]: The thesis author was the main author of this chapter, whereas the co-author provided only editorial comments to a near-final draft. Similar to the case of [P1], the thesis author conducted all the necessary background literature review and independently decided on the detailed contents of the chapter. The author created all of the figures in the chapter and originated the concept of the “context pyramid”. Finally, one of the main contributions of this chapter is the formulation and description (including figures) of rather complex mathematical concepts (e.g. Support Vector Machines) in such a way that they are easily understandable for someone with basic knowledge in algebra and probability.

[P3]: The thesis author was the second author of this publication, but the first author has affirmed in writing that the thesis author’s contribution was roughly equal to that of the first author. The thesis author contributed equally to the design and implementation of the experiments. Implementation of the data collection application on a smartphone was primarily the responsibility of the thesis author. He also assisted in the analysis of the test results. In addition, he was the originator of the idea used for indoor-outdoor detection based on GPS signal-to-noise ratio and WiFi signal strength and implemented this method on a smartphone. The thesis author contributed to the development of the WiFi fingerprinting indoor positioning system and prepared the test environment by measuring and marking reference points and setting up additional WiFi access points. As stated earlier, he was the originator of the "context pyramid" concept, which is also described and employed in this paper. He was also the originator of the idea of using a graph-based grouping of the reference points. Lastly, he contributed to the preparation of the manuscript, including writing some sections and editing it in its entirety.

[P4]: The thesis author was the sole author of this publication and received assistance only in data collection, as well as general guidance from his supervisors. He also implemented all the necessary software used in the experiments, apart from the Weka software platform used in the data analysis. Some extensions to Weka, in terms of automating analysis and integrating Weka with Matlab, were also implemented by the author.

[P5]: The thesis author was the first author of this publication and was the originator of the idea of using a graph-based approach and the A* algorithm for ice-aware route optimization. He formulated the concept of combining a sea spatial model, ship maneuverability model, sea ice model, and ship performance model. In particular, the ship maneuverability model, which defines the graph structure and discretizes the ship's maneuverability, was the thesis author's own invention. Furthermore, he implemented the optimization algorithm in Matlab, basing the implementation only roughly on an open source implementation. The cost function was developed mainly by the thesis author, although various preliminary ideas were discussed together with the second author. The parts of the method that were not developed or provided by the thesis author were the resistive ship speed model, the ice data, and the historical ship data, all of which were provided by co-authors. Finally, the thesis author led the manuscript preparation, writing most sections, preparing all figures, and editing the manuscript in its entirety.

6. CONCLUSIONS

This chapter offers some conclusions based on our research. It is organized as follows: Section 6.1 briefly summarizes the thesis. Section 6.2 outlines our main findings. Section 6.3 explains the significance or potential impact of the results. Section 6.4 describes our future work planned in the areas addressed by this thesis. Finally, Section 6.5 provides a few concluding remarks.

6.1 Summary

The overall goal of this thesis was to improve our understanding of how computing devices can better understand us and our needs. As argued in this thesis, such understanding is often embodied, at least partly, in a concept known as context awareness. The primary method used to endow computers with context awareness has been—and we argue it will continue to be—machine learning.

In examining these topics, we have narrowed the focus to application areas related to navigation. Despite this narrowing of application areas, there are still many diverse needs in navigation, and this thesis focused on three particular use cases within navigation where context awareness is deemed beneficial: (1) detecting of different human activities inside a typical office environment to improve indoor location tracking, (2) detecting different "mobility contexts" of a smartphone user to improve outdoor location tracking, and (3) enabling "ice aware" route optimization for ships sailing in ice-covered waters to improve and automate the route planning needs of such ships. These use cases demonstrate the breadth of potential application areas of context-aware technology. Three of the included publications ([P3]–[P5]) aim to improve the state-of-the-art in these application areas by introducing either novel methods, novel combinations of existing methods, or in-depth analysis of the performance of existing methods.

In addition to examining these application areas, this thesis has extensively reviewed the literature concerning context awareness and machine learning. In presenting and summarizing these topics, we have attempted to provide clear, tutorial-like examples, in order to aid readers unfamiliar with these subjects. We also provide a chapter on navigation to familiarize the reader with the intended application area.

In presenting the conceptual underpinnings of context awareness, we have introduced two conceptual frameworks for understanding context awareness and contextual reasoning. The first was adapted from the writings of an ancient Greek orator Hermagoras of Temnos, known as the "seven circumstances". The second, which we have dubbed the "context pyramid," describes the process of contextual reasoningin terms of six levels ranging from raw data to "rich context". These two frameworks, general in nature, can assist the researcher and developer aiming to build context-aware systems by dividing the problem up into different categories of contextual information and steps in contextual reasoning.

On the topic of machine learning, this thesis has examined the original goal of machine learning, as envisioned by pioneers such as Arthur Samuel. We emphasize the concept of *automatic learning* using computer chess as an example. We then examine the modern notion of machine learning, including the two major types, supervised learning and unsupervised learning. We provide a tutorial-like example of both types of learning, using an example problem from context awareness.

During this overview on machine learning, we have emphasized the importance and benefits of automatic learning. That is, supervised learning usually requires manual labeling of training data, whereas unsupervised learning can largely meet the desire for automated learning, although it often requires some human interpretation of the results.

Finally, the included publications provide further details on machine learning and its application to context awareness, and in particular [P3] and [P4] demonstrate the use of machine learning in problems related to navigation. Lastly, [P5] provides an example application of context awareness in the field of navigation, i.e. an ice-aware route optimization method.

6.2 Main Findings

In Section 1.2, we identified two overall research questions addressed by this thesis. These questions are implicitly discussed in various parts of the thesis, including the five publications. In this section, we explicitly summarize our findings regarding these questions.

What are the benefits and constraints of introducing context awareness in navigation?

The main benefit of introducing context awareness in navigation is to increase the level of automation that can be achieved in performing navigation functions. Although humans are inherently good at recognizing and understanding context, computers are relatively deficient in this ability. Nonetheless, the techniques described in this thesis and other state-of-the-art context awareness research help to endow computers with such abilities. Several examples have been given related to the three tasks investigated in this thesis, and their application to navigation has been described. Without these abilities, the user of a navigation system would have a greater burden in terms of explicitly switching into different navigation modes, or performing manual integration of data.

In the particular case of maritime navigation applications, awareness about ice conditions (as a function of space and time) can be exploited to perform automated route optimization. Such capability could augment or even replace the currently human-intensive task of route planning performed by crews of ships sailing in ice-covered waters. Our research showed that graph-based approaches are feasible for modeling maritime transportation in ice-covered waters and that the A* algorithm can be applied to find optimal paths. In order to realize an implementation of the A* algorithm, our research presents a simple but novel cost function that takes into account the operational constraints posed by ice breaker assistance. Essentially, this cost function captures contextual information about a ship's theoretical speed through an ice field, taking into account the ship's own ice-breaking performance and possible assistance from an ice breaker. The results of this method allow different proposed routes to be compared, in terms of voyage times, and provides decision support for the final selection of the ship's planned route.

The main constraint of context awareness in navigation is that the level of detail concerning context that can be recognized by computers is relatively low. Although the abilities in the state-of-the-art are rapidly increasing, still a human can much better describe the important elements of context in a succinct yet rich manner. Furthermore, the models developed for context awareness are not error free. In our research work, we achieved successful context recognition in the range of 90% to 98% of the studied cases. Efficient, fail-safe methods to deal with errors from context awareness systems must be developed in an application-specific manner. This topic has not been investigated in this thesis, and the literature on the subject remains scarse.

Another constraint of context awareness in navigation is that context awareness systems consume resources, in terms of the power, mass, and cost used for sensors, computational units, etc. If the improvement to the navigation system is only marginal, or if the user does not feel significantly burdened by the manual alternative to a context-aware system, then it will be difficult to justify the use of these extra resources. This research area is perhaps still to immature to determine with certainty whether the benefits associated with context awareness justify the costs, especially in the domain of navigation.

How can machine learning be used to build context or situation awareness, in order to solve problems in navigation?

This thesis, in particular [P3] and [P4], shows that machine learning is a powerful tool to enable context awareness, in order to solve various problems in navigation. We employ a number of different supervised learning algorithms, and in particular the method of building classification models via supervised learning suits the problem area very well. Although we did not compare the use of supervised learning against other types of model building, we can at least conclude that machine learning produces performance levels that are quite promising from a research standpoint. We have shown that smartphones can reliably detect different mobility contexts (>97% recall rate) and detect different office-environment activities (>90% recall rate). We have also introduced a method to detect whether a user is indoors or outdoors. Existing algorithms from supervised learning provide adequate levels of performance for these use cases, although generalization to large user populations will require the collection of more extensive training data. Especially in the case of context-aware smartphone applications, context awareness is presently feasible and can be realized using existing machine learning techniques.

Another point concerning the above research question concerns how machine learning techniques should be evaluated. Although machine learning constitutes a powerful set of methods for endowing computers with context awareness, a systematic evaluation of different available machine learning algorithms should be undertaken when applying machine learning to the problem of context awareness, especially if the aim is to maximize performance. This important fact is often overlooked by navigation researchers working on context awareness. After evaluating the performance of 20 different supervised learning algorithms, we determined that the RandomForest algorithm performed the best on our dataset. Our results also showed that performance is optimized only after applying extensive feature selection and parameter tuning. Another general recommendation from our research is that, due to the need to train many different classification models (with various feature sets and parameter settings), a high level of automation for this type of analysis is desirable, and we developed some software tools to improve the automation of this type of analysis.

One limitation of supervised learning is that the cost of obtaining labeled data is, in general, quite high. This issue will be discussed further in Section 6.4.3.

In this thesis, we have primarily investigated machine learning techniques that are built upon assumptions that the input data are independent and identically distributed. This is one major limitation of our research, since it is clear that most aspects of context are temporally correlated. Some machine learning techniques which exploit temporal correlations, such as HMMs are discussed in [P2], but thus far we have not applied these techniques in our context awareness research. This topic will be discussed further in Section 6.4.1.

In summary, our overall research on the use of machine learning in context awareness shows the feasibility of developing context-aware navigation applications for the three use-case scenarios investigated. In addition, our research suggests many other applications of context awareness are evident in emerging technologies related to navigation. We believe context awareness will play an even stronger role in navigation in the future, especially as so-called “smart devices” continue to proliferate.

6.3 Significance of the Results

This thesis contributes to the overall body of knowledge on context awareness, which as previously discussed, has many potential applications in the field of navigation. We have introduced two general and flexible frameworks related to context awareness—one for the systematic encoding of contextual information and the other for the processing of raw sensor data into "rich context". These frameworks serve as a methodological skeleton on which other researchers and developers can build new context-aware systems, not only in navigaion but more generally.

Much of this thesis focuses on context awareness that can be achieved using only the sensors in smartphones. Because of the widespread prevalence of smartphones in modern society, the results of smartphone-based context awareness research have a strong potential for widespread adoption. We can compare this to previous context awareness studies that rely on custom sensors being installed in the environment. By requiring installation of new hardware into the environment, the cost of adoption increases. According to our knowledge, our research is the first to look at enabling office-related context awareness using only smartphone sensors and standard WLAN access points.

Regarding the performance results reported in our mobility context research, it is difficult to determine definitively whether or not our results represent an improvement over the state-of-the-art. This is due to the fact that we have not compared our results against those obtained with comparable datasets. This issue will be further discussed below in Section 6.4.3. What we can conclude confidently is a methodological fact: that extensive analysis, consisting of evaluating different machine learning algorithms, performing feature selection, and systematically tuning parameters of the algorithm, will result in better performance regardless of the dataset[1]. Surprisingly, many research works in context awareness, especially in the navigation community, overlook this fact and report results from evaluating only one or a few machine learning algorithms and without reporting any results of parameter tuning. We hope that the methodology and results described in this thesis provides a new example as to the benefits of such detailed analysis.

Lastly, our research demonstrates the feasibility of developing an ice-aware maritime

[1] Admittedly, this is not a novel finding.

navigation system, specifically to provide automatic route optimization. Although further refinement is needed, the proposed methods have the potential to provide significant savings for the maritime transportation system. After presenting these methods to Director-General of the Finnish Ministry of Transportation and Communications, Mr. Pekka Plathan, he commented that this research has the potential to save billions for the Finnish maritime industry. Further innovations, improvements, and validation work, however, are required before such savings can be realized.

Ice-aware route optimization can not only bring economic benefits for the maritime industry, but it also has the potential to provide safety benefits. The methods described in this thesis are flexible in that the cost function can be modified to optimize non-economic factors as well. For example, using models for the risk that ice poses towards damaging ships or getting the ship completely stuck in the ice, one can design an appropriate cost function aimed to minimize these risks.

6.4 Future Work

In many ways, this thesis has only scratched the surface in exploring context awareness for navigation applications. In tackling the broader goal "to improve our understanding of how computing devices can better understand us and our needs," we feel even less compelled to declare our work complete. This section outlines some of our planned future work in developing context-aware navigation applications.

Our future work can be divided into three broad categories: (1) future work in the three application areas covered by the included publications, (2) future work in new application areas, and (3) future work that can benefit context awareness broadly. These areas of future work will be discussed in separate sub-sections below.

6.4.1 Future Work in Investigated Applications

In our research on detecting office-environment contexts, we investigated a small number of different workplace contexts, including working in one's office, having lunch, taking a break, and fetching coffee or water. There are obviously a large number of other workplace contexts that could be investigated, such as having a meeting, giving a presentation, having an impromptu conversation, talking on the phone, etc. In our research, we grouped all “non-defined” contexts into a single category called “undefined context”. Also, we conducted this research in only one particular office environment. Extending this research to many diverse office environments and types of work would improve the robustness of the results.

Another way to improve this line of research would be to expand the sources of raw data to include other sensors. For example, smartphones have microphones, and sound could be an important source of contextual information. In fact, we informally explored using audio as a feature, but there are some challenges in this regard. For example, when the phone is in the pocket, the audio signal becomes very muffled and contact with clothing can cause loud undesirable signals. Nonetheless, we believe audio is an important source of contextual information, and we aim to explore this further in the future.

Also, different social context aspects of the workplace environment will be included in our future research. In this thesis, we did not address social context at any depth, but especially in an office environment, it should be feasible to recognize different social contexts because in many workplaces the identities of most of the people present are largely known. Using various proximity sensing technologies, a smartphone could apply contextual reasoning about different social contexts, such as “with the boss,” “with subordinates,” “with colleagues,” “with a customer,” etc. Such contextual information may not be needed necessarily for navigation applications, but it would certainly have other applications related to mobile computing.

On the subject of mobility contexts, we also plan to expand the range of contexts under investigation, such as cycling, riding trams, riding metros, etc. In addition, we plan to investigate whether we can recognize a number of other mobility-related leisure activities, such as hiking, berry or mushroom picking[2], playing golf, dog-walking, etc.

Also, in this thesis, for detecting mobility context we only utilized two smartphone sensor types, namely GPS and accelerometers. In the future, we aim to include other types of sensors, such as gyroscopes, pressure sensors, microphones, and light sensors.

Lastly, we plan to expand the number of test subjects participating in the collection

[2] These are popular leisure activities in Finland.

of training data. This is important to ensure that the trained models are robust. In this thesis, we asked the test subjects to keep the smartphone in a particular location (pants pocket), so in future research we will also study the effects of placement of the smartphone in different locations, such as a backpack, handbag, belt "holster," etc.

In both [P3] and [P4], all of the machine learning algorithms utilized operate independently on each data sample. That is, no time dependence between the data samples is exploited. As already mentioned in Section 6.2, contexts are strongly correlated with time, so we would expect to improve context recognition performance by using models that exploit these temporal correlations. Our future work will focus on such models and algorithms, including HMMs, Conditional Random Fields (CRF), and Markov chain Monte Carlo (MCMC) methods.

On the subject of ice-aware route optimization, further work is needed to validate the proposed method. This should include further analysis of historical data from Automatic Identification System (AIS), as well as simulator-based studies and actual testing of routes at sea. Also, in this thesis the route optimization method focused on minimizing travel times for ships, but in the future other aspects should be investigated, such as fuel usage, operational efficiency, safety, and reliability. Lastly, contextual information regarding the ice conditions should be enriched compared to the model used in this thesis. For example, the current model does not take into account ice compression, which can have a large effect on ice-going ship performance.

6.4.2 Future Applications

In Figure 5.1 we hinted at future application areas or "use case scenarios," many of which fall outside the domain of navigation. Several of the highlighted use case scenarios are part of near future work. For example, in one recently initiated project, we aim to develop a "tactical situation awareness system" for soldiers.

Military applications of context awareness are particularly promising because the cost limitations are not as strict as in other application areas and specially-designed sensors can be installed, e.g. attached to various body parts of a soldier (helmet, boots, chest, etc.) or to other military equipment, providing a rich set of raw sensor data from which to generate context awareness. On the other hand, in military applications, reliability requirements are very high, and typically there is a strong requirement for real-time functionality. For example, if a system is designed to detect when a soldier is in danger or injured, then false negatives, as well as false positives could prove very costly.

Another application area that has strong potential is healthcare and fitness monitoring. With the growing popularity of "wearable devices," such as smartwatches and small heart-rate monitors, such applications have greater widespread consumer appeal. Many devices already exist that can, e.g. monitor calorie usage by tracking steps, but it remains a challenge to reliably and automatically detect different activities such as walking, running, cycling, hiking, etc. This is, of course, strongly overlapping with the topic of [P4], but we believe healthcare and fitness monitoring can go much beyond mere "mobility context" and incorporate other aspects, such as recognizing social interactions, detecting abnormal health or changes to a person's routine that might affect health and fitness, and warning users of dangerous or unhealthy situations. The concept of a "personal health assistant" is not really a matter of science-fiction but could be realized in the coming years. Context awareness and machine learning are the technologies that are likely to make this concept a reality.

6.4.3 General Issues and Potential Solutions

Lastly, we have noticed in our research several general issues that are relevant to context awareness in a broad sense. These issues are summarized as follows:

1. Supervised learning requires labeled data, and labeled data are expensive.
2. There is a lack of standardization in context awareness research.
3. Many context awareness experiments are not easy to repeat or independently verify.

The first general issue above is related to the use of supervised learning, which is often the adopted approach in many research works (such as in [P3] and [P4]). While supervised learning has many advantages compared to unsupervised learning, it can be very costly and time-consuming to generate the required labeled data. Furthermore, it is generally the case that the more data that can be collected, the more performant and reliable the resulting model will be. For example, if we are aiming to develop a context-aware smartphone application that works well across a large population of users, then we will need to collect training data from a large, diverse population of test users. This is very costly, especially in a research setting.

There are two potential solutions to this issue. The first is that researchers and developers would publish and share their training data. This would benefit the overall research community. We have practiced this approach in publication [P4], and there are a few other examples of data sharing in the context awareness literature (e.g. [14] [119]). Generally, this is not a common practice in context awareness research. The second approach would be to collect a sizable amount of labeled training data, and then to supplement it with unlabeled data (which is less costly to collect). Performing machine learning using a combination of labeled and unlabeled data is known as *semi-supervised learning*. This topic is outside the scope of this thesis but will be explored in our future work. As an example of this approach, a research and development team could collect a limited amount of labeled data using its own staff and volunteers and then supplement it with a large amount of crowdsourced unlabeled data. This is exactly the approach we are taking in a recently initiated project called MyGeoTrust (see [120]). The other approach was discussed in Section 3.4; using unlabeled data to strategize about and prioritize the collection of labeled data.

The second general issue has to do with standardization. To put it precisely, there is a lack of standardization in context awareness research, and this issue makes it difficult to compare results among different studies. As described in Chapter 4, context is understood in many different ways, and there is no one "correct" way to categorize and organize the context space. Table 2.1 demonstrates this problem.

This lack of standardization is understandable, due to the fact that different researchers have different applications in mind and different ideas about how to segregate the context space, but it would be more beneficial for the overall research community if some level of standardization were applied. Many context ontologies have been proposed in the literature (e.g. [121] [122] [123] [124] [125]), and one of these could form the basis of a context ontology standard. Then, when presenting results, researchers could reference these standards, i.e. "the following classification results are according to standard X.Y...". Also, there is no reason to limit results to one particular standard; data could be processed, according to several different standards and presented in the same publication. The problem is that no forerunners for a standard have emerged, and no good software tools for working with the proposed ontologies have been made available (to our knowledge). In our future work we aim to contribute to and advance the notion of standard context ontologies, including open source tools for working with such ontologies.

The last general issue we would like to discuss is somewhat related to the second issue, and the solution is ironically similar to the first solution described above. One of the long-standing tenets of scientific research is reproducibility. Experiments should be described in enough detail so that other researchers can independently verify the results. In the case of context awareness research, this means that an independent researcher should be able repeat another researcher's data collection, apply the same algorithms, and achieve similar, if not identical, results. In reality, there are so many complex factors related to the environment, devices, and test subjects that collecting comparable data that produces comparable results is not always realistic.

The solution is straight-forward. As an alternative, context researchers should always publish the data upon which their results are based, along with sufficient documentation so that the data is usable by independent researchers. As already stated, this is rarely done in context awareness research. It is, however, a common practice in the machine learning community to test techniques against benchmark data. For example, the University of California, Irvine (UCI) maintains a repository of over 300 datasets that can be used for machine learning research [126]. Unfortunately, very few of these datasets relate to context awareness. A few other sources of open data for context awareness research exist, including the Mobile Data Challenge (MDC) Dataset collected as part of the Lausanne Data Collection Campaign [127] and data from the University of Helsinki's "Context project" [128]. We aim to follow open data practices in our future work and also to actively promote this practice, either by promoting the use of UCI's machine learning repository or by setting up a dedicated portal for context awareness research.

6.5 Concluding Remarks

The remainder of this thesis consists of reprints of the five included publications described earlier. The order of the publications has been chosen to go from the most general to more specific and detailed applications. They can be read, however, in any order, depending on the reader's specific interests.

BIBLIOGRAPHY

[1] T. Mitchell, B. Buchanan, G. DeJong, T. Dietterich, P. Rosenbloom, and A. Waibel, "Machine learning," *Annual Review of Computer Science*, vol. 4, pp. 417–433, 1990.

[2] E. Brynjolfsson and A. McAfee, *The second machine age: work, progress, and prosperity in a time of brilliant technologies*. New York: W. W. Norton & Company, 2014.

[3] Canalys.com, Ltd., "64 million smart phones shipped worldwide in 2006." Available online: http://bit.ly/1HSNZBZ, 2007.

[4] Gartner, "Gartner says worldwide smartphone sales reached its lowest growth rate with 3.7 per cent increase in fourth quarter of 2008," March 2009.

[5] N. Mawston, "Worldwide smartphone population tops 1 billion in Q3 2012." Available online: http://bit.ly/1DZuyYo, October 2012. (archived on 26 February 2015).

[6] "Smartphone users expected to hit 2.5 billion next year." *The Korea Times*, Available online: http://bit.ly/1HSO5cQ, 2014.

[7] J. McCarthy, M. Minsky, N. Rochester, and C. Shannon, "A proposal for the Dartmouth summer research project on artificial intelligence," *AI Magazine*, 2006. Reprint of proposal originally written in 1955.

[8] D. Crevier, *AI: The tumultuous history of the search for artificial intelligence*. Basic Books, Inc., 1993.

[9] P. D. Groves, L. Wang, D. Walter, H. Martin, K. Voutsis, and Z. Jiang, "The four key challenges of advanced multisensor navigation and positioning," in

Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION, pp. 773–792, IEEE, 2014.

[10] S. Abolfazli, Z. Sanaei, A. Gani, F. Xia, and L. T. Yang, "Rich mobile applications: genesis, taxonomy, and open issues," *Journal of Network and Computer Applications*, vol. 40, pp. 345–362, 2014.

[11] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[12] M. Petovello, "How does a GNSS receiver estimate velocity?." Available online: http://bit.ly/1GwTuJN, March/April 2015. Inside GNSS.

[13] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu, "Transportation mode detection using mobile phones and GIS information," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 54–63, November 2011.

[14] M.-C. Yu, T. Yu, S.-C. Wang, C.-J. Lin, and E. Y. Chang, "Big data small footprint: The design of a low-power classifier for detecting transportation modes," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1429–1440, 2014.

[15] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Proceedings of the national conference on artificial intelligence*, vol. 20, pp. 1541–1546, July 2005.

[16] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh, "Using mobile phone barometer for low-power transportation context detection," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pp. 191–205, ACM, 2014.

[17] H. Yan and T. Selker, "Context-aware office assistant," in *Proceedings of the 5th international conference on Intelligent user interfaces*, pp. 276–279, ACM, 2000.

[18] M. Danninger and R. Stiefelhagen, "A context-aware virtual secretary in a smart office environment," in *Proceedings of the 16th ACM international conference on Multimedia*, pp. 529–538, ACM, 2008.

[19] H. W. Gellersen, A. Schmidt, and M. Beigl, “Multi-sensor context-awareness in mobile devices and smart artifacts,” *Mobile Networks and Applications*, vol. 7, no. 5, pp. 341–351, 2002.

[20] P. Nurmi and P. Floréen, “Reasoning in context-aware systems: a position paper.” Available online: http://www.cs.helsinki.fi/u/ptnurmi/papers/positionpaper.pdf, 2004.

[21] M. Tähti, V.-M. Rautio, and L. Arhippainen, “Utilizing context-awareness in office-type working life,” in *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pp. 79–84, ACM, 2004.

[22] Y. Manabe, H. Saito, K. Akiyama, R. Ikeda, S. Kanda, and K. Sugawara, “Perceptual functions for context-awareness of an office worker,” in *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, pp. 583–589, IEEE, 2010.

[23] L. M. Marti, R. Mayor, and S. M. Ma, “Managing states of location determination.” US Patent Application 13/715,710, 19 June 2014.

[24] R. E. Guinness, “Beyond where to how: A machine learning approach for sensing mobility contexts using smartphone sensors,” in *Proceedings of the 26th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2013)*, (Nashville, TN), pp. 2868–2879, September 2013.

[25] J.-H. Nam, I. Park, H. J. Lee, M. O. Kwon, K. Choi, and Y.-K. Seo, “Simulation of optimal arctic routes using a numerical sea ice model based on an ice-coupled ocean circulation method,” *International Journal of Naval Architecture and Ocean Engineering*, vol. 5, no. 2, pp. 210–226, 2013.

[26] M. Choi, H. Chung, H. Yamaguchi, and K. Nagakawa, “Arctic sea route path planning based on an uncertain ice prediction model,” *Cold Regions Science and Technology*, vol. 109, pp. 61–69, 2015.

[27] V. Kotovirta, R. Jalonen, L. Axell, K. Riska, and R. Berglund, “A system for route optimization in ice-covered waters,” *Cold Regions Science and Technology*, vol. 55, no. 1, pp. 52–62, 2009.

[28] C. R. Colon, "An Efficient GPS Position Determination Algorithm." *Air Force Institute of Technology*, Available online: http://1.usa.gov/1KFZnPK, 1999.

[29] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.

[30] R. Chen and R. E. Guinness, *Geospatial computing in mobile devices*. Boston: Artech House, 2014.

[31] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2006.

[32] J. Haverinen and A. Kemppainen, "Global indoor self-localization based on the ambient magnetic field," *Robotics and Autonomous Systems*, vol. 57, no. 10, pp. 1028–1035, 2009.

[33] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[34] "Luciad - geospatial situational awareness." *Company website:* http://www.luciad.com/, 2015.

[35] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[36] A. Turing, *Chess*, ch. The Essential Turing, Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma. Oxford University Press, 2004. Reprint of paper originally printed in 1953.

[37] C. E. Shannon, "XXII. programming a computer for playing chess," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[38] F.-H. Hsu, *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press, 2002.

[39] D. Spicer and K. Tashev, "The quest to build a thinking machine: A history of computer chess," *CompuComputer: New exhibit showcases game's past and museum's future*, May 2006. A publication of the Computer History Museum.

[40] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, New Jersey: Prentice Hall, 3rd edition ed., 2010.

[41] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[42] K. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[43] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[44] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[45] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[46] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Applied statistics*, pp. 100–108, 1979.

[47] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *SIGMOD Rec.*, vol. 28, pp. 49–60, June 1999.

[48] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[49] J. A. Bilmes *et al.*, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[50] N. Vlassis and A. Likas, "A greedy EM algorithm for gaussian mixture learning," *Neural processing letters*, vol. 15, no. 1, pp. 77–87, 2002.

[51] P. Smyth, "Note set 4: Finite mixture models and the EM algorithm." Lecture notes for course on Probabilistic Learning at University of California–Irvine, Available online: http://www.ics.uci.edu/ smyth/courses/cs274/notes/notes4.pdf, 2015.

[52] P. W. Foltz, W. Kintsch, and T. K. Landauer, "The measurement of textual coherence with latent semantic analysis," *Discourse Processes*, vol. 25, no. 2, pp. 285–307, 1998.

[53] M. Bazire and P. Brézillon, “Understanding context before using it,” in *Modeling and Using Context: Lecture Notes in Computer Science* (A. Dey, B. Kokinov, D. Leake, and R. Turner, eds.), Springer Berlin/Heidelberg, 2005.

[54] J. McCarthy, “Notes on formalizing context,” in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, (San Mateo, California), pp. 555–562, Morgan Kaufmann, 1993.

[55] “Context.” *Merriam-Webster Dictionary*, Available online: http://bit.ly/1LK7K2O, 2015.

[56] V. Akman and M. Surav, “Steps toward formalizing context,” *AI magazine*, vol. 17, no. 3, p. 55, 1996.

[57] Wikipedia, “Five Ws — Wikipedia, the free encyclopedia.” Available online: http://bit.ly/1SIXRkI, 2015.

[58] B. S. Bennett, “Hermagoras of temnos,” *Classical Rhetorics and Rhetoricians: Critical Studies and Sources*, p. 187, 2005.

[59] J.-y. Hong, E.-h. Suh, and S.-J. Kim, “Context-aware systems: A literature review and classification,” *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.

[60] B. Schilit and M. Theimer, “Disseminating active map information to mobile hosts,” *Network, IEEE*, vol. 8, no. 5, pp. 22–32, 1994.

[61] M. Weiser, “The computer for the 21st century,” *Scientific American*, vol. 265, no. 3, pp. 94–104, 1991.

[62] R. Want, A. Hopper, V. Falcao, and J. Gibbons, “The active badge location system,” *ACM Transactions on Information Systems (TOIS)*, vol. 10, no. 1, pp. 91–102, 1992.

[63] T. Moran and P. Dourish, “Introduction to this special issue on context-aware computing,” *Human-Computer Interaction*, vol. 16, no. 2-4, pp. 87–95, 2001.

[64] J. McCarthy, *Programs with common sense*. Defense Technical Information Center, 1963.

[65] J. McCarthy, “Generality in artificial intelligence,” *Communications of the ACM*, vol. 30, no. 12, pp. 1030–1035, 1987.

[66] D. Foxvog, “Cyc,” in *Theory and Applications of Ontology: Computer Applications*, pp. 259–278, Springer, 2010.

[67] R. V. Guha, *Contexts: a formalization and some applications*, vol. 101. Stanford University Stanford, CA, 1991.

[68] J. McCarthy and S. Buvac, “Formalizing context (expanded notes),” in *Computing Natural Language, CSLI, Lecture Notes* (A. Aliseda, R. J. van Glabbeek, and W. D., eds.), pp. 13–50, Center for the Study of Language and Information, Stanford University, 1998.

[69] S. Buvač and I. A. Mason, “Propositional logic of context,” in *Proceedings of the eleventh national conference on artificial intelligence*, sn, 1993.

[70] P. Brézillon, “Context in artificial intelligence: I. a survey of the literature,” *Computers and artificial intelligence*, vol. 18, pp. 321–340, 1999.

[71] V. Akman, “Context in artificial intelligence: a fleeting overview,” in *La svolta contestuale* (C. Penco and V. Akman, eds.), Milano: McGraw-Hill, 2002.

[72] M. Tähti, V.-M. Rautio, and L. Arhippainen, “Utilizing context-awareness in office-type working life,” in *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pp. 79–84, ACM, 2004.

[73] M. Heiskala, E. Palomäki, M. Vartiainen, K. Hakkarainen, and H. Muukkonen, “A research framework for the smartphone-based contextual study of mobile knowledge work,” in *Design, User Experience, and Usability. User Experience Design for Diverse Interaction Platforms and Environments*, pp. 246–257, Springer, 2014.

[74] X. Huang, J. Weng, and Z. Zhang, “Office presence detection using multimodal context information,” in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP’04). IEEE International Conference on*, vol. 3, pp. iii–773, IEEE, 2004.

[75] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, IEEE, 1999.

[76] Y. Manabe, H. Saito, K. Akiyama, R. Ikeda, S. Kanda, and K. Sugawara, “Perceptual functions for context-awareness of an office worker,” in *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, pp. 583–589, IEEE, 2010.

[77] K. Kiyokawa, M. Hatanaka, K. Hosoda, M. Okada, H. Shigeta, Y. Ishihara, F. Ooshita, H. Kakugawa, S. Kurihara, and K. Moriyama, “Owens luis–a context-aware multi-modal smart office chair in an ambient environment,” in *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pp. 1–4, IEEE, 2012.

[78] K. K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P. J. Rentfrow, “Smartphone sensing offloading for efficiently supporting social sensing applications,” *Pervasive and Mobile Computing*, vol. 10, pp. 3–21, 2014.

[79] N. Kern, B. Schiele, and A. Schmidt, “Multi-sensor activity context detection for wearable computing,” in *Ambient Intelligence*, pp. 220–232, Springer, 2003.

[80] S. Pirttikangas, K. Fujinami, and T. Nakajima, “Feature selection and activity recognition from wearable sensors,” in *Ubiquitous Computing Systems*, pp. 516–527, Springer, 2006.

[81] H. Park, J. Park, H. Kim, J. Jun, S. Hyuk Son, T. Park, and J. Ko, “Relisce: Utilizing resource-limited sensors for office activity context extraction,” *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 45, pp. 1151–1164, Aug 2015.

[82] M. J. Duncan, H. M. Badland, and W. K. Mummery, “Applying GPS to enhance understanding of transport-related physical activity,” *Journal of Science and Medicine in Sport*, vol. 12, no. 5, pp. 549–556, 2009.

[83] S. A. Hoseini-Tabatabaei, A. Gluhak, and R. Tafazolli, “A survey on smartphone-based systems for opportunistic user context recognition,” *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 27, 2013.

[84] V. Pejovic and M. Musolesi, “Anticipatory mobile computing: A survey of the state of the art and research challenges,” *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, p. 47, 2015.

[85] Y. Kwon, K. Kang, and C. Bae, “Unsupervised learning for human activity recognition using smartphone sensors,” *Expert Systems with Applications*, vol. 41, no. 14, pp. 6067–6074, 2014.

[86] F. Foerster and J. Fahrenberg, “Motion pattern and posture: correctly assessed by calibrated accelerometers,” *Behavior research methods, instruments, & computers*, vol. 32, no. 3, pp. 450–457, 2000.

[87] S.-W. Lee and K. Mase, “Activity and location recognition using wearable sensors,” *IEEE pervasive computing*, vol. 1, no. 3, pp. 24–32, 2002.

[88] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, “A hybrid discriminative/generative approach for modeling human activities,” in *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 766–772, 2005.

[89] J. Pärkkä, M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen, “Activity classification using realistic data from wearable sensors,” *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 1, pp. 119–128, 2006.

[90] J. Suutala, S. Pirttikangas, and J. Röning, “Discriminative temporal smoothing for activity recognition from wearable sensors,” in *Ubiquitous Computing Systems* (H. Ichikawa, W.-D. Cho, I. Satoh, and H. Youn, eds.), vol. 4836 of *Lecture Notes in Computer Science*, pp. 182–195, Springer Berlin Heidelberg, 2007.

[91] K. Kunze and P. Lukowicz, “Dealing with sensor displacement in motion-based onbody activity recognition systems,” in *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 20–29, ACM, 2008.

[92] J. Yin, Q. Yang, and J. Pan, “Sensor-based abnormal human-activity detection,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 8, pp. 1082–1090, 2008.

[93] J. Yang, "Toward physical activity diary: motion recognition using simple acceleration features with mobile phones," in *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*, pp. 1–10, ACM, 2009.

[94] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, pp. 13:1–13:23, 2010.

[95] L. Pei, R. Chen, J. Liu, W. Chen, H. Kuusniemi, T. Tenhunen, T. Kröger, Y. Chen, H. Leppäkoski, and J. Takala, "Motion recognition assisted indoor wireless navigation on a mobile phone," in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010)*, pp. 3366–3375, 2010.

[96] K. Frank, M. Nadales, P. Robertson, and M. Angermann, "Reliable real-time recognition of motion related human activities using MEMS inertial sensors," in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010)*, pp. 2919–2932, 2001.

[97] L. Pei, R. Chen, J. Liu, H. Kuusniemi, Y. Chen, T., and Tenhunen, "Using motion-awareness for the 3d indoor personal navigation on a smartphone," in *Proceedings of the 24th International Technical Meeting of The Institute of Navigation (ION GNSS 2011), Portland, Oregon, USA, September 20-23*, 2011.

[98] M. Susi, D. Borio, and G. Lachapelle, "Accelerometer signal features and classification algorithms for positioning applications," in *Proceedings of the 2011 International Technical Meeting of The Institute of Navigation*, pp. 158–169, 2011.

[99] J. B. Bancroft, D. Garrett, and G. Lachapelle, "Activity and environment classification using foot mounted navigation sensors," in *Proceedings of the 2012 Indoor Positioning and Indoor Navigation (IPIN) Conference*, 2012.

[100] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly

support vector machine," in *Ambient assisted living and home care*, pp. 216–223, Springer, 2012.

[101] M. Susi, V. Renaudin, and G. Lachapelle, "Motion mode recognition and step detection algorithms for mobile phone users," *Sensors*, vol. 13, no. 2, pp. 1539–1562, 2013.

[102] T. Feng and H. J. Timmermans, "Transportation mode recognition using gps and accelerometer data," *Transportation Research Part C: Emerging Technologies*, vol. 37, pp. 118–130, 2013.

[103] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, p. 13, ACM, 2013.

[104] L. Stenneth, *Detecting Human Activities Using Smartphones and Maps*. Ph.D. thesis, University of Illinois at Chicago, February 2014.

[105] H. Xia, Y. Qiao, J. Jian, and Y. Chang, "Using smart phone sensors to detect transportation modes," *Sensors*, vol. 14, no. 11, pp. 20843–20865, 2014.

[106] M. Elhoushi, J. Georgy, M. Korenberg, and A. Noureldin, "Robust motion mode recognition for portable navigation independent on device usage," in *Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION*, pp. 158–163, IEEE, 2014.

[107] J. Parviainen, J. Bojja, J. Collin, J. Leppänen, and A. Eronen, "Adaptive activity and environment recognition for mobile phones," *Sensors*, vol. 14, no. 11, pp. 20753–20778, 2014.

[108] J.-H. Chiang, P.-C. Yang, and H. Tu, "Pattern analysis in daily physical activity data for personal health management," *Pervasive and Mobile Computing*, vol. 13, pp. 13–25, 2014.

[109] J.-M. Yu and S.-B. Cho, "A low-power context-aware system for smartphone using hierarchical modular Bayesian networks," in *Hybrid Artificial Intelligent Systems*, pp. 543–554, Springer, 2015.

[110] M. Choi, H. Chung, H. Yamaguchi, and L. W. A. De Silva, "Application of genetic algorithm to ship route optimization in ice navigation," in *Proceedings*

of the International Conference on Port and Ocean Engineering Under Arctic Conditions, 2013.

[111] I. H. Park, J. H. Nam, and K. S. Choi, “A graphical approach to determine the optimal sea route of icebreakers in arctic region,” in *Proceedings of the Korean Association of ocean science and technology societies join conference*, June 2011. Published in Korean.

[112] J. Esa, “Fuel and economic efficiency of an ice-going vessel on the northern sea route,” Master of Science thesis, Aalto University, June 2015.

[113] J. Montewka, F. Goerlandt, P. Kujala, and M. Lensu, “Towards probabilistic models for the prediction of a ship performance in dynamic ice,” *Cold Regions Science and Technology*, vol. 112, pp. 14–28, 2015.

[114] D. LaPrairie, M. Wilhelmson, and K. Riska, *A transit simulation model for ships in Baltic ice conditions : Documentation of the calculation routine*. Helsinki University of Technology, 1995.

[115] P. Valanto, S. J. Jones, E. Enkvist, and K. Izumiyama, “The resistance of ships in level ice,” *Transactions of the Society of Naval Architects and Marine Engineers*, vol. 109, pp. 53–83, 2001.

[116] B. H. Fock, A. Beitsch, D. Broehan, M. Dobynin, A. Gierisch, L. Kaleschke, T. Pohlmann, and K. H. Schlünzen, “Ice forecast and route optimization,” 2012. Poster presented at the SMOSIce User Workshop, Hamburg, Germany.

[117] M. Dobrynin, B. H. Fock, A. M. Gierisch, T. Pohlmann, L. Kaleschke, H. Schlünzen, *et al.*, “Prediction of arctic sea ice for ship routing: Forecast experiment and ship cruise,” in *OTC Arctic Technology Conference*, Offshore Technology Conference, 2015.

[118] A. K. Dey and G. D. Abowd, “Towards a better understanding of context and context-awareness,” Tech. Rep. GIT-GVU-99-22, Georgia Institute of Technology, 1999.

[119] F. J. Ordónez, P. de Toledo, and A. Sanchis, “Activity recognition using hybrid generative/discriminative models on home environments using binary sensors,” *Sensors*, vol. 13, no. 5, pp. 5460–5477, 2013.

[120] R. E. Guinness, H. Kuusniemi, J. Vallet, T. Sarjakoski, J. Oksanen, M. Islam, M. Syeed, H.-M. Halkosaari, P. Kettunen, M. Laakso, and M. Rönneberg, "MyGeoTrust: A platform for trusted crowdsourced geospatial data," in *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, (Tampa, Florida), 2015.

[121] H. Guermah, T. Fissaa, H. Hafiddi, M. Nassar, and A. Kriouile, "An ontology oriented architecture for context aware services adaptation," *arXiv preprint arXiv:1404.3280*, 2014.

[122] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *The Knowledge Engineering Review*, vol. 18, no. 03, pp. 197–207, 2003.

[123] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology based context modeling and reasoning using OWL," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pp. 18–22, Ieee, 2004.

[124] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," in *Proceedings of communication networks and distributed systems modeling and simulation conference*, vol. 2004, pp. 270–275, 2004.

[125] T. Strang, C. Linnhoff-Popien, and K. Frank, "CoOL: A context ontology language to enable contextual interoperability," in *Distributed applications and interoperable systems*, pp. 236–247, Springer, 2003.

[126] M. Lichman, "UCI machine learning repository." Available online: http://archive.ics.uci.edu/ml/, 2015.

[127] J. K. Laurila, D. Gatica-Perez, I. Aad, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. Miettinen, "The mobile data challenge: Big data for mobile computing research," in *Pervasive Computing*, no. EPFL-CONF-192489, 2012.

[128] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen, "ContextPhone:

a prototyping platform for context-aware mobile applications," *Pervasive Computing, IEEE*, vol. 4, pp. 51–59, Jan 2005.

PUBLICATIONS

PUBLICATION 1

R. Chen, **R. E. Guinness**, “Context Awareness” in *Geospatial computing in mobile devices*. Boston: Artech House, ch. 8, pp. 150-170, 2014.

Chapter 8

Context Awareness

Up to this point, this book has focused on two fundamental aspects of geospatial computing: (1) methods to compute one's position and (2) methods to store and visualize location-referenced information. These elements provide the basis for *location awareness* and for creating and consuming mobile LBSs, a topic that was covered in detail in Chapter 7.

The next topic, *context awareness*, may at first seem like a departure from the topics covered in the previous chapters, which focused mainly on *location*. In fact, this chapter is a broadening of our focus because, as we shall see, location awareness falls under the larger umbrella of *context awareness*. We will consider various definitions of this term, but a precursory (and open-ended) definition might simply be that context awareness is the superset of location awareness and many other types of awareness about a mobile user's environment, activities, and state of being.

Although context awareness has existed as a computer science research topic for nearly 20 years, the topic has in recent years received increasing interest, largely due to the proliferation of mobile devices in everyday life, as discussed in Chapter 1. It is simultaneously an extremely promising yet challenging topic, given the wide range of capabilities it encompasses.

This chapter presents context awareness at a conceptual level and provides historical perspective on the development of the field. In addition, we attempt to show at a practical level how different aspects of contextual awareness can be implemented in a mobile device. Due to space limitations, we have selected one popular mobile platform, the Android operating system, in order to show specific code examples of several concepts. We will also point readers toward methods of the Android API classes that they can use to implement their own context-aware applications. Similar capabilities are available in other major mobile platforms, such as iOS and Windows Phone 7/8, but unfortunately space does not allow us to demonstrate them here.

8.1 DEFINING CONTEXT AND CONTEXT AWARENESS

Context is one of those words, like system and information, that is difficult to define precisely. It is used prominently in many fields, including linguistics, psychology, neuroscience, law, and computer science, but ironically the definition of context depends heavily on its context of use. Loosely speaking, we can say that context is a category of information, but there is no general agreement on which types of information belong to this category.

Some researchers have studied the etymology of the word context and have even attempted to formalize and to build consensus concerning the definition [1–4]. Although these semantic topics are beyond the scope of this chapter, we require some working notional definition of context before we can define and examine context awareness (this chapter) or discuss *contextual reasoning* (Chapter 9).

Note that whatever definitions we adopt, they will surely reflect the particular focus of this book, namely mobile geospatial computing. In fact, in defining these concepts, we notice a definite trade-off between generality and concreteness. We have tried to be as general as possible without losing our focus on how context awareness can be implemented in mobile devices.

8.1.1 What Is Context?

In the *Merriam Webster Dictionary* [5], we find two definitions of the word context:

- The parts of a discourse that surround a word or passage and can throw light on its meaning;
- The interrelated conditions in which something exists or occurs: ENVIRONMENT, SETTING.

In this book, we adopt the second definition because we are not directly concerned with human discourse but rather with conditions of an environment or setting (i.e., geospatial information) that can be sensed by machines (i.e., computers and sensors). Clearly, these two definitions are interrelated in the sense that discourse can be (and most usually is) used as a representation of an environment or setting.

In other words, natural language is a common form in which contextual information is encoded. Our focus though is on techniques to sense and represent context automatically. Hence, when we refer to *context*, we refer directly to the conditions in the environment. When we use the term *contextual information*, we refer to representations of context, either in natural language or other form. Furthermore, when we want to explicitly distinguish between the two types of context defined above, we will use the terms *discourse context* and *conditions context*, respectively.[17]

In lieu of a formal and generally accepted definition of context, the above dictionary definition (second one) is broad enough to serve as a working definition for use in exploring context awareness. We will further elucidate this concept with a framework and example below.

There are a few general notes regarding context to cover before we dive deeper into the topic. First, note that our definition implies that the specific "conditions" relevant to context are dependent on the "something" to which the context applies. In other words, context is always specified from a particular viewpoint or *frame of reference* of an object or person. This is similar to the concept in physics, where a physical measurement like velocity is defined within a particular frame of reference. Within the domain of mobile geospatial computing, our frame of reference most often will be the mobile device itself, although oftentimes we take the frame of reference to be the same as that of the mobile user.

Second, we must choose some techniques for describing a particular context that help to build a framework for our contextual information. If our goal is to describe a particular context in natural language, then we might employ the classic technique of journalism (since journalism is an age-old craft for describing conditions and events), known as the *five Ws*: who, what, where, when, and why [6]. In fact, this technique dates back at least to the late second century BC when Hermagoras of Temnos defined seven elements of circumstance, which include (in addition to the five Ws) "in what manner" and "by what means" [7].

Using these questions as a starting point (with a slightly different order), we list possible elements of a particular context with a demonstrative example:

What: A small, impromptu gathering of colleagues.
Who: Mary, a smartphone user, as well as three of Mary's coworkers who are nearby.
Where: 60.1609°N, 24.5460°E (WGS84); inside the main lobby of the FGI, specifically inside Mary's pocket.
When: Friday, 20 April 2014 at 12:03 p.m.
Why: This gathering occurred because Mary and her colleagues are going out to lunch together. They are waiting for a fifth colleague, Steve, to arrive.

[17] The word context is also used in some image analysis literature to refer to pixels in the vicinity of a particular pixel (e.g., the neighbors of that pixel). See references [8–10] for examples. This choice of terms is, in fact, related to the *discourse* definition of context; just as the surrounding discourse can shed light on a particular word or phrase, the surroundings of a particular pixel can help in interpreting its value. In any case, this meaning of context is in some ways different from conditions context, and we mention this usage merely to caution the reader not to confuse the two.

In what manner: Mary's smartphone is experiencing small, sporadic movements, consistent with the phone being in the pocket of someone who is standing and having a casual conversation.

By what means: All of the above information has been sensed or reasoned by the sensors and software existing in a smartphone, or acquired via a networked resource. In this case, the smartphone is a Samsung Galaxy S4 with Android 4.3 OS, which includes a GNSS receiver, WLAN-based positioning engine, Bluetooth module, microphone and audio analyzer, ambient light sensor, accelerometers, gyroscopes, and magnetometers.

This account of the situation is not likely to yield a Pulitzer prize in journalism, but it is enough to get a basic sense about what the mobile user is doing, as viewed from the perspective of her mobile device. We will see later how this contextual information can be used to create context-aware applications that record a user's activities and even anticipate his or her needs.

There are of course many other contextual elements we could list, including details about the smartphone itself (e.g., battery level and mobile network that it is connected to) or about the immediate surrounding environment (e.g., the weather conditions outside). Like a good journalist, however, we only include the most relevant facts that are necessary to understand the situation at hand. If, for example, the weather outside were particularly noteworthy, we might have included it in the list, but we can assume from its omission that most probably it is nothing exceptional or affecting to the situation.

Last, we note that there are some elements omitted from this list that are certainly both contextual and relevant, such as details about the user's state of mind or what the topic of conversation is. This illustrates the point that our contextual information will in some ways always be incomplete. In the domain of mobile geospatial computing, we are limited to the information that can be obtained or reasoned by a mobile device. As mobile technology rapidly advances, the breadth of information available to mobile devices will surely increase, but it will always face limitations, especially compared to the robust and subtle sensing ability of the human being.

8.1.2 What Is Context Awareness?

Now that we have a basic notion of context and a simple example to aid our understanding, we can define the term *context awareness*. Although this term inherits the same difficulties facing the term context, it is otherwise straight forward to define. Context awareness is the quality of having knowledge (being aware) of context. We say a device or application is *context-aware* if it possesses this quality, and we define *context-aware computing* as the set of computational techniques designed to obtain and use context.

Using the journalistic framework described above, we can describe one objective of context-aware computing as the building up of a description of a particular situation or context. As an ultimate goal, this description should be indistinguishable from the description that a perceptive (human) journalist would compose. Thus, the process can be roughly framed as 1) acquiring information from sensors and other sources and 2) translating this information into prose (or ordinary spoken language).

Just as a person can have different levels of awareness, a device or application can have different levels of context awareness. To a certain extent, nearly every mobile device is context-aware in the sense that it knows certain details about its state (e.g., battery level, networks it is connected to, and whether or not a phone call is taking place with it) and perhaps about its user or owner (name, phone number, etc). We arbitrarily distinguish, however, between context-aware and non-context-aware devices based on the amount and breadth of contextual information that the device can obtain and whether it possesses sensors and software that are employed specifically for the purpose of obtaining contextual information. Admittedly, this is a blurry definition, and this type of classification is often a matter of opinion.

Don't be surprised if some of this starts to sound a bit "sci-fi." The goal of creating a context-aware device is related to one of the primary prerequisites of artificial intelligence (AI). A machine exhibiting AI must have a general representation of the world, especially concerning the objects and beings in its immediate surroundings [11]. We shall see, however, that a device can achieve some elements of context awareness without even coming close to achieving AI. Hence, the two subjects are related but distinct.

Last, in order to aid the reader's understanding, we present briefly an example of a real-world, present-day context awareness product—Google Now. The web site for this product provides a succinct description of its context awareness capabilities [12]:

Google Now gets you just the right information at just the right time.

It tells you today's weather before you start your day, how much traffic to expect before you leave for work, when the next train will arrive as you're standing on the platform, or your favorite team's score while they're playing. And the best part? All of this happens automatically. Cards appear throughout the day at the moment you need them.

Of course, none of these capabilities would be possible without a source of contextual information, such as the user's position, preferences, or activity (e.g., leaving for work). It is precisely the acquisition of this type of information that we will explore further in this chapter; we will learn how to process it in Chapter 9.

8.1.2.1 Is Location Awareness Part of Context Awareness?

As stated above, location awareness is one element of context awareness, thus we could claim that any device that is location-aware is therefore, to a certain extent, context-aware. In fact, location awareness can be viewed as an important pioneer and forerunner in context awareness. There is no doubt that location is one of the most important elements of context, hence the focus of much of this book on positioning methods. This is one aspect of the "geospatial era," discussed in Chapter 1, where many forms of computing are adopting location as a central contextual element. The most recent advances in mobile technology, however, point to a future where mobile devices will be aware of much more than just location. In fact, positioning technologies are becoming so ubiquitous that a location-aware device may not seem noteworthy enough to merit the special distinction of being context-aware. Indeed, our view is that location awareness is a necessary precursor to context awareness (at least for the vast majority of applications) but does not in itself imply context awareness.

8.1.2.2 Relation Between Context Awareness and Situation Awareness

Some readers may note that a related term *situation awareness* (or *situational awareness*) is commonly used in certain disciplines. The distinction between context awareness and situation awareness is largely of a historical nature rather than conceptual. The former comes from mobile computing researchers, whereas the latter is rooted in military aviation.

Originally, situation awareness referred to the pilot's "awareness of conditions and threats in the immediate surroundings" [13], and its importance was known as early as World War I, especially by German flying ace Oswald Boelke [14]. Eventually situation awareness was also used to refer to machine awareness of such conditions and threats. Furthermore, the term became popular in a plethora of fields outside of aviation applications, including maritime[18] or ground-based operations, and even outside of military applications [15]. Hence, today the concepts are essentially the same, and the two terms can be used almost interchangeably. Because context awareness is more commonly used in mobile computing, whereas situation awareness is more commonly used in military, safety, and security applications, we will primarily use the former in this book.

8.2 WHY IS CONTEXT AWARENESS IMPORTANT?

Now that we have a basic understanding of the concept "context awareness," it is worth asking the question, "Why is context awareness important?" Given its

[18] Also known as maritime domain awareness.

prominence as a topic in computer science research, there should be compelling reasons for why it is studied.

As Anind Dey, one of the pioneers of context awareness, and others have pointed out, the goal of creating context-aware devices is largely about improving the richness and ease of human-computer interaction [16]. If a computer can be made more aware of the conditions and environment in which it and its users reside, then there will be less need for the users to explicitly tell the computer what it is they need the computer to do. To a certain degree, the computer will "know" what our needs are. This is because developers will have prescribed what is typically needed in a given situation into an application designed to serve such needs (or allowed the user to specify his or her needs in given situations).

A simple example could occur when a mobile device detects that you will be late for a meeting (because you are 30 minutes by car from the meeting place with only 15 minutes until the meeting starts), and could automatically send a message to a contact saved in your mobile device and identified in your mobile calendar as the organizer of the meeting. Therefore, to a large extent, context awareness is about increasing the automation capabilities of our computing devices, or *minimizing* the amount of human-computer interaction that is required to perform a given task.

Many other examples of such context-enabled capabilities could be listed here, but like many worthy scientific and technological endeavors, it is impossible to say what the most important reason for pursuing context awareness is. It is a bit like speculating in the early 1960s about the importance of the Internet (or Intergalactic Computer Network, as it was known in an early phase) that we see so clearly from our vantage point in the twenty-first century. Certainly at that time computer scientists could see the value of networking together four to eight mainframe computers to share resources and transfer data, but they could hardly envision at that time how such entities as Google, Facebook, Twitter, and Skype would change the way that billions of people operate on a daily basis; how businesses would operate; and even how revolutions would be organized! Similarly, although we can see some of the immediate promises of context awareness for mobile computing, we must rely on our intuition and curiosity to believe that making computers aware of context is an effort that will pay off in countless, unimaginable ways.

8.3 HISTORY OF CONTEXT-AWARE COMPUTING

It is worthwhile to review the history of context awareness from the literature and see how the concept has developed and been employed. Reviewing this history is challenging, however, due to the wide and varying use of the concept "context," as discussed above. Some argue that the topic of context can be traced back to

Frege [18, 19] or Russell [20], where the role of context in determining the meaning of symbols and words is discussed [17–20].[19] These philosophical issues regarding the word "context" are not, however, essential for a practical understanding of context awareness, so we will only mention them in passing during this brief review.

In a similar vein, we can find studies from the 1950s and 1960s, where context is used in various applications, such as machine recognition of handwriting, printed text, and verbal speech [21–23]. These works, however, are primarily concerned with discourse context rather than conditions context. Discourse context is fundamentally different from conditions context because in the former there is (by definition) a human in the loop, whereas representations of the latter could be generated automatically.[20] As we stated at the onset, we are interested in automatic methods; hence these studies are only of peripheral interest.

In some sense, however, discourse (i.e., natural language) is relevant to context awareness because it can be used to infer conditions context. Thus, *natural language processing* can help provide inputs to context-aware applications. A full survey of natural language processing techniques is beyond the scope of this book, so the reader is referred to Porzel's extensive coverage of the topic in [24].

As mentioned above, the subject of context awareness exhibits strong ties to the field of AI. As early as 1963, one of the "fathers of AI," John McCarthy, described the concept of a *fluent* (later called *situational fluent*), which we might also call a *contextual proposition* or *contextual statement*. A fluent is expressed as a function of a *situation*, which is described as "the complete state of affairs at some instant of time." One simple example of representing a fluent that McCarthy provides is `raining(s)`, which indicates that it is raining in situation `s`. From this example, we can clearly see the parallel between a set of *fluents* for a particular situation and a context [25].

It is curious to note, however, that McCarthy does not explicitly use the term context until a 1987 paper titled "Generality in Artificial Intelligence" [26]. Even then, he does not provide an explicit definition of context nor comment on its relation to the concepts of fluent or situation. Nonetheless, this paper seems to be the spark that set off an intense effort to study and even formalize the concept of context. By the early 1990s, we can find important works on formalizing context and on contextual reasoning [1, 2, 27, 28]. Although this literature does not

[19] Russell does not explicitly use the term "context," but one can recognize its implicit presence.

[20] Computers, of course, can simulate human discourse and can even generate discourse about the conditions context. Still, we feel justified in making this distinction because it seems that a computer capable of human-like discourse requires first some knowledge of at least some elements of conditions context. Hence, computer-generated discourse would constitute a particular use of conditions context, rather than a method of acquiring conditions context.

directly herald an era of context-aware devices, it constitutes important theoretical groundwork in the development of context awareness as a research topic.

Another important precursor to context awareness can be seen in the often cited article "The Computer for the 21st Century," published in *Scientific American* [29]. In this highly readable article, Weiser envisions a future state of technology where context awareness plays a crucial (albeit unspoken) role:

Sal awakens; she smells coffee. A few minutes ago her alarm clock, alerted by her restless rolling before waking, had quietly asked, "Coffee?" and she had mumbled, "Yes."

This fictional account goes on to describe a number of automated and assisted functions that are performed by *ubiquitous computing* devices, which in many cases can be considered to be context-aware. When this article was published in 1991 it probably sounded like mostly science fiction. With the recent advent of wireless devices like Fitbit (which can monitor sleep periods), such scenarios are now on the verge of reality.

For the first reported implementation of a context-aware device, we look to the Active Badge Location System, developed at the Olivetti Research Laboratory in Cambridge, England, in the early 1990s [30]. Not surprisingly, this system was focused exclusively on creating *location awareness*, especially for employees in an office environment. A few years later, Schilit and Theimer were the first to explicitly use the term context-aware computing, which they defined as "the ability of a mobile user's applications to discover and react to changes in the environment they are situated in" [31].

By the mid 1990s, the study and development of "context-aware applications" becomes an active area (e.g., PARCTAB, stick-e notes [32], CyberGuide [33], CyberDesk [34]). The reader can find an excellent snapshot of the turn-of-the-century, state-of-the-art in context-aware computing in a special issue of the journal *Human-Computer Interaction*, published in 2001 [35]. What is clear from the literature of this period is that context awareness had become a strong research focus for several research groups, typified by prototype and architecture development. It is also clear, however, that an abundance of challenges still remained to achieving rich and robust context awareness.

During the past 10 years, context awareness research has continued to press forward, and it has begun to work its way into several commercial products. Several frameworks for context awareness have been developed by researchers. One challenge that remains, however, is that no widespread standard yet exists for representing contextual information.

As far as commercial products, the example of Google Now was given above, but there are plenty of other examples as well. Dexetra currently offers an application called "friday," which keeps track of a user's daily activities, such as commuting, activity on social networks, and various uses of the user's phone (calls, photo-taking, etc.), as well as the location where these features were used.

Qualcomm Retail Solutions offers a context awareness platform, called Gimbal, with SDKs available for iOS and Android. Already in 2012, Samsung's Galaxy S3 smartphone was marketed as a context-aware device with its ability to recognize when a face is looking at the screen and control whether the screen stays on base on this information. The Motorol Moto X, released in 2013, also exhibits many context-aware features. These include sensing when the phone is picked up (in order to turn on the screen and display notifications) and sensing certain hand gestures (in order to activate the camera application). An application called Motorola Assist[21] uses context to detect activities like driving and sleeping. It allows users to predetermine how to handle or respond to incoming text messages and phone calls, based on the current activity (e.g., reading text messages to the user while driving).

8.4 EXAMINING CONTEXT IN DETAIL

The remainder of this chapter examines in more detail the various elements of context first introduced in Section 8.1.1. Some of these elements are more straightforward to detect in a mobile device than others. For the more challenging elements, we will provide only precursory coverage in this section.

8.4.1 What: The Activity Context

The first element of context is a high-level description of what activity is taking place or what the overall current situation is.[22] Examples could include: driving a car, waiting at a bus stop, having lunch, and participating in a meeting. In practice, there can be many possible ways to formulate such high-level activity or situation descriptions, as well as many different techniques for classifying and sensing them. The overall objective of classifying and sensing activities is often called *activity recognition*, and a great deal of research has been conducted in this area alone.

The optimal set of techniques to detect high-level context is specific to the set of sensors or other data available, the scope of activities that one desires to detect, as well as the specific requirements of the application (such as whether or not the context must be detected in real time). Therefore, it is very difficult to provide general guidance on how to perform activity recognition. In broad terms, however, we can use one or more techniques from the fields of *machine learning* and

[21] At the time of this writing, Motorola Assist was compatible with four different Motorola smartphone models, including the Moto X.

[22] We recognize that "situation" is essentially a synonym for context, but it is used here to embody the highest level of contextual information that functions as a summary of the context. We also note that a context may sometimes be characterized by a great deal of *inactivity*, but even "sitting idly" can be one classification within a set of activity classifications.

pattern recognition. Specifically, the following signal processing techniques have been used for activity recognition:

- Decision tree;
- Hidden Markov model;
- Naïve Bayes classifier;
- Support vector machine;
- Kalman filter;
- Particle filter.

We will provide further details on some of these techniques in Chapter 9. Machine learning and pattern recognition are fields with a rich set of literature, and we can only scratch the surface in this book. Readers wishing to go deeper are encouraged to explore especially the references cited in Chapter 9.

8.4.2 Who: The User and Social Context

The second element of context is concerned with 1) who the user of the mobile device is and, possibly, 2) the social context in which the user is situated. Acquiring the identity of the mobile user is usually dependent on information the user has stored in the mobile device and/or on privacy permissions that the user has set to control access to various account information. For example, on Android devices a user can create a local profile (see code example below), or personal information can be imported via an account, such as Gmail, Facebook, or Twitter. Most major web services offer an SDK for Android that allows privacy-controlled access to profile information (e.g., Facebook SDK for Android [36]).

```
public HashMap<String, String> getProfileInfo() {
   HashMap<String,String> profile = new HashMap<String,String>();
   ContentResolver cr = getContentResolver();
   Cursor cursor = cr.query(
      // Specify the URI of the "Profile" contact.
      Uri.withAppendedPath(
      ContactsContract.Profile.CONTENT_URI,
      ContactsContract.Contacts.Data.CONTENT_DIRECTORY),
      Query.PROJECTION, null, null, null);
   cursor.moveToFirst();

   // Loop through each cursor row, appending key/value to HashMap
   while (!cursor.isAfterLast()) {
      String key = null;
      // Determine key for HashMap based on MIMETYPE column in
      // cursor row. Example shown for name field. Other fields
      // left for reader to implement as desired.
      if (cursor.getString(Query.MIMETYPE).contains("/name")){
            key = "email";
         }
         if (key != null){
            profile.put(key, cursor.getString(Query.DATA));
         }
         cursor.moveToNext();
      }
      return profile;
}
```

The amount and type of information that can be obtained via these methods varies greatly based on the particular information source. For each particular user, it also depends on what information that user has entered into his or her profile and what accounts are accessible on the device. Therefore, it is important to implement fallback methods, such as manual prompts, in case required user information is not available via automated methods.

8.4.3 Where: The Location Context

As discussed above, location is an important element of context. Since much of this book is concerned with describing mobile-based techniques to obtain a user's position, we will not repeat such descriptions here. In the Android platform, there are various techniques used to obtain geographic coordinates (i.e., latitude, longitude, and altitude) that hide much of the underlying complexity used to calculate them. Most importantly, the `LocationManager` class provides methods to obtain location (i.e., position) updates from either a GNSS[23] or a network-based provider. Table 8.1 below highlights the most important methods from this class.

It is important to note that location can be expressed in many different forms, such as in geodetic coordinates (see Section 2.1) or semantic descriptions like "at home." From the perspective of context awareness, it is often such semantic descriptions that provide the most readily useful information. In this form, the context is rather self-evident, whereas the contextual meaning of geodetic coordinates (e.g., latitude and longitude) must be interpreted.

This interpretation could be relatively straightforward, such as querying a database of points of interest (recall this topic from Chapter 7), or it could be complex, such as analyzing the location history of a particular user to infer the relevance of the location. In either case, it is clear that the relevant information is not the geodetic coordinates themselves but rather the significance of the location which the coordinates reference.

[23] As noted in Chapter 1, GNSS is the preferred term when referring to satellite-based navigation systems in general. In Android, there are classes and methods using the acronym GPS when most modern mobile devices have multi-GNSS enabled receivers.

There is one other location-related topic that has not been covered elsewhere in this book, that of *microlocation sensing*. This refers to the use of proximity-based positioning (recall Section 2.2.6) to detect that a mobile device is within a very small-scale region (e.g., centimeter and meter scale), such as near a user's desk or near a particular display in a store. This concept has also been referred to as "hyper local." As the size of the region becomes smaller, oftentimes the context becomes clearer. For example, if a mobile device has localized itself as lying on the nightstand next to the user's bed, then the context is likely related to sleep. Therefore, this topic is of particular relevance to context awareness.

Table 8.1

List of Useful Methods from `android.location.LocationManager` Class

Property	*Description*
addGpsStatusListener	Adds a GNSS status listener, which is powerful way to get low-level updates from the GNSS receiver.
addNmeaListener	Adds an NMEA status listener, which can receive NMEA sentences as they are output from the GNSS receiver.
addProximityAlert	Adds an event listener that is triggered when the device enters or exits a given circular region (supplied as latitude/longitude/radius).
getAllProviders	Returns a list of all available location providers.
getGpsStatus	Returns a GpsStatus object, containing the current state of the GNSS engine. Used in conjunction with `addGpsStatusListener`.
getLastKnownLocation	Returns a location object, containing the last known location from a specified provider (i.e., GNSS or network).
requestLocationUpdates	Used to set up events, which are triggered when a location update is available from the specified provider (i.e., GNSS or network).

There are several technologies related to microlocation sensing. For example, in late 2013 Apple starting using devices called iBeacons in its stores to detect at a fine scale where customers are located. These beacons use Bluetooth low-energy RF signals to detect when mobile devices are nearby and communicate with them accordingly. Qualcomm also offers devices called Gimbal Proximity Beacons, which operate in a similar fashion but are designed to interact with the Gimbal platform (mentioned in Section 8.3). Bluetooth low-energy is not the only signal that is used for microlocation sensing. Samsung also sells an add-on product for its mobile devices, called TecTiles, which are programmable NFC stickers. They can be programmed to interact with a mobile device whenever it is within NFC range (a few centimeters), allowing for extremely fine-scale localization. Imagine the potential applications if such NFC were deployed more widely. NFC tags could be integrated into such items as pockets in clothing and in handbags, laptops and desktop computers (e.g., for identification purposes), and cashier or payment systems. The possibilities are endless!

8.4.4 When: The Time and Date Context

The next element of context is trivial to obtain from a mobile device's OS. For example, in Android you can get the device's system time with two lines of code:

```
SimpleDateFormat sdf = new SimpleDateFormat(
   "MM/dd/yyyy HH:mm:ss");
String dateTime = sdf.format(System.currentTimeMillis());
```

A string representing the date and time (in local time zone) will be stored in the variable `dateTime` for use in your application. The difficult part can come in figuring out how to use this information.

A date and time coordinate must be interpreted in much the same way a location coordinate is interpreted. For example, 6:00 a.m. for one person might represent a normal weekday wake-up time, whereas for another person it might represent a precious "Do not disturb" sleep time. Similarly, 5:00 p.m. on December 21 might represent an entirely different context than 5:00 p.m. on June 21 (especially at higher latitudes). Last, significant dates, such as holidays and personal events (birthdays, anniversaries, etc.), must be inferred using some additional information source. These difficulties make inferring the context from a raw time and date measurement a nontrivial matter.

There are various techniques available for inferring the contextual relevance of time and date information. The most obvious choice is by utilizing the built-in calendar application with which most (if not all) mobile devices are equipped. The drawback of this approach is that it relies on information supplied by the user, which may not be complete or up-to-date. In most cases, however, it is the easiest of the available options to implement.

An alternative approach would be to analyze the activity history of a user to infer the likelihood that he or she would perform a certain activity, such as commuting to or from work, during certain times, days of the week, or dates of the calendar year. This approach, however, may not yield detailed enough information for some applications, and it is considerably harder to implement.

8.4.5 Why: The Motivational Context

The fifth element of context—the *motivational context*—is perhaps the most elusive of them all. It involves analyzing one or more other contextual elements in order to infer a user's intentions. The techniques used to perform such inferences are largely dependent on the specific contextual domain or application where the inferences will be utilized. We can already see several minor examples of motivational context being applied in commercial devices, although clearly we are in the early stages of development in this regard. For example, Motorola's Moto X smartphone uses its rich set of sensors and specialized processors, as well as the features of Google Now, to infer the needs and desires of its users. This includes sensing when the phone is picked up (in order to turn on the screen and display notifications) and sensing certain hand gestures (in order to activate the camera application).

In general, techniques from machine learning and pattern recognition could be used. It is clear, however, that further research and development is needed before motivational context can be inferred and utilized on a large scale. To build up accurate models of human behavior and human intent, it may be necessary to generate large amounts of training data, perhaps using a crowd-sourcing approach. This topic will be considered further in Chapter 10.

8.4.6 In What Manner: Motion Context and Other Details of Context

The sixth element of context can rightly be judged as somewhat of a catch-all of other contextual information. It is formed from the sixth component of Hermagoras's *circumstance* and can be used to groups together additional details of a situation that do not fit into the other elements. For example, if the answer to *what* is the activity "dancing," then *in what manner* might include such concerns as the type of dance and the tempo of the dance. In particular, motion attributes fit nicely into this category. For example, one can use motion recognition techniques to detect the particular mode of motion and to describe other motion attributes such as speed, heading, and acceleration.

8.4.7 By What Means: The Context-Aware Device and the Methods of Sensing Context

The last element of context includes descriptions of the context-aware device itself, as well as the methods employed to sense the context. Given the wide range of mobile devices available today (recall Chapter 1), it is important for an application to be aware of details about the device it is operating on. This will supply the application with information about the capabilities of the device and about what other contextual information might be available to it. Some details about the state of the device can even constitute a situation or context themselves (e.g., low battery context).

Basic information about the device for use in an application, such as device name, manufacturer, model, operating system, and build version, is generally available via the OS's APIs. For example, the Android APIs make available a set of system properties in the class `android.os.Build`. See Table 8.2 for a list of some of these properties, which may be useful for describing the mobile device.

In addition, it is generally straightforward to obtain contextual information about the data communication functionalities of the mobile device, such as whether or not a call is in progress and what cellular network or WLAN the mobile device is connected to. In Android, one can use the `TelephonyManager` class to obtain information related to the cellular radio. Examples of useful methods available from this class are shown in Table 8.3. Similar information about the WLAN radio of an Android device can be obtained using the `WLANManager` class. Alternatively, if one is interested in data connectivity regardless of whether it is through mobile networks, WLAN, Bluetooth, or other means, the `ConnectivityManager` provides basic information about the connected or available data networks.

Table 8.2

List of Useful System Properties (Fields) from `android.os.Build` Class

Property	*Description*
Device	Codename given to the device (e.g., maguro)
Brand	Brand-customized version of the operating system (e.g., Verizon)
CPU	The name of the instruction set (CPU type + ABI convention) of native code (e.g., armeabi-v7a)
Display	An ID representing the build version of the operating system (e.g., JRN84D)
Manufacturer	Manufacturer of the device (e.g., Samsung)
Model	Model name of the device (e.g., Galaxy Nexus)
Product	Codename given to the firmware version of device (e.g., takju)
Radio	Version of the cellular radio firmware (e.g., I9250XXKK6)
Serial	Serial number of the device (e.g., 014682070502301D)
Version.release	Version number of the device's operating system (e.g., 4.1)

Other information about the device's current state or condition, such as the current battery state, amount of memory available, and current usage of the CPU, is often also available. For example, in Android, detailed information about the battery state can be obtained from the BatteryManager class. For memory usage, use the ActivityManager. For CPU usage, there is no specialized method for obtaining overall usage, but since Android is based on Linux, one can use the `/proc/stat` file to obtain the desired information.

In addition to these classes and methods, Android offers a powerful technique to make applications aware of various actions or events triggered on the device, known as *intent messaging*. Intents are essentially intradevice messages, and by setting up *intent filters* in an application, the app can receive these messages, containing various types of information depending on the type of intent. A small sampling of intents that are triggered automatically by different system events includes: answering a phone call, pressing the "call" button, unlocking the keyguard, starting the camera application, plugging in a headset, switching airplane mode on or off, docking into a desktop or car dock, installing or removing an application, rebooting or shutting down the device, low battery warning, and low storage space warning.

Table 8.3

List of Useful Methods from `android.telephony.TelephonyManager` Class

Method	*Description*
getCallState()	Returns an integer representing the call state of device (e.g., CALL_STATE_RINGING = 1)
getDataState()	Returns an integer representing the data state of device (e.g., DATA_CONNECTED = 2)
getDataActivity ()	Returns an integer representing the current data activity of the device (e.g., DATA_ACTIVITY_OUT = 2)
getDeviceId()	Returns an integer that uniquely identifies the device (IMEI, MEID, or ESN) (e.g., 352563176148781)
getNetworkOperatorName()	Name of the cellular network operator (e.g., FI SONERA)
getNetworkType()	Returns an integer representing the type of cellular network currently in use (e.g., NETWORK_TYPE_UMTS = 3)
getPhoneType()	Returns an integer representing the type of radio used to transmit voice calls (e.g., PHONE_TYPE_GSM = 1)
getSimSerialNumber()	Serial number of the SIM card (e.g., 8893579030109339463)
getSubscriberId()	Unique subscriber ID for mobile customer (e.g., IMSI)
isNetworkRoaming()	Returns a Boolean value depending on network roaming state

Finally, custom intents can also be implemented, which in effect allows any software component in the device to be aware of any action of any other component, provided the appropriate intents and intent filters are specified. Indeed, intent messaging provides a powerful means to implement various types of context awareness in Android devices.

8.5 HOW TO USE CONTEXT

Now that we've seen an overview of the various elements of context, we will look at how context can actually be utilized in context-aware applications. In order to demonstrate this, we return to the fictional scenario outlined in Section 8.1.1, where Mary and her colleagues are waiting for their colleague Steve to arrive, in order to leave together for lunch.

It turns out that the colleagues had gathered for lunch after responding to an "*AreUin?*" request from Mary, a mobile app that they frequently use in their department to see who is interested in joining for lunch or for after-work social outings. Steve had answered the request with "*I'm in*," so it is a bit odd that he hadn't shown up in the lobby yet.

Timo says to Mary, “Did you try calling Steve?”

“Yes, but he didn’t answer. *Locator* says that he’s in the lab. It could be though that he left his phone there because it shows as completely static for the past hour and connected to a PC. I’ll run and check.”

Meanwhile, Timo starts the “*RestaurantFinder*” app on his phone to look for possible restaurants. The app sorts nearby restaurants according to the preferences of those who are joining for lunch, whose identities have been sent from the “*AreUin?*” app. *RestaurantFinder* shows at the top of the list that a new Mexican restaurant opened up this week—only a 10-minute drive from their work—and that Mexican food ranks highly among the group’s preferences.

“There’s a new Mexican place that opened up on Turuntie. You guys interested?” asks Timo.

“Sure,” says Anindya, as the others also nod approvingly. At that moment, Mary returns alongside Steve.

“Sorry guys, I had my headphones on and was working at the soldering table. Lost track of time,” explains Steve.

“No worries, Steve. But you do know that AreUIn has a built-in reminder feature, right?” replies Timo with a wink of the eye.

“Really? I didn’t know.”

“Yeah, it can even connect to your smartwatch, in case you don’t have your phone on you. So you can rock out to Zeppelin and forget about us…until it vibrates as a reminder,” jokes Timo. “Anyways, it’s good Mary found you because there’s a new Mexican place nearby called Pancho’s, and I already checked to see if there’s a table open. Not only was the answer yes, but they just sent me a coupon. 20% off for groups of five or more! That sound OK to you, Mary?”

“Sounds great. I have a big car, so I’ll drive. Can you send me the address?”

“Done. Ok, let’s go!”

As our characters walk to the parking lot, Mary’s smartphone has already started a navigation application to show the way to “Pancho’s.” Let’s turn away for a moment and take a look at how context has been utilized in this story.

First, Steve’s position—more precisely the position of Steve’s phone—was determined by the locator app. Furthermore, Mary had some contextual information that Steve’s phone had been idle for some time and connected to a

PC. This is an example of how motion patterns and information about the phone's state can be used to provide additional details beyond mere position.

In addition, this example illustrates an important point; when the smartphone is the sole sensing platform, the inference of *user context* is dependent on the smartphone being in close proximity to the user. If Steve had simply left his phone in the lab and was in another location, Mary would have been out of luck. This is one of the reasons why context awareness may shift in the future toward wearable devices, such as smartwatches. Such wearable devices can still communicate and cooperate with other mobile devices, but in addition they have considerable sensing and networking capabilities of their own. This technology is still in its infancy, but already nearly a dozen manufacturers are selling various incarnations of this idea. Wearable devices and sensors will be discussed further in Chapter 10.

Next, Timo used the *social context*, namely the identities supplied by the "*AreUIn?*" app and their preferences stored by the *RestaurantFinder* app, to select a suitable restaurant. *RestaurantFinder* also used the approximate position of Timo's phone to filter the choices, which is a typical feature in LBSs.

Last, Timo performed a context-dependent query to determine if the restaurant currently had a table available for five people. The restaurant replied in the affirmative and also appears to have engaged in what we might call *context-sensitive marketing*: based on the busyness of the newly opened restaurant and the opportunity to bring in five potentially regular customers, it offered a discount coupon. Let's return to the story to see if context was further employed during this lunch outing.

After their meals at Pancho's were served, the jovial colleagues got into a lively discussion about soccer. "Barcelona is going all the way this year. Messi is healthy now and playing incredibly well," argues Steve.

"They will go far, but my pick is Bayern Munich. The team is just as strong as last year's, and Pep Guardiola is doing a great job as manager," counters Timo.

"Hey guys, look." Mary holds up her phone to show a notification. The notification shows that the monthly department meeting starts in 15 minutes.

"Oh, man. I totally forgot. It's the third Friday of the month, isn't it?" notes Anindya.

"Yep. I'm glad I put it in my calendar. Let's get going. Waiter, check please!"

Here we see a somewhat more innovative use of context. Mary's phone had detected that she was located several kilometers from her workplace, in fact, a 10 minute drive. Since her calendar showed a meeting starting at 1:30 p.m., and it was already 1:15 p.m., it displayed a high-priority notification. Apparently the others had either not put this meeting into their calendars or had not set up such notifications. Luckily for them, Mary had done both, so they avoid being late this time.

This fictional example provides an example of how context can be utilized in context-aware devices, namely smart mobile devices. From our introductory examination of context awareness in this chapter, it should now be fairly straightforward to envision how these features could be implemented. Admittedly, this example is not as "futuristic" and impressive as the *ubiquitous computing* scenario that Weiser depicted in 1991. Our intention was to demonstrate the concepts in this chapter and to provide a bridge to more advanced context awareness capabilities researchers and developers can implement with the current generation of mobile devices.

This example also demonstrates that context awareness, at least at the present level of technology, does not completely replace human intelligence and initiative. It can, however, go a long way in unburdening humans from various tasks, both mental and physical (in a human-computer interaction sense), freeing our human capacities for other purposes. Thus, the role of the context awareness researcher and developer should be now clearer—we must minimize the burden required to perform various tasks and open up new possibilities that would not be possible without context awareness.

8.6 SUMMARY

This chapter introduced the concept of *context* in terms of its importance to mobile geospatial computing. Since context is notoriously difficult to define, we have presented an intuitive framework for understanding the various elements of context, centered around seven questions: *what, who, where, when, why, in what manner,* and *by what means*. This in turn was used to explore the concept of *context awareness*, which we defined broadly as the quality of having knowledge of context. We briefly examined context awareness from a historical perspective, where we could see it blossoming as a research topic in the 1990s.

Next, we examined each of the elements of context in more detail, providing some examples of how contextual information within particular elements can be obtained. For several of these elements, including *what*, *why*, and *in what manner*, the appropriate methods vary greatly depending on the desired contextual information. In addition, these are often the most challenging elements to implement. For these reasons, further coverage of these topics is reserved for Chapter 9 on *contextual reasoning.*

Last, we used a fictional story, depicting a common workspace scenario in order to demonstrate how context awareness can be used in mobile devices. This story also highlighted several new concepts, such as *social context* and *context-sensitive marketing*, which are increasingly important in modern mobile devices.

References

[1] McCarthy, J., "Notes on formalizing context," *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 555-562, San Mateo, California: Morgan Kaufmann, 1993. [Online]. Available: http://bit.ly/OEHAyO.

[2] Akman, V., and M. Surav, "Steps toward formalizing context," *AI Magazine*, Vol. 17, No. 3, 1996, pp. 55-72, available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.3474.

[3] Foltz, P. W., W. Kintsch, and T. K. Landauer, "The measurement of textual coherence with latent semantic analysis," *Discourse Processes*, Vol. 25, No. 2 & 3, pp. 285-307, 1998.

[4] Bazire, M., and P. Brézillon, "Understanding context before using it," *Modeling and Using Context: Lecture Notes in Computer Science*, pp. 113–192, A. Dey, B. Kokinov, D. Leake, and R. Turner (eds.), Berlin, Heidelberg: Springer Berlin/Heidelberg, 2005, available: http://dx.doi.org/10.1007/11508373_3.

[5] "Context," In *Merriam-Webster.com*, 2012. [Online]. Available: http://www.merriam-webster.com/dictionary/context.

[6] "Five Ws," In *Wikipedia.com*, 2012. [Online]. Available: http://en.wikipedia.org/wiki/Five_Ws.

[7] Bennett, B. S., "Hermagoras of Temnos," In *Classical Rhetorics And Rhetoricians: Critical Studies And Sources*, pp. 187-193, M. Ballif and M. G. Moran (eds.), Westport, CT: Praeger Publishers, 2005.

[8] Welch, J. R., and K. G. Salter, "A context algorithm for pattern recognition and image interpretation," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-1 , No. 1, 1971, pp. 24-30.

[9] Toussaint, G. T., "The use of context in pattern recognition," *Pattern Recognition*, Vol. 10, 1978, pp. 189-204.

[10] Swain, P. H., S. B. Vardeman, and J. C. Tilton, "Contextual Classification of Multispectral Image Data," NASA Technical Report SR-PO-00443, 1980.

[11] McCarthy, J., and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," *Machine Intelligence*, Vol. 4, 1969, pp. 463-502.

[12] Google, Inc., Google Now web site, [Online]. Available: http://www.google.com/landing/now/.

[13] Morishige, R. I., and J. Retelle, "Air combat and artificial intelligence," *Air Force Magazine*, October 1985, pp. 91-93.

[14] Endsley, M. R., "Design and evaluation for situation awareness enhancement," *Proceedings of the Human Factors Society—32nd Annual Meeting*, 1988, pp. 97-101.

[15] Gilson, R. D., "Special Issue Preface," *Human Factors* (Special Issue on Situation Awareness), Vol. 37, No. 1, 1995, pp. 3-4.

[16] Dey, A. K., "Context-Aware Computing," In *Ubiquitous Computing Fundamentals*, Boca Raton, FL: CRC Press, 2010, pp. 321-352.

[17] Brézillon, P., "Context in human-machine problem solving: A survey," *The Knowledge Engineering Review*, Vol. 14, No. 1, 1999, pp. 1-34.

[18] Frege, G., *The Foundations of Arithmetic, a Logico-Mathematical Enquiry into the Concept of Number,* transl. J. L. Austin, Oxford: Basil Blackwell, 1974. Originally published in German as *Die Grundlagen der Arithmetik, eine logisch mathematische Untersuchung über den Begriff der Zahl,* Breslau: Verlag von Wilhem Koebner, 1894.

[19] Frege, G., “On sense and reference,” *The Philosophical Review*, Vol. 57, No. 3, 1948, pp. 209-230. Originally published in German as “Über Sinn und Bedeutung,” *Zeitschrift für Philosophie und philosophische Kritik*, Vol. 100, 1892, pp. 25-50.

[20] Russell, B., “On denoting,” *Mind*, New Series, Vol. 14, No. 56, 1905, pp. 479-493.

[21] Bledsoe, W. W., and I. Browning, “Pattern recognition and reading by machine,” *1959 Proceedings of the Eastern Joint Computer Conference*, 1959, pp. 225-232.

[22] Miller, G. A., G. A. Heise, and W. Lichten, “The intelligibility of speech as a function of the context of the test materials,” *Journal of Experimental Psychology*, Vol. 41, No. 5, 1951, pp. 329-335.

[23] Duda, R. O., and P. E. Hart “Experiments in the recognition of hand-printed text: Part II—Context analysis,” 1968 *Proceedings of the Fall Joint Computer Conference*, 1968, pp. 1139-1149.

[24] Porzel, R., *Contextual Computing: Models and Applications*, Heidelberg: Springer, 2011.

[25] McCarthy, J., “Situations, Actions, and Causal Laws,” Stanford Artificial Intelligence Project Memo No. 2, 1963.

[26] McCarthy, J., “Generality in artificial intelligence,” *Communications of the ACM*, Vol. 30, No. 12, 1987, pp. 1030-1035.

[27] Giunchiglia, F. “Contextual reasoning,” *Epistemologia, Special Issue on “I Linguaggi e le Macchine*,” Vol. XVI, pp. 345-364, 1993.

[28] Giunchiglia, F., and P. Bouquet “Introduction to contextual reasoning,” *Perspectives on Cognitive Science*, Vol. 3, 1997, pp. 138-159.

[29] Weiser, M., “The computer for the 21st century,” *Scientific American*, Vol. 265, No. 3, 1991, pp. 94-104.

[30] Want, R., et al., “The active badge location system,” *ACM Transactions on Information Systems*, Vol. 10, No.1, 1992, pp. 91-102.

[31] Schilit, B. N., and M. M. Theimer, “Disseminating active map information to mobile hosts,” *IEEE Network*, Vol. 8, No. 5, 1994, pp. 22-32.

[32] Brown, P. J., “The stick-e document: a framework for creating context-aware applications,” *Electronic Publishing*, Vol. 8, No. 2/3, 1995.

[33] Abowd, G. D., et al., “Cyberguide: a mobile context-aware tour guide,” *Wireless Networks.*, Vol. 3, No. 5, 1997, pp. 421-433.

[34] Dey, A. K., G. D. Abowd, and A. Wood, “CyberDesk: a framework for providing self-integrating context-aware services,” *Knowledge-Based Systems*, Vol. 11, 1998, pp. 3–13.

[35] Moran, T. P., and P. Dourish (eds.), “Special Issue on Context-Aware Computing,” *Human-Computer Interaction*, Vol. 16, Nos. 2-3, 2001.

[36] Facebook, Inc., “Android SDK Reference,” [Online]. Available: http://bit.ly/PoIpbX.

PUBLICATION 2

R. Chen, **R. E. Guinness**, “Contextual Reasoning” in *Geospatial computing in mobile devices*. Boston: Artech House, ch. 9, pp. 171-197, 2014

Chapter 9

Contextual Reasoning

In the last chapter we introduced the concepts of context and contextual information. In this chapter, we look more deeply at how smartphones can obtain and process this type of information. There are a number of terms that have been suggested to encompass the idea of processing contextual information, such as contextual thinking [1] and contextual intelligence [2]. We have chosen *contextual reasoning* [3], but all of these terms are more or less equivalent. We will present several examples of the techniques used for contextual reasoning, though we will not attempt to be exhaustive in our coverage. To aid those wishing to go deeper in this topic, we will point out ample references where additional information can be found.

9.1 WHAT IS CONTEXTUAL REASONING?

We begin with a formal definition. *Contextual reasoning is the process of forming higher level inferences about context from lower level information.* "Higher" and "lower" are relative terms in this definition, where the higher levels are more general and abstract, and the lower levels are more specific details that individually don't provide the "big picture" of a context. In colloquial terms, it is the process of seeing the forest through the individual trees.

Figure 9.1 illustrates this process conceptually. Known as the "context pyramid," this figure divides the contextual reasoning process into six different abstraction levels, although in practice there may be fewer or more levels, depending on the application. The first level represents the raw data received from sensors, whether we speak of hardware sensors like accelerometers and gyroscopes, or "soft sensors" like a GNSS receiver or application code that detects various states of the smartphone (e.g., low battery warning). In the next level of abstraction, physical parameters are derived from the raw sensor data, such as the speed that the user is moving or decibel level recorded by a microphone. In the third level (which may be optional in some applications), various features or patterns may be extracted from the lower levels, such as combining successive parameters to calculate a moving average or variance of the parameters.

The next three levels represent the "heart" of contextual reasoning. We move from the realm of numeric data to more semantic representations of context. In the first such level, simple contextual descriptors are inferred from the lower-level data, such as the motion pattern of the user (e.g., standing and walking), or a semantic representation of the user's position (e.g., at work). In the next level, several simple contextual descriptors may be combined to infer a higher-level context, such as the user's current activity. Finally, all of the available contextual information, including information from external sources, may be combined to a final level, which (if successful) can be described as a *rich context.*[24] Ideally this should be expressed in natural language in a form that approaches prose. This is the final result of asking and answering the questions from the journalistic framework described in Chapter 8.

9.2 A HYPOTHETICAL EXAMPLE

To make these concepts more concrete and to provide a geospatial computing example, consider the following low-level information presented in Table 9.1 from a hypothetical example.

A logical high-level inference that might be made from this information is, "Mary is walking to work." This could probably be considered an activity-level descriptor (level 5) in terms of the context pyramid as shown in Figure 9.1. If additional contextual details were to be added, such as the time of day, weather conditions, or any interesting events or details that occurred in the recent past, then it might rise to the level of rich context. As one can imagine, the exact boundaries between these levels are blurry, and the context pyramid should be considered only as a tool for understanding the process of contextual reasoning.

Note that probably several intermediate steps were required in the process of inferring, "Mary is walking to work." For example, we might have calculated that at Mary's current position, she is 500m away from the FGI and that her heading indicates she is going toward the street that leads to FGI.

Clearly humans make inferences like "Mary is walking to work" all the time. It is one of the main functions of our brain to perform this kind of reasoning. Sometimes there are a few intermediate steps performed in the subconscious, and the inference is performed instantaneously. Other times, there may be many intermediate steps, and the inference might require careful analysis of all the available information. It might even require hypothesizing and probing for additional information to confirm a hypothesis.

[24] In using the adjective "rich," we invoke the concept from AI of a *rich object*, which is defined as "an object which cannot be completely described or represented but about which assertions can be made" [33].

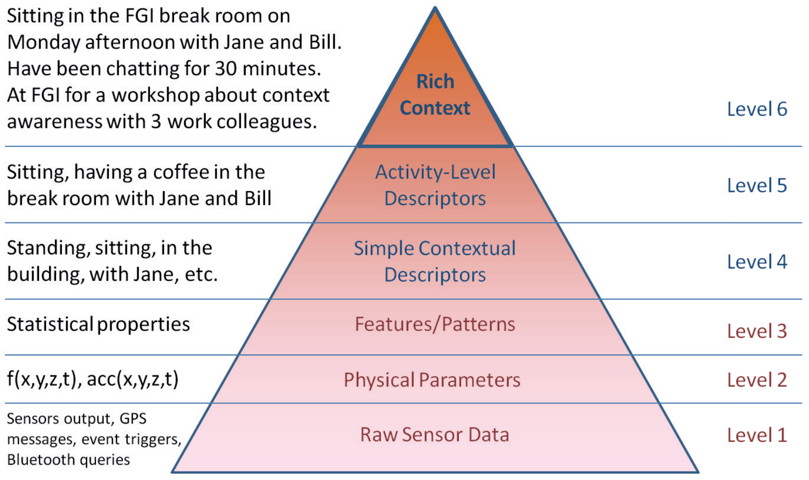

Sitting in the FGI break room on Monday afternoon with Jane and Bill. Have been chatting for 30 minutes. At FGI for a workshop about context awareness with 3 work colleagues.
Rich Context
Level 6
Sitting, having a coffee in the break room with Jane and Bill
Activity-Level Descriptors
Level 5
Standing, sitting, in the building, with Jane, etc.
Simple Contextual Descriptors
Level 4
Statistical properties
Features/Patterns
Level 3
f(x,y,z,t), acc(x,y,z,t)
Physical Parameters
Level 2
Sensors output, GPS messages, event triggers, Bluetooth queries
Raw Sensor Data
Level 1

Figure 9.1 The context pyramid.

Table 9.1

Example of Contextual Information, Its Source, and Corresponding Level

Information	*Source*	*Level*
Mary is located at 60.161 N, 24.542 E.	GPS	Level 2
Mary has a heading of 170 degrees.	GPS	Level 2
Mary is walking.	Motion classifier	Level 4
Mary works at FGI.	Personal profile	External
FGI is located at 60.161 N, 24.546 E.	Significant location DB	External

The same is true for computer-based contextual reasoning. Sometimes there is enough available information and the inference is simple enough to be performed in realtime without any human interaction. In other cases, it can be extremely complex (probabilistic in nature), and require significant processing power. It may even require additional input from the user (i.e., human-computer interaction).

Consider the following additional information related to the above example:

It is 6 p.m. on a Sunday evening.
Mary normally works from 9 a.m.–5 p.m. on weekdays.

From this additional information, a computer-based contextual reasoning program might output that there is a 60% chance Mary is walking to work and a 40% chance she is walking somewhere else. Due to this ambiguity, the program queries the system for additional information:

Mary is walking with her dog.
Mary lives at 60.10 N, 25.00 E.

After this additional information and a few intermediate calculations, the program outputs:

Mary is walking her dog, near her workplace and headed away from her home.

We can see from this example that there are different inferences that can be made from different subsets of low-level information. Furthermore, we see that, in general, the richer the set of information we have to formulate our inferences, the better chance we have of generating a correct one.

Finally, observe that contextual reasoning can also be thought of as a process of *encapsulation* of information. For many applications, it is not interesting to the user nor directly useful for the application that "FGI is located at 60.10 N, 25.10 E" or (worse) that "the standard deviation in Mary's speed for the past 5 seconds is 0.58 m/s." What the application (or user) needs to know is "what is Mary doing," "who is she with," "in what manner is this activity playing out," and similar higher level questions. Thus, contextual reasoning can be alternatively thought of as the process of transforming the multitude of available information into a more concise and useful contextual result, where the definition of "useful" is highly dependent on the desired application.

9.3 WHAT ARE THE METHODS OF CONTEXTUAL REASONING?

The primary tool for making contextual inference is *machine learning*, also known as *statistical learning* or *pattern recognition.*[25] Machine learning is deeply rooted in probability theory, decision theory, and information theory. Unfortunately, we can't cover all of these topics in this chapter, but we refer those readers who are not familiar with these topics to introductory texts on machine learning, where these topics are adequately covered. In truth, it could take years of careful study to deeply understand the state-of-the-art in machine learning and contextual reasoning. Fortunately, however, there are several simple, basic techniques that work well for many applications. This section is intended to give a gentle

[25] Historically, pattern recognition grew out of engineering disciplines, whereas machine learning is the term adopted by most computer scientists [5]. For obvious reasons, statistical learning is the term favored by statisticians. O'Connor humorously noted that at Stanford University there are two almost identical courses offered: "machine learning" by the computer science department and "statistical learning" taught by the statistics department [6].

introduction to some of the common techniques of machine learning that are applied to contextual reasoning.

9.3.1 Introduction to Machine Learning

According to Mitchell et al., "machine learning research seeks to develop computer systems that automatically improve their performance through experience" [4]. In the domain of contextual reasoning, this means that, given the tools of machine learning, the more the system is used, the better it should be able to reason about the context the user is in. The word *learning* is appropriate because the system adaptively learns how to recognize different contexts that it encounters, in the same way that a small child (or even adult) learns to recognize different situations based on his or her experiences. In both cases (machine and human), there is a set of inputs or stimuli that are used to reason that a certain situation or context exists. The process of learning (which happens naturally and almost automatically for humans) is figuring out the optimal way to transform a set of inputs into the correct output—in our case, the context.

In general, the output of a machine learning algorithm can be of two types: 1) one or more continuous variables represented by real numbers, 2) a discrete class, which is normally represented by a textual descriptor (e.g., green, male, good, and happy) or an integer or binary code, from a finite set of classes. When the output is of the continuous type, the machine learning task is known as *regression*. When it is of the discrete type, it is known as *classification*. Since context is primarily of a discrete nature, we will focus mostly on classification in this chapter.

In classification, we can understand the learning problem as a process of learning a function, f(·), that maps a vector of inputs $\boldsymbol{x} = (x_1, x_2, \ldots x_n)$ to the correct output $y \in Y=\{y_1, y_2, \ldots y_m\}$. This is expressed symbolically as:

$$f : \boldsymbol{x} \rightarrow y \tag{9.1}$$

Note that the function need not be a deterministic one. It may consist of probability distributions of the input and output variables, describing a stochastic relationship between them. Ultimately, however, a classifier chooses a single class based on the function's inputs and using a particular selection criterion, such as *maximum likelihood*.

Machine learning techniques mostly fall into one of two categories based on how the function f is learned: 1) *supervised learning* and 2) *unsupervised learning*. In supervised learning, a set of "training data" is used. The training data is a limited set of input data for which the correct or optimal output is known [i.e., $\mathbf{s} = (\boldsymbol{x}, y)$]. This is sometimes known as labeled data because in most cases a human has manually labeled the dataset with the correct output. Therefore, it is usually laborious and costly to obtain such data. On the other hand, in unsupervised learning no training data are used, and the goal is to recognize implicit patterns in the data. There is, in fact, a third category of machine learning, known as *semisupervised learning*, in which both labeled and unlabeled data are used. The aim in semisupervised learning is to combine techniques from supervised and unsupervised learning and use all available data (labeled and unlabeled) to achieve a better learning result. In this book, we will mainly focus on supervised learning techniques, as they are the most commonly used in context recognition.

Most of the techniques in machine learning have been developed to deal with data samples that are independent from one another. For example, in a machine learning algorithm to classify credit card applications as "high-risk" or "low-risk," it is reasonable to assume that one credit card application does not influence the risk level of another credit card application. In contextual reasoning, however, our data samples mostly (if not exclusively) come from *time-series data*, where the assumption of independence does not strictly hold.

We can understand this intuitively from the fact that what a person is doing at one moment usually affects what he or she does at the next moment. In other words, a data sample at any given time epoch is dependent on the values of the data at prior time epochs, especially those epochs immediately prior to the given epoch. To a certain extent, we can choose to ignore these dependencies, in order to use a machine learning model with a more simple structure, but the best performance will be achieved when these time dependencies are taken into account. The subset of techniques from statistics and machine learning that work with this kind of data is known as *time-series analysis* [7] or *sequential machine learning* [8, 9].

There are certainly many such techniques to choose from, each having relative advantages and disadvantages. Table 9.2 presents a condensed list of techniques from machine learning that could be applied to contextual reasoning. It is not exhaustive, but provides a broad overview of the most important machine learning techniques for contextual reasoning. A subset of these techniques will be discussed at an introductory level in the sections below.

It is tempting when presented with a long list of machine learning techniques to want to find and focus on the "best" one, applying it indiscriminately in any given domain. The *no free lunch theorem for supervised learning* states, however, that no single machine learning algorithm performs better than any other across all problems. Thus, it is unavoidable that some process of comparison or analysis must be carried out for any particular domain before the most appropriate machine learning technique is chosen.

9.3.2 Naïve Bayes' Classifiers

Before we turn to examples of sequential machine learning techniques, we start with an example of a simple machine learning technique, in order to motivate more complex concepts later on. This simple yet surprisingly effective classifier is called the *naïve Bayes' classifier*. It falls within a larger set of powerful graphical probability models, called *Bayesian networks*, but takes advantage of a few simplifying assumptions.

Table 9.2

List of Machine Learning Techniques and Associated References

Supervised Learning	*References*
Used with sliding window method:	
Naïve Bayes'	[10-11] [13]
Bayesian networks	[24]
Decision trees	[25] [42]
LDA/QDA	[34-36] [38-40]
Logistic regression	[40-42]
SVMs/kernel machines	[26-28] [34] [41]
Neural networks	[40] [42]
Used directly on time-series data:	
HMMs	[18-19] [23] [35] [43]
Conditional random fields	[29-32] [34] [43]
Unsupervised Learning	*References*
Clustering	[37]
Principal component analysis	[36]
Self-organizing maps	[44-46]

First, we assume that all data samples are *independent and identically distributed* (*iid*). In other words, we ignore the time dependence between the data samples, and we assume that the set of processes producing the data are stable throughout the dataset (i.e., stationary processes). Second, although the data samples may be multivariate, we assume that all of the measured variables are conditionally independent from one another given the class. In many (if not most) applications, these assumptions are not strictly true, but the time dependencies and the possible dependencies between the measured variables are simply ignored in order to reduce the problem to a more tractable solution, hence the name "naïve."

Despite these simplifications, naïve Bayes' classifiers have been shown to perform well in a number of machine learning domains, including document classification [10], image classification [11], and activity recognition [12, 13]. We will use them as a starting point in our discussion of contextual reasoning because they illustrate a number of fundamental concepts before we move on to more complex machine learning techniques.

The basic structure of a naïve Bayes' classifier is shown graphically in Figure 9.2, where the unshaded node represents a class set $C_k = \{c_1,c_2,\ldots,c_n\}$ and the shaded nodes represent a d-dimensional input vector $\boldsymbol{x} = (x_1,x_2,\ldots,x_d)$. The arrows represent the dependencies between the classes and the input vector. The lack of arrows between the variables in the input vector indicates that they are conditionally independent given the class.

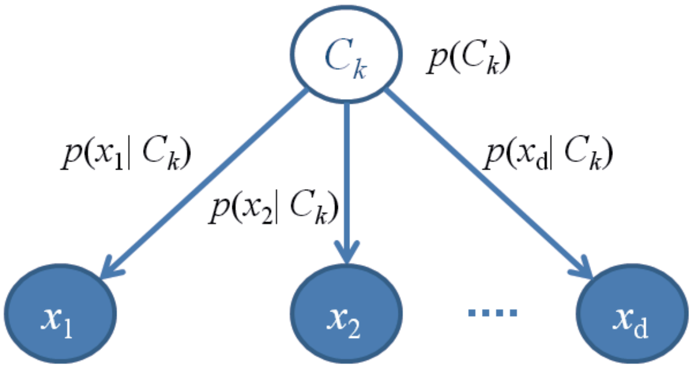


Figure 9.2 Structure of the naïve Bayes' classifier.

We assume that, in general, the class cannot be observed directly, whereas the input vector can be; therefore, we would like to classify a dataset into classes, according to this input vector. In other words, we want to know the posterior probability distribution of C_k given $\boldsymbol{x}$, which according to Bayes' theorem is:

$$p(C_k \mid \boldsymbol{x}) = \frac{p(C_k)p(\boldsymbol{x} \mid C_k)}{p(\boldsymbol{x})} = \frac{p(C_k)p(\boldsymbol{x} \mid C_k)}{\sum_k p(\boldsymbol{x} \mid C_k)p(C_k)} \tag{9.2}$$

In order to evaluate the likelihood $p(\boldsymbol{x}|C_k)$, we would ordinarily need to compute the product:

$$p(x_1 \mid C_k)\prod_{i=2}^{d} p(x_i \mid x_1,\ldots,x_{i-1},C_k),$$

expressing the conditional dependence among the input variables within each class, which may be very difficult if the number of input dimensions is large. By assuming that the input variables are conditionally independent given the class, the product reduces to:

$$p(\mathbf{x}|C_k) = \prod_{i=1}^{d} p(x_i|C_k) \tag{9.3}$$

The most common way to evaluate $p(\boldsymbol{x}|C_k)$ is to use a training dataset, where the values are of $\boldsymbol{x}$ are labeled with the correct class (i.e., supervised learning). If $\boldsymbol{x}$ is composed of discrete variables (e.g., with multinomial distribution), $p(x_i|C_k)$ can be estimated as the frequency that x_i takes on a particular value for a particular class; stated another way:

$$p\left(x_i = x_j \middle| C_k = C_m\right) = \frac{\text{count}(x_i = x_j, C_k = C_m)}{\text{count}(C_k = C_m)} \tag{9.4}$$

If $\boldsymbol{x}$ is composed of continuous variables, then other methods must be used to estimate this class conditional density, such as a histogram, kernel estimator, or a k-nearest-neighbor approach. Alternatively, one can use parametric methods, especially if certain assumptions about the distribution can be made (e.g., that it is Gaussian). For more details on parametric and nonparametric estimation methods, see [14].

The prior probabilities of the classes, $p(\boldsymbol{C_k})$, can also be estimated using frequency counts:

$$p(C_k = C_m) = \frac{\text{count}(C_k = C_m)}{N} \tag{9.5}$$

where N is the total number of training samples. Thus, $p(C_k = C_m)$ is simply the fraction of the training samples that are labeled with class C_m. In some cases, however, it may be that the training data disproportionately represent certain classes compared to the real-world case. Therefore, one can use domain knowledge to adjust the class priors appropriately. For example, in some applications, it may be appropriate to use equal probabilities for each class. Just be sure that the probabilities over all classes sum to one.

Note that in (9.2), the denominator will be the same for all classes. Since our ultimate goal is to choose the most likely class [i.e., the largest posterior probability, $p(C_k|\boldsymbol{x})$], it is not necessary to actually compute the value of the denominator. Given data sample $\boldsymbol{x_j}$, one can find the largest value of $p(C_k)p(\boldsymbol{x_j}|C_k)$ and classify the sample into the respective class given by the arguments.

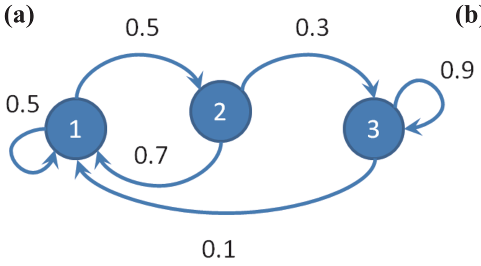
9.3.3 Hidden Markov Model (HMM)–Based Classifiers

In the next section, we turn to a slightly more complex but more powerful machine learning technique that uses HMMs. Generally speaking, HMM-based classifiers will perform better than simple naïve Bayes' classifiers for time-series data because they are able to model the time dependency in the data. For this reason, HMMs are one of the most popular machine learning techniques for contextual reasoning.

9.3.3.1 Markov Chains and the Markov Property

First, we introduce the concept of a *Markov chain*, a mathematical system used to model stochastic processes that exhibit a *Markov property*, which will be described shortly. These processes are usually interpreted as belonging to a system that can be, at any given time, in one state from a set of possible states, S. In this chapter, we will consider only discrete state processes with a finite set of possible states, S = $\{S_1, S_2, \ldots S_M\}$. The system transitions from one state to another in a stochastic manner, meaning that the transitions can be analyzed using probabilities. Such a system is represented graphically in Figure 9.3(a), where the probabilities governing the evolution of states {1, 2, 3} are shown as arrows between the nodes in the graph or as loops back onto themselves.

(a)



(b)

Sj / Si	State 1	State 2	State 3
State 1	0.5	0.5	0.0
State 2	0.7	0.0	0.3
State 3	0.1	0.0	0.9

Figure 9.3 A Markov chain and associated transition probability matrix.

For example, if the system is in state 1 at time *t*, it has a 50% probability of transitioning at time *t*+1 to state 2 and a 50% probability of remaining in state 1. Note that the lack of an arrow or loop indicates a zero probability of that particular transition. The full set of transition probabilities can also be expressed as an *M* x *M* matrix, where *M* is the number of states, as shown in Figure 9.3(b) for this example Markov chain. These correspond to conditional probabilities, where element *ij* is the conditional probability of transitioning from the *i*th state to the *j*th state at any epoch *t*.

When a Markov chain can be fully specified in this manner, it is said to be a first-order Markov process. The two-dimensional structure of the transition probabilities ensures that all of the information available to predict the next state of the process is contained in the present state. In probability terms, this means that the conditional probability of the system being in state s_j at time *t*+1, given knowledge that the system is in state s_i at time *t*, is equal to the conditional probability of the same outcome at $t + 1$, given the same knowledge of the system's state at time t plus information of its state at time $t - 1$ and earlier; i.e. stated another way:

$$P\big(S(t+1) = s_j \big| S(t) = s_i\big) = \\ P\big(S(t+1) = s_j \big| S(t) = s_i, S(t-1) = s_k, \ldots, S(1) = s_l\big) \qquad (9.6)$$

When (9.6) holds for a system, we say it has the *first-order Markov property*. The second-order Markov property is when the state at $t + 1$ depends on the current state and the state at $t - 1$. This, however, would require a three-dimensional transition probability matrix ($M \times M \times M$) to be fully specified. In general, it is also possible for a system to have an nth-order Markov property, which would require an n-dimensional transition probability matrix. Only first-order Markov processes will be considered further in this chapter.

In other words, a (first-order) Markov process' exact "path" to a particular state does not provide any more useful information than the current state of the system. An example of this kind of process would be a chess game, where in order to assess the possible next move, the current configuration of the board is just as useful as the exact sequence of moves in the game up to the current configuration.

9.3.3.2 Hidden States and Observable Signals

There are two differences between an ordinary Markov model and a HMM. The first is that in HMMs, the state of the system at any given moment is not directly observable (i.e., is hidden). Second, for each transition of states, the system "emits" an observable signal, which is stochastically correlated with the unobservable state. All that one can "know" about the system's state is contained in the observed signal (which may have several components), and one can make probabilistic inferences about the state based on the correlation between the states and the observations.

As an example, consider again a chess game. Suppose now that an observer is far away and cannot make out the board but can see the players' faces and gestures and also knows something about their characters. For example, based on previous experience, this observer knows that player 1 usually smiles broadly after making a good move and often groans when she realizes she made a bad move. Player 2 is much more calm and relaxed during the game but almost always smiles the moment he realizes he has checkmate.

This could possibly be modeled using a HMM where the states are {*game even, player 1 winning, player 2 winning, player 1 wins, player 2 wins*}. The observed signals are the smiles, gestures, and sounds of the players, but the actual state is "hidden" because the board is far away. Another variant of this example would be where the observer just doesn't know the rules of chess, so is unable to assess the state, except at the very beginning and very end.[26]

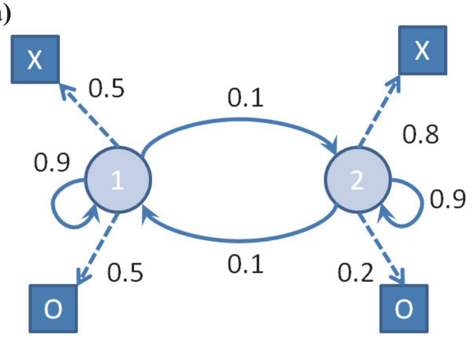
HMMs can either be discrete or continuous, depending on the nature of the observed signal. For simplicity we will consider only discrete HMMs in this chapter, but many approaches have been developed to handle continuous observation signals, for example, by describing the signal parametrically. See [19] for examples.

Figure 9.4(a) shows a graphical representation of a HMM, similar to that of a Markov chain but adding the observation signal. In this example, there are only

[26] This would be an example of a partially HMM. See [15] for details.

two possible states {1, 2}, and the lighter shading indicates they are not directly observable. In this particular example, the transition probabilities constrain the system to only occasionally change states, since most of the probability for each state (90%) is concentrated in the loops. These transition probabilities are shown as before in Figure 9.4(b). In addition, there are two possible output signals Y = {X, O}. In state 1, these are expressed with equal probability, whereas state 2 has a tendency to express more X's than O's (80% vs. 20%). These "emission" probabilities are shown in Figure 9.4(c). In general, for discrete HMMs the emission probabilities are given by an MxN matrix, where M is the number of states and N is the number of observation symbols.[27]

(a)



(b)

S_j S_i	State 1	State 2
State 1	0.9	0.1
State 2	0.1	0.9

(c)

Y_i S_i	X	O
State 1	0.5	0.8
State 2	0.5	0.2

Figure 9.4 A Markov chain and associated transition probability matrix.

9.3.3.3 Machine Learning Tasks Using HMMs

Now that we have laid out the basic concepts of HMM, let us explore how they are used in machine learning and contextual reasoning. From here forward it will be important to use clearly defined mathematical notation for HMMs in order to keep the text concise. We adopt the notations listed in Table 9.3, which have already been partially used above.

There are primarily four machine learning tasks that are commonly performed with HMMs, and each task has its own standard algorithm. These tasks are described as follows:

[27] For continuous HMMs, the emission probabilities might be expressed by a parametric model (e.g., a Gaussian with different parameters for each state).

- **Supervised model learning.** Given a sequence of observations Y_1^T that are labeled with the true sequence of states S_1^T, estimate the model parameters λ.[28]
- **Unsupervised model learning.** Given a sequence of observations Y_1^T and an initial guess of the model parameters λ_0, find the set of parameters λ_n that optimally matches the observations.
- **Online recognition.** Given a model λ and a sequence of observations Y_1^t, find the most likely state s_i at time *t*. Repeat for each epoch, *t*=1,…,T to obtain the sequence of states S_1^T. This is also known as sequential processing.
- **Offline recognition.** Given a model λ and a sequence of observations Y_1^T, find the most likely sequence of states S_1^T [i.e., maximum $\Pr(S_1^T|Y_1^T,\lambda)$]. This is also known as batch processing.

Table 9.3

Mathematical Notations of the HMMs

Notation	*Description*
T	The number of elements in a sequence of observations.
Y_1^T	A sequence of observations, indexed from 1 to *T*.
M	The number of possible states in the model.
$S = \{s_1, s_2, \dots, s_M\}$	The set of possible states.
$S(t)$	The state at time *t*.
S_1^T	A sequence of states, indexed from 1 to *T*.
N	The number of possible observation symbols.
$Y = \{y_1, y_2, \dots, y_N\}$	The set of possible observation symbols.
$Y(t)$	The observation symbol at time *t*.
$a_{ij} = \Pr(S(t+1) = s_j \vert S(t) = s_i)$	The conditional probability that the system will be in state s_j at time t+1, given the state s_i at time *t*. Also called the transition probability from state *i* to *j*.
$b_k = \Pr(Y(t) = y_k \vert S(t) = s_j)$	The probability that state s_j will emit observation symbol y_k. Also called the emission probability for y_k and s_j.
$B = \{b_j(k)\}$ for $j = 1, \dots, M$ and $k = 1, \dots, N$	The set of emission probabilities for all observation and symbols and states, also called the emission probability matrix.
$\pi = Pr\{S(1) = s_j\}$ for $j = 1, \dots, M$	The set of initial state probabilities for each state s_j.
$\lambda = \{A, B, \pi\}$	The complete parameter set of the model.

There are two subtle but important differences between task 3 and task 4. First, task 4 is an offline task, so the algorithms used to achieve it can use the full set of observations Y_1^T at any time in its operation, whereas task 3 can only use

[28] Note that we have just broken the assumption about the state of the system not being directly observable. This is true only for the model learning phase (i.e., training). During the recognition phase, we still assume the states are hidden.

past observations. Second, task 3 operates at each epoch individually and will produce an optimal choice of state at that particular epoch (given the available observations), whereas task 4 chooses the optimal sequence S_1^T, which has the highest probability of producing the observations Y_1^T. In other words, they have two distinct choices of optimality criterion, which can sometimes lead to differing results.

Supervised Model Learning

This is the most straightforward of the four tasks. First, to estimate the transition probability matrix A, we simply need to count up the occurrences of each transition type i→j (including where j=i) for all i,j={1,…,M}. The transition probabilities are then estimated as:

$$a_{ij} = \frac{\text{count}(S(t) = s_i, S(t+1) = s_j)}{\text{count}(S(t) = s_i)} \tag{9.7}$$

This will give us M^2 transition probabilities, allowing us to fully specify the matrix A. In practice, however, some of these transition probabilities will be zero. This *could* be because the probability of that particular transition is actually zero, or it could be simply because the transition is not represented in the limited training sequence S_1^T. Therefore, it is good practice to manually inspect the matrix A and use one's domain knowledge to determine if some possible transitions are not represented in the dataset when they are indeed possible. One approach to rectify this problem is to add a *pseudocount* to each such transition. The value of the pseudocount can either be one (Laplace's rule) or can be set to some other positive integer value based on domain knowledge.

Next, to estimate the emission probability matrix B, we count for each state the number of times each observation symbol has been expressed. The emission probability is then estimated as:

$$b_j(k) = \frac{\text{count}(Y(t) = y_k, S(t) = s_j)}{\text{count}(S(t) = s_j)} \tag{9.8}$$

Again we can use Laplace's rule or some other pseudocount value, based on domain knowledge, in order to correct any probabilities that would otherwise be unreasonably set to zero.

Last, we must estimate the initial state probabilities that should be specified in π. In practice, the methods for setting this parameter are application-dependent. The simplest approach is to set it to an equal value for each state that sums to one over all states (i.e., $1/M$). If for some particular application, it is more likely that the HMM is initialized in some particular state or states, the values for these states can be adjusted according to domain knowledge (maintaining that the probabilities over all the states sum to one). Another possibility is when the initial state of an HMM is known with great certainty, the value for that state can be set to one and all other initial state probabilities set to zero. Note that no matter how parameter π is specified, its importance diminishes as t increases, so even the simple $1/M$ approach may be adequate.

Unsupervised Model Learning

In this task, we would like to estimate the parameters of the HMM based on an unlabeled observation sequence Y_1^T. In order to do so, we need an initial guess of the parameters, λ_0. The standard algorithm for optimizing the parameters based on this initial guess and an observation sequence is called the *Baum-Welch algorithm* or the *forward-backward algorithm*. A detailed treatment of this algorithm (which includes several variants) is beyond the scope of this chapter, but we only mention here that it is an expectation–maximization (EM) procedure, where an improved estimate of the model parameters is given after each iteration, until converging to a local maximum of $Pr(Y_1^T|\lambda)$. The result varies based on the initial guess of the parameters, so one approach is to run the algorithm a number of times with randomly generated initial guesses (subject to reasonable constraints). This won't necessarily yield a globally optimized set of parameters, but no finite time approach to obtain a global maximum of $Pr(Y_1^T|\lambda)$ is known. More details about the Baum-Welch algorithm can be found in [14] or [19].

Online Recognition

Given a HMM defined by parameters λ and a history of observations Y_1^T, the goal of this task is to determine the most likely state s_i at time t. In other words, we need to find the state that gives the maximum probability $Pr(S(t)|Y(t),\lambda)$ from the set of all possible states, also known as the maximum a posteriori (MAP) estimate of S(t):

$$\hat{S}(t)_{MAP} \equiv \arg\max_i \Pr\,(S(t) = s_i|Y(t),\lambda) \tag{9.9}$$

In order to evaluate this probability, we use an identity called the Chapman–Kolmogorov equation and the Markov property to obtain a prediction equation:

$$p(S(t) = s_j \mid Y(t-1)) = \sum_{j=1}^{M}\sum_{i=1}^{M} a_{ij}\, p(S(t-1) = s_i \mid Y(t-1))\delta(s_j - s_i) \tag{9.10}$$

where $\delta(s_j - s_i)$ is the Dirac delta measure that is equal to zero for $s_i \neq s_j$ and one for $s_i = s_j$ [16]. Then, given a new observation $Y(t)$, we use Bayes' rule to obtain the updated equation:

$$p(S(t) \mid Y(t)=k) = \frac{b_j(k)p(S(t)=s_j \mid Y(t-1))}{\sum_{i=1}^{N} b_i(k)p(S(t)=s_i \mid Y(t-1))} \tag{9.11}$$

These two equations form the basis of the prediction and update steps of this recognition algorithm, respectively. At each time epoch, the prediction density $p\big(S(t)=s_j \big| Y(t-1)\big)$ is calculated and then updated using the new observation and (9.11). This technique is often known as Bayesian optimal filtering or recursive Bayesian estimation. In some references, it has been called the grid-based method, presumably because the transition probabilities are represented as a grid (i.e., matrix) [16].

Offline Recognition

The goal of this task is to find the most likely sequence of states S_1^T, given a model λ and a sequence of observations Y_1^T. Since this task is performed after *all* of the observations in the sequence are available, there is more information available to perform it compared to the online recognition task. Several different algorithms have been used in order to perform this type of task, but the most commonly used is the Viterbi algorithm [14].

The Viterbi algorithm defines two matrices, $\delta(i,t)$ and $\psi(i,t)$. $\delta(i,t)$ represents "best path" probabilities for traveling particular paths in the sequence of states. For example, $\delta(3,4)$ is the probability of reaching state 3 at time 4 using the best or most likely path to get there. $\psi(i,t)$, on the other hand, represents "back pointers" to the states that produce the most probable paths. For example, if $\psi(3,4) = 2$, this means that, in order for the HMM to be in state 3 at time 4 with maximum probability $\delta(3,4)$, it must be in state 2 at time 3. Using the same notation $\delta(i,T)$ represents the probability of being in state i at the end of the sequence (i.e., time T).

The algorithm starts at time epoch one and then recursively calculates the successive values in $\delta(i,t)$ and $\psi(i,t)$. When the values of $\delta(i,T)$ have been calculated, the algorithm simply chooses the value of i that gives the maximum $\delta(i,T)$ and then backtracks through the sequence using values of $\psi(i,t)$ to determine the most likely path.

Formally, the Viterbi algorithm is implemented in four steps: 1) initialization, 2) recursion, 3) termination, and 4) path backtracking [17], described as follows.

1) Initialization: The values of $\delta(i,t)$ and $\psi(i,t)$ for time t=1 are set:

$$\begin{matrix} \delta(i,1) = \pi_i b_j(y_1) \\ \psi(i,1) = 0 \end{matrix} \quad 1 \le i \le M \tag{9.12}$$

Note that $\psi(i,1)$ has no clear meaning, since it specifies a state of the system prior to the first epoch. By convention it is set to zero, but it is never actually used in the algorithm.

2) **Recursion:** The values of $\delta(j,t)$ and $\psi(j,t)$ for times $t=2,\ldots,T$ are recursively calculated:

$$\left.\begin{aligned} \delta(j,t) &= \max_{1\le i\le M}\left[\delta(i,t-1)a_{ij}\right]b_j(Y(t)) \\ \psi(j,t) &= \arg\max_{1\le i\le M}\left[\delta(i,t-1)a_{ij}\right] \end{aligned}\right\} \begin{cases} 2 \le t \le T \\ 1 \le j \le M \end{cases} \tag{9.13}$$

Note that we have changed the state index variable used in δ and ψ from i to j. This is done in order to remain consistent with the notation for a_{ij}, representing the probability of transitioning from state i to state j. Therefore, here i represents the state at time $t-1$ and j represents the state at time t.

3) **Termination:** Select the maximum probability, P_T, among the values of $\delta(i,T)$ and the corresponding state, $S(T)$, according to the following criteria:

$$\begin{aligned} P_T &= \max_{1\le i\le M}\left[\delta(i,T)\right] \\ S(T) &= \arg\max_{1\le i\le M}\left[\delta(i,T)\right] \end{aligned} \tag{9.14}$$

4) **Path backtracking:** Determine the corresponding most likely path (state sequence) using the back pointers, $\psi(i,t)$, and the following backward recursion:

$$S(t) = \psi(S(t+1),t+1) \quad t = T-1, T-2, \ldots, 1 \tag{9.15}$$

9.3.3.4 Concluding Remarks about HMMs

This concludes our coverage of HMMs in this chapter. For readers not well-versed in probability theory (especially concepts such as Markov chains), these sections may have been challenging to digest. Fortunately, implementations of HMMs and the related algorithms for working with them are readily available in many programming languages (e.g., Java, C, and Matlab). In order to use them correctly, it is necessary to understand the concepts outlined here at a basic level. We have not addressed in this chapter various implementation details, such as techniques to avoid numeric underflow or variations of the algorithms to reduce memory requirements. Therefore, particular implementations may differ somewhat from what was presented here. More detailed coverage of HMMs can be found in [5, 18, 19].

9.3.4 The Sliding Window Method

As mentioned in Section 9.2.1, most of the techniques in machine learning have been developed to deal with *independent and identically distributed* (*iid*) data (HMMs are an exception to this). Such methods, in their original form, do not exploit the time dependence inherent in sequential data. A simple but effective technique to benefit from the wide array of available machine learning techniques yet also exploit this time dependency is the *sliding window method*, illustrated in Figure 9.5 [20]. This method effectively encapsulates the (local) time dependence of the data into a new data structure, a "sliding window," allowing the time-series data to be used in the same way as one would use iid data. As a result, one can use any of the classical (iid) machine learning techniques to classify samples of this sliding window.

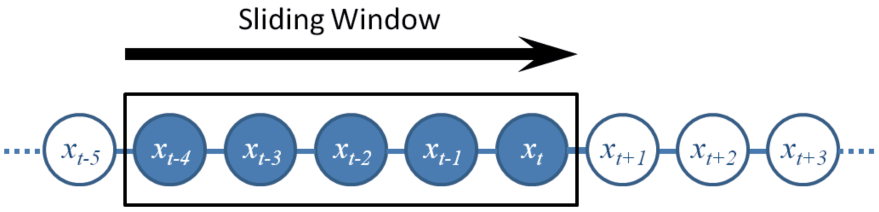


Figure 9.5 The sliding window method.

The best way to elucidate the sliding window method is by example. Suppose we have the following one-dimensional (unitless) data sequence:

$$X_1^{10} = (7,10,3,9,6,5,8,10,5,3)$$

First, a window width, w, must be chosen. The best choice of window width is dependent on the application and determines how much of the local time dependence is preserved in the data. Large windows will capture more time dependence but will also result in more computationally complex learning and classification. Therefore, the best choice is a trade-off between classification performance and computational time. In our example, we will choose a window size of five epochs.

Next, the windows are composed, which will result in a new data structure, H_t. This structure will have $w*d$ dimensions, where d is the original dimensionality of the data. In our case, H_t will be five-dimensional. There are two slightly different approaches to composing the windows, depending on how one wants to handle epochs 1 to $w - 1$, where there is not enough data to "fill" the window. The first approach is keep the number of samples in H_t the same as that of the original data and to use null values where necessary. For example, the first three windows composed from X_1^{10} would be:

$$H_1 = (\text{null}, \text{null}, \text{null}, \text{null}, 7)$$
$$H_2 = (\text{null}, \text{null}, \text{null}, 7, 10)$$
$$H_3 = (\text{null}, \text{null}, 7, 10, 3)$$

The other approach is simply to reduce the *length* of H_t by $w - 1$. Thus, epoch five from our original dataset becomes epoch one of H_t, and the first three windows become:

$$H_1 = (7, 10, 3, 9, 6)$$
$$H_2 = (10, 3, 9, 6, 5)$$
$$H_3 = (3, 9, 6, 5, 8)$$

In either case, as the window "slides" to the right in time, a new value from the sequence X_1^{10} comes in at the last (rightmost) position in H_t and an existing value (or null) slides out of the window with the values in between also being shifted accordingly.

After all the windows in H_t have been composed, the ordering of H_t is no longer important—it can now be regarded as a set instead of a sequence. In other words H_1, H_2, H_3, etc., can be shuffled like a deck of cards into any order. So long as the individual elements within each window are not rearranged, the local time dependence is preserved.

This is beneficial for many reasons. For example, in machine learning, data are usually separated into different groups for training the classifier and later for testing its performance. Now these groups can be chosen by randomly selecting samples from the set H_t, thus ensuring there is no selection bias in these groupings.

9.3.5 Bayesian Networks

Bayesian networks are directed graphical models, where nodes and directed links (i.e., edges) between the nodes represent conditional dependencies between variables (or sets of variables). As mentioned in Section 9.2.2, naïve Bayes' classifiers are a special case from the general framework of Bayesian networks. We saw in Section 9.2.2 that naïve Bayes' classifiers assume the input variables in the model are independent (i.e., no interconnecting arcs between them), but since this assumption is often not correct, Bayesian networks in general can model this interdependence among input variables. In fact, HMMs can also be viewed as a specific type of Bayesian network, called a *dynamic Bayesian network*, with certain constraints placed on its structure [23].

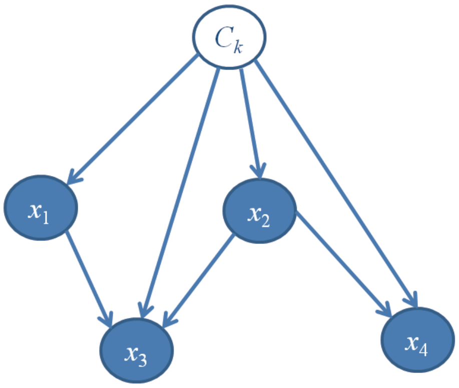


Figure 9.6 Example of a Bayesian network.

In general, Bayesian networks can be used to represent *any* joint probability distribution that consists of a product of conditional distributions. For example, the Bayesian network shown in Figure 9.6 corresponds to the joint distribution defined by the following equation:

$$\begin{aligned} &p(C_k, x_1, x_2, x_3, x_4) = \\ &\quad p(C_k)p(x_1 \mid C_k)p(x_2 \mid C_k)p(x_3 \mid C_k, x_1, x_2)p(x_4 \mid C_k, x_2) \end{aligned} \tag{9.16}$$

In this way, Bayesian networks are a compact and intuitive means to express the probabilistic relationships among observable variables and hidden variables, such as the class label. Unfortunately, we don't have space to cover in detail how Bayesian networks are used in machine learning, but algorithms exist both to learn the graph structure from a labeled dataset and to perform inference on a given Bayesian network given unlabeled data. For detailed treatment of these subjects, see [5] or [24].

9.3.6 Decision Trees

Decision trees, or tree-based methods (which include several variants such as CART, ID3, and C4.5), comprise a simple but widely used machine learning technique that functions as a hierarchical set of if-then-else statements. For example, in the decision tree shown in Figure 9.7(a), classification is performed starting from the top node. For a particular data sample, if the condition lying between nodes 1 and 2 is satisfied (i.e., $x_1 > 5$), the classifier moves to the second node. Otherwise, it proceeds to node 3. The pattern is continued until a "leaf node" is reached (denoted in Figure 9.7 using uppercase letters), at which point a class label is assigned for that data sample.

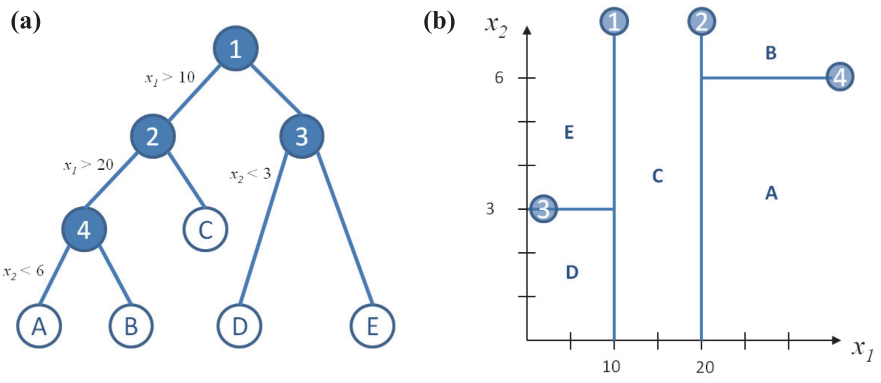


Figure 9.7 Example of a decision tree.

An alternative way to understand decision trees is that they divide an input space into distinct class regions, where each boundary corresponds to one node (i.e., "decision") from the decision tree, as shown in Figure 9.7(b). Each region can define a separate class (as in Figure 9.7), or several regions can be assigned to the same class. It is not necessary that the boundaries are orthogonal like in our example. In so-called oblique decision trees, linear combinations of variables can be used, in order to define non-orthogonal boundaries.

The overall goal with decision trees is to correctly classify as many data samples as possible, while minimizing the total number of regions (in order to avoid the problem of *overfitting*). There are two main alternatives for how to construct a decision tree. The first is to start with a single node and add one node at a time, until a threshold performance criteria is met. The other approach is to first grow a very large tree and then "prune" it (i.e., remove nodes), until a satisfactory balance is achieved between the performance of the tree and its complexity. A detailed discussion of decision trees can be found in [25].

9.3.7 Support Vector Machines (SVMs)

In recent years, SVMs have become a popular choice for building classifiers. The mathematics involved in constructing an SVM classifier are rather complex and beyond the scope of this book, but in this section we will provide sufficient detail to understand the basic concepts of SVMs.

9.3.7.1 Defining a Feature Space Using Kernel Functions

SVMs work by defining a much higher-dimensional space, called a *feature space*, mapped from the original input space. The added dimensions in the feature space are formed from transformations of one or more input variables [e.g., $\varphi\colon (x_i, x_j) \rightarrow$

$(x_i^2,\ x_i x_j,\ e^{x_j}, \dots)$]. Generally speaking, it is not necessary to explicitly perform a mapping from the input space to the feature space, but rather the so-called *kernel trick* is used. The kernel trick uses a *kernel function*, $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which corresponds to the inner product of the feature space. Here the indices i and j reference different data samples in the input space, which are assumed to be vectors. Common kernel functions used in SVMs include:

- Homogeneous polynomials: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^d$;
- Inhomogeneous polynomials: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j + c)^d$;
- Gaussian radial basis functions: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right)$, where $\gamma > 0$;
- Hyperbolic tangents: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\kappa \boldsymbol{x}_i \cdot \boldsymbol{x}_j + c)$.

We will demonstrate the kernel trick with a simple example, where the input space is two-dimensional ($\mathbb{R}^2$). Here we use a homogeneous polynomial kernel function where d=2. We will compute the kernel function for two samples from the input space, and for greater clarity we define these as $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{y} = (y_1, y_2)$. Note that the subscripts now reference the dimension of the input space, as opposed to earlier where they referenced the different data sample. The kernel function for $\boldsymbol{x}$ and $\boldsymbol{y}$ is computed as follows:

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{y}) &= (\boldsymbol{x} \cdot \boldsymbol{y})^2 = (x_1 y_1 + x_2 y_2)^2 \\ &= x_1^2 y_1^2 + 2 x_1 y_1 x_2 y_2 + x_2^2 y_2^2 \end{aligned} \tag{9.17}$$

Inspecting (9.17), we can see that the corresponding feature space has three dimensions, and the two data samples in this feature space become:

$$\varphi(\boldsymbol{x}) = (x_1^2,\ \ \sqrt{2} x_1 x_2,\ \ x_2^2) \tag{9.18}$$

$$\varphi(\boldsymbol{y}) = (y_1^2,\ \ \sqrt{2} y_1 y_2,\ \ y_2^2) \tag{9.19}$$

We can verify (9.18) by checking that $k(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x}) \cdot \varphi(\boldsymbol{y})$. As mentioned above, it is not necessary to explicitly map the input space to the feature space. Only the inner product, computed using the kernel function, is needed to build the classifier. In our simple example, it may seem like this is a trivial advantage, but in more complex examples, the feature space may consist of hundreds or thousands of dimensions. In fact, if a Gaussian radial basis function is used, it is usually evaluated using a Taylor series expansion. Therefore, the corresponding feature space would have infinite dimensions. Fortunately, only the inner product is needed, which can be truncated to desired precision by ignoring higher-order terms.

9.3.7.2 Maximum-Margin Hyperplanes

The main advantage of using a high-dimension space is that the data samples are more easily separated by *hyperplanes* (planes in spaces of arbitrary dimension), allowing linear classifiers to be defined in that hyperspace. These hyperplanes can be considered as similar to the boundaries between the class regions defined in decision trees (recall previous section). They are defined by maximizing the distance between the nearest data sample and the hyperplane, the so-called *maximum margin.* Such data samples lying nearest to the hyperplanes are called *support vectors* because their position supports the definition of the decision boundaries.

This concept is notionally depicted in Figure 9.8. The left side shows a two-dimensional space with data from two classes that are not linearly separable. On the right side, the data are remapped to a higher-dimensional space via a kernel function and plotted in only two selected dimensions (where the linear separation can be seen). The support vectors are the larger-sized data points lying closest to the solid red line, and this line constrains the orientation of the hyperplane in the high-dimensional space.

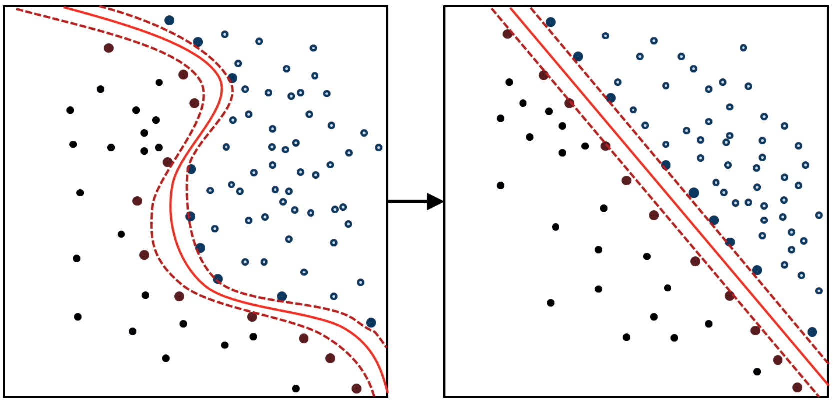


Figure 9.8 SVMs remap data into a higher-dimensional space, where the classes are linearly separable.

9.3.7.3 Concluding Remarks about SVMs

If one is interested in using SVMs to perform classification, it is not necessary to know the mathematical details about how to construct a maximum-margin hyperplane. Several open-source libraries exist for constructing SVMs and for using them to perform classification (e.g., [26, 27]). For most users of SVMs, it is probably sufficient to have a notional idea about how they function and perhaps an understanding of how different kernel functions operate. Users can experiment with different kernel functions applied to their own datasets. The performance of the resulting SVM classifier will vary, depending on the dataset and the chosen kernel function. Because the kernel functions contain one or more free parameters, *parameter tuning* is often performed to optimize the performance. More detailed coverage of SVMs, including techniques for parameter tuning, can be found in [5, 14, 28].

9.4 SUMMARY

In this chapter, we defined contextual reasoning as *"the process of forming higher level inferences about context from lower level information."* We presented this process in the framework of the context pyramid, which aims to aid understanding of contextual reasoning at the conceptual level. We also provided a hypothetical example of this process within the domain of geospatial computing in smartphones.

Next we presented the primary method of contextual reasoning, namely machine learning. We presented three machine learning techniques in detail, including the naïve Bayes' classifier, HMMs, and the sliding window method. Finally, we gave a brief overview of several other commonly used machine learning techniques, including Bayesian networks, decision trees, and SVMs. Detailed coverage of these techniques can be found in the cited references.

In conclusion, by combining the wide variety of sensor data and geospatial information available in smartphones with state-of-the-art techniques of machine learning, smartphones can be made to "reason" about the context in which their users are living. This reasoning allows smartphones to become context-aware, enabling a wide range of context-aware applications and services.

References

[1] Puccio, G. J., M. C. Murdock, and M. Mance, *Creative Leadership: Skills That Drive Change*, San Diego, CA: Sage, 2007.

[2] Hurdus, J. G., and D. W. Hong, "Behavioral programming with hierarchy and parallelism in the DARPA urban challenge and RoboCup," *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 20-22 Aug. 2008, 2008, pp. 503–509.

[3] Giunchiglia, F., "Contextual reasoning," *Epistemologia*, Special Issue on "I Linguaggi e le Macchine," Vol. XVI, 1993, pp. 345-364.

[4] Mitchell, T., et al., "Machine Learning," *Annual Review of Computer Science,* Vol. 4, 1990, pp. 417-433.

[5] Bishop, C. M., *Pattern Recognition and Machine Learning*, New York: Springer, 2006.

[6] O'Connor, B., "Statistics vs. machine learning, fight!" *AI and Social Science* (blog), http://bit.ly/VtpHEX, 2008.

[7] Shumway, R. H., and D. S. Stoffer, *Time Series Analysis and Its Applications: With R Examples*, New York: Springer, 2011.

[8] Kingsbury, N., D. B. H. Tayy, and M. Palaniswamiz, "Multi-scale kernel methods for classification," *Proceedings of the 2005 IEEE Workshop on Machine Learning for Signal Processing*, 2005, pp. 43-48.

[9] Visatemongkolchai, A., and H. Zhang, "Building probabilistic motion models for SLAM," *Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics*, 2007, pp.1629-1634.

[10] Ting, S. L., W. H. Ip, and A. H. C. Tsang, "Is naïve Bayes a good classifier for document classification?" *International Journal of Software Engineering and Its Applications*, Vol. 5, No. 3, 2011, pp. 37-46.

[11] Boiman, O., E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1-8.

[12] Tapia, E. M., S. S. Intille, and K. Larson, "Activity recognition in the home setting using simple and ubiquitous sensors," *Pervasive Computing: Second International Conference, PERVASIVE 2004*, pp. 158-175, A. Ferscha and F. Mattern (eds.), Berlin: Springer, 2004.

[13] Bancroft, J. B., D. Garrett, and G. Lachapelle, "Activity and environment classification using foot mounted navigation sensors," *2012 International Conference on Indoor Positioning and Indoor Navigation*, 2012.

[14] Alpaydin, E., *Introduction to Machine Learning, Second Edition*, Cambridge, MA: The MIT Press, 2010.

[15] Bordes, L., and P. Vandekerkhove, "Statistical inference for partially hidden Markov models," *Communications in Statistics—Theory and Methods*, Vol. 34, No. 5, 2005, pp. 1081-1104.

[16] Ristic, B., S. Arulampalm, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Norwood, MA: Artech House, 2004.

[17] Liu, J., "Hybrid Positioning with Smart Phones," In *Ubiquitous Positioning and Mobile Location-Based Services in Smart Phones*, pp. 159-193, R. Chen (ed.), Hershey, PA: IGI-Global, 2012.

[18] Rabiner, L. R., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, 1989, pp. 257-286.

[19] Fraser, A. M., *Hidden Markov Models and Dynamical Systems,* Philadelphia, PA: Society for Industrial and Applied Mathematics, 2008.

[20] Dietterich, T. G., "Machine learning for sequential data: A review," *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, 2002, pp. 15-30.

[21] Wolpert, D., "The lack of a priori distinctions between learning algorithms," *Neural Computation*, Vol. 8, No. 7, 1996, pp. 1341-1390.

[22] Schaffer, C., "A conservation law for generalization performance," *Proceedings of the Eleventh International Conference on Machine Learning*, 1994, pp. 259-265.

[23] Ghahramani, Z., "An introduction to hidden Markov models and Bayesian networks," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 15, No. 1, 2001, pp. 9-42.

[24] Friedman, N., D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, Vol. 29, 1997, pp. 131-163.

[25] Sutton, C. D., "Classification and Regression Trees, Bagging, and Boosting," *Handbook of Statistics*, Vol. 24, 2005, 303-329.

[26] Chang, C. C., and C. J. Lin, *LIBSVM—A Library for Support Vector Machines*, http://bit.ly/VkqyrS, 2012.

[27] Katholieke Universiteit Leuven, *Least Squares—Support Vector Machines,* http://bit.ly/VkurwO, 2012.

[28] Steinwart, I., and A. Christmann, *Support Vector Machines*, New York: Springer, 2008.

[29] Sutton, C., and A. McCallum, "An Introduction to Conditional Random Fields," *Foundations and Trends in Machine Learning*, Vol. 4, No. 4, 2011, 267–373.

[30] Lafferty, J. D., A. McCallum, F. C. N. Pereira, "Conditional random fields: probabilistic models for segmenting and labeling sequence data," *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 282-289.

[31] Klinger, R., and K. Tomanek, *Classical Probabilistic Models and Conditional Random Fields*, Algorithm Engineering Report TR07-2-013, Technical University of Dortmund, 2007.

[32] Van Kasteren, T., A. Noulas, G. Englebienne and B. Kröse, "Accurate Activity Recognition in a Home Setting", In *Proceedings of the 10th international conference on Ubiquitous computing (UbiComp '08)*, pp. 1-9, 2008.

[33] Fisher, R.A, The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[34] Abidine, M. B., "Evaluating C-SVM, CRF and LDA classification for daily activity recognition," *Multimedia Computing and Systems (ICMCS), 2012 International Conference on*, 2012, pp. 272–277.

[35] Ward, J. A., et al., "Activity recognition of assembly tasks using body-worn microphones and accelerometers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 10, 2006, pp. 1553-1567.

[36] Andreu, J., R. D. Baruah, and P. Angelov, "Real Time Recognition of Human Activities from Wearable Sensors by Evolving Classifiers," *2011 IEEE International Conference on Fuzzy Systems*, 2011, pp. 2786-2793.

[37] Phung, D., et al., "High accuracy context recovery using clustering mechanisms," *IEEE International Conference on Pervasive Computing and Communications, (PerCom)2009*, 2009, pp. 1-9.

[38] Aziz, O., and S. N. Robinovitch, "An analysis of the accuracy of wearable sensors for classifying the causes of falls in humans," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 19, No. 6, 2011, pp. 670-676.

[39] Javed, J., H. Yasin, and S. F. Ali, "Human movement recognition using Euclidean Distance: A tricky approach," *Image and Signal Processing (CISP), 2010 3rd International Congress on*, 2010, pp. 317-321.

[40] Donohoo, B., et al., "Exploiting spatiotemporal and device contexts for energy-efficient mobile embedded systems," *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 1274-1279.

[41] Krishnan, N. C., and S. Panchanathan, "Analysis of low resolution accelerometer data for continuous human activity recognition," *Acoustics, Speech and Signal Processing (ICASSP) 2008. IEEE International Conference on*, 2008, pp. 3337-3340.

[42] Kwapisz, J. R., G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter,* Vol. 12, No. 2, 2011, pp. 74-82.

[43] Vail, D. L., M. M. Veloso, and J. D. Lafferty, "Conditional random fields for activity recognition," *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007, pp. 1331-1338.

[44] Kohonen, T., "The self-organizing map," *Proceedings of the IEEE* 78, no. 9, 1990: 1464-1480.

[45] Suzuki, S., et al., "Activity recognition for children using self-organizing map," *RO-MAN, 2012 IEEE*, 2012, pp. 653-658.

[46] Huang, W., and J. Wu, "Human action recognition based on self organizing map," *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010, pp. 2130-2133.

PUBLICATION 3

L. Pei, **R. E. Guinness**, R. Chen, J. Liu, H. Kuusniemi, Y. Chen, L. Chen, and J. Kaistinen, "Human behavior cognition using smartphone sensors," *Sensors*, vol. 13, no. 2, pp. 1402–1424, 2013.

Sensors **2013**, *13*, 1402-1424; doi:10.3390/s130201402

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

Human Behavior Cognition Using Smartphone Sensors

Ling Pei [1,*], Robert Guinness [1], Ruizhi Chen [1,2], Jingbin Liu [1], Heidi Kuusniemi [1], Yuwei Chen [1], Liang Chen [1] and Jyrki Kaistinen [3]

[1] Department of Navigation and Positioning, Finnish Geodetic Institute, FIN-02431 Masala, Finland; E-Mails: robert.guinness@fgi.fi (R.G.); ruizhi.chen@fgi.fi (R.C.); jingbin.liu@fgi.fi (J.L.); heidi.kuusniemi@fgi.fi (H.K.); yuwei.chen@fgi.fi (Y.C.); liang.chen@fgi.fi (L.C.)

[2] Conrad Blucher Institute for Surveying & Science, Texas A&M University Corpus Christi, Corpus Christi, TX 78412, USA; E-Mail: ruizhi.chen@tamucc.edu

[3] Psychology of Evolving Media and Technology Research Group, Institute of Behavioural Sciences, University of Helsinki, 00014 Helsinki, Finland; E-Mail: jyrki.kaistinen@helsinki.fi

* Author to whom correspondence should be addressed; E-Mail: ling.pei@fgi.fi; Tel.: +358-9-2955-5315; Fax: +358-9-2955-5200.

Received: 28 November 2012; in revised form: 26 December 2012 / Accepted: 15 January 2013 / Published: 24 January 2013

Abstract: This research focuses on sensing context, modeling human behavior and developing a new architecture for a cognitive phone platform. We combine the latest positioning technologies and phone sensors to capture human movements in natural environments and use the movements to study human behavior. Contexts in this research are abstracted as a Context Pyramid which includes six levels: Raw Sensor Data, Physical Parameter, Features/Patterns, Simple Contextual Descriptors, Activity-Level Descriptors, and Rich Context. To achieve implementation of the Context Pyramid on a cognitive phone, three key technologies are utilized: ubiquitous positioning, motion recognition, and human behavior modeling. Preliminary tests indicate that we have successfully achieved the Activity-Level Descriptors level with our LoMoCo (Location-Motion-Context) model. Location accuracy of the proposed solution is up to 1.9 meters in corridor environments and 3.5 meters in open spaces. Test results also indicate that the motion states are recognized with an accuracy rate up to 92.9% using a Least Square-Support Vector Machine (LS-SVM) classifier.

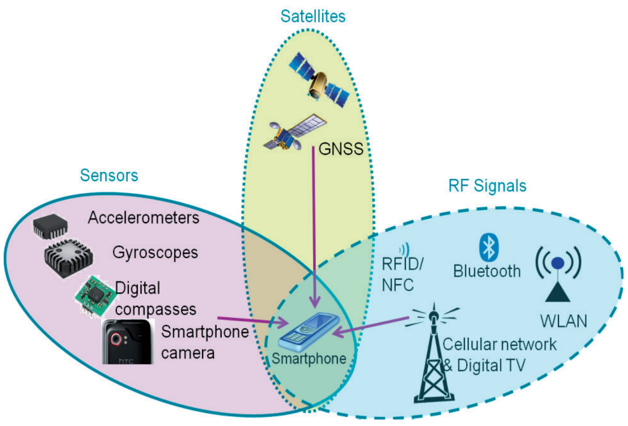
Keywords: sensing; location; motion recognition; LS-SVM; cognitive phone; human behavior modeling

1. Introduction

Human behavior modeling and activity interpretation are of increasing interest in the information society. Social applications such as assisted living and abnormal activity detection draw a lot of attention among scientists [1]. Meanwhile, smartphone sensing technologies are nowadays developing at an incredible pace. The smartphone boasts a healthy variety of sensor options for sensing the social environment. Various locating and context related sensors and network technology are embedded into mobile phones, such as GPS, WLAN (a.k.a. Wi-Fi), cellular network antennae, Bluetooth, accelerometers, magnetometers, gyroscopes, barometers, proximity sensors, humidity sensors, temperature sensors, ambient light sensors, cameras, microphones, *etc.* With this array of input or stimulus options, coupled with capable computational and networking functions, the smartphone becomes an attractive "cognitive" platform, which has a great potential to achieve an enough high intelligence to take up on the questions of social context, such as "Where are you?", "What are you doing?", "How are you feeling?", "Who are you with?", "What is happening?", and "Why are you here?". This article presents an approach to sensing human behavior using a cognitive phone and summarizes the current status of our research work.

The question "where are you?" has been studied in the navigation and positioning fields for many decades. With the explosive growth of the capabilities in handheld computing devices, an increasing amount of research has been focused on positioning solutions using a mobile phone. In order to achieve location awareness both indoors and outdoors, as shown in the Figure 1, three families of smartphone-based positioning solutions have been studied extensively: satellite-based solutions, sensor-based solutions, and RF (radio frequency) signal-based solutions [2].

Figure 1. Three families of smartphone-based positioning solutions.



For outdoors, navigation mainly relies on satellite-based technologies. Having a wide coverage and high accuracy, standalone global navigation satellite systems (GNSS), namely for example the Global Positioning System (GPS), are the most widely applied positioning technology in smartphones. Due to the developments of visible GNSS constellations, the GNSS receiver of a smartphone has extended the positioning capability to multiple satellites systems. For instance, the Chinese phone manufacturer ZTE, together with Russian GLONASS chipset manufacturer AFK Sistema, has developed the first smart phone which embeds both GLONASS and GPS receivers. In addition, assisted GPS, also known as A-GPS or AGPS, enhances the performance of the standard GPS with additional network resources [3,4].

The existing RF infrastructures introduce some alternatives to positioning technologies on a smartphone. Positioning methods using the cellular network and WLAN are now standard features of various smartphones, such as iPhone and Android phones. Nokia has likewise developed a WiFi triangulation system, which now means that the user is more likely to get a positioning fix while indoors or in an urban canyon [5]. Furthermore, short-range RF signals such as Bluetooth [6–11] and RFID [12] are also the options for making estimates of a mobile user's location, for instance, by using proximity, fingerprinting, or triangulating.

Built-in sensors of a smartphone offer the opportunity of continuous navigation when the positioning infrastructures are unavailable. Typically, built-in sensors of a smartphone such as accelerometer, magnetometer, and gyroscope can be utilized to calculate the smartphone's speed, heading, orientation, or motion mode. The above mentioned outputs can then be applied in a pedestrian dead reckoning (PDR) algorithm to assist positioning in challenging environments where the GPS performance is poor or WLAN positioning is unavailable [13–15]. In addition, the camera in a smart phone is also a potential positioning sensor. Ruotsalainen [16,17] uses a camera on a Nokia N8 smartphone to detect the heading change of a mobile phone user. Taking advantage of the magnetometer in modern smartphones, IndoorAtlas Ltd. (Oulu, Finland) pioneers magnetic anomaly-based indoor positioning [18]. Lastly, hybrid solutions [19–21] are adopted to improve the availability and reliability of positioning by integrating all three types of solutions.

Meanwhile, human motion has been widely studied for decades, especially in recent years using computer vision technology. Poppe gives an overview of vision-based human motion analysis in [22]. Aside from vision-based solutions, sensor-based approaches are also extensively adopted in biomedical systems [23–26]. Most of the previous motion recognition related research assumed that the Micro-Electro-Mechanical Systems (MEMS) inertial sensors used are fixed on a human body in a known orientation [27–30] (e.g., in a pocket, clipped to a belt or on a lanyard) and that an error model can be obtained via training to a handful of body positions. Yang [31] uses a phone as the sensor to collect activities for off-line analysis purposes. In general, human physical activity recognition using MEMS sensors has been extensively applied for health monitoring, emergency services, athletic training, navigation, [32,33]. Since motion sensors such as accelerometers, gyroscopes and magnetometers are integrated into a smartphone, they bring the opportunity to assist navigation with knowledge about the motion of a pedestrian [34].

Together these developments suggest that locating and motion recognizing capabilities can enable the cognitive ability of sensing human behavior using a smartphone. For instance, Eagle and Pentland [35] introduce a system for sensing complex social systems using Bluetooth-enabled phones. Adams *et al.* [36] present online algorithms to extract social context: Social spheres are labeled locations of significance, represented as convex hulls extracted from GPS traces. Anderson *et al*. [37] explore the potential for use of a mobile phone as a health promotion tool. They develop a prototype application that tracks the daily exercise activities of people, using an Artificial Neural Network (ANN) to analyse GSM (Global System for Mobile communications) cell signal strength and visibility to estimate a user's movement. Choudhury and Pentland [38] develop methods to automatically and unobtrusively learn the social network structures that arise within human groups based on wearable sensors. Choudhury *et al.* [39] introduce some of the current approaches in activity recognition which use a variety of different sensors to collect data about users' activities. In this paper probabilistic models and relational information are used to transform the raw sensor data into higher-level descriptions of people's behaviors and interactions. Lane *et al*. [40] survey existing mobile phone sensing algorithms, applications, and systems. Campbell and Choudhury first introduce the Cognitive Phone concept and enumerate applications utilizing cognitive phones in [41]. Even though the term Cognitive Phone has not been officially defined yet, from the examples given by [41], the Cognitive Phone is argued to be the next step in the evolution of the mobile phone, which has the intelligence of sensing and inferring human behavior and context.

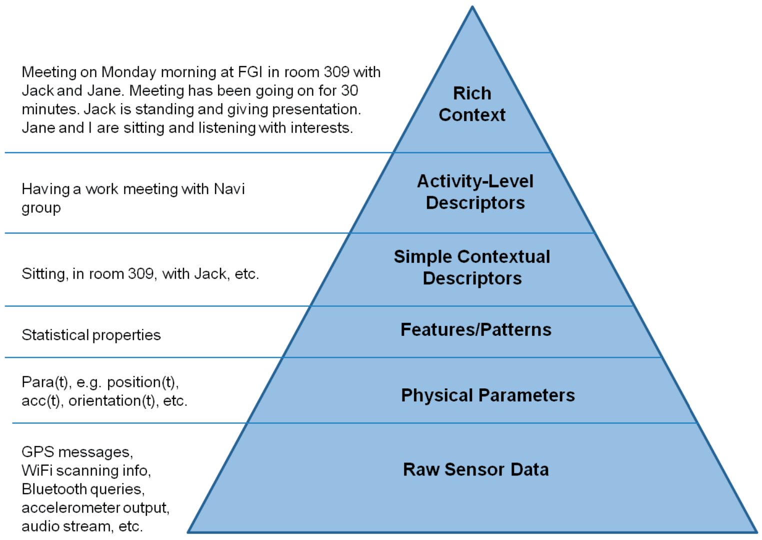
Similarly, this paper will introduce an approach to sensing human behavior, which primarily relies on ubiquitous positioning technologies and motion recognition methods. In the above cognitive research, positioning technologies such as GPS [36] and proximity [35] have been used for social context sensing. However, only outdoor activities are available because GPS is unavailable. Bluetooth proximity technology is applied for identifying users are close in terms of location. Different from the above cognition research, this approach will fully utilize seamless locating technologies on a smartphone for human behavior modeling purpose. In addition, motion states, which are usually applied for detecting personal activities [31] or some positioning purposes [33,34], will also be used for modeling human behavior in our proposed cognitive phone solution. A human behavior modeling approach named Location-Motion-Context (LoMoCo) is proposed for fusing location and motion information and inferring user's contexts. The rest of this paper is organized as follows: Section 2 provides an overview of the background of this research; Section 3 presents the proposed methods of ubiquitous positioning. We describe details of motion recognition in Section 4. Details of the LoMoCo model are represented in Section 5. Section 6 evaluates the proposed solution with experimental results. Finally, Section 7 concludes the paper and provides directions for future work.

2. Background and Related Work

This research is supported by a project titled INdoor Outdoor SEamless Navigation for Sensing Human Behavior (INOSENSE), funded by the Academy of Finland. The goal of the project is to carry out a study on sensing social context, modeling human behavior and developing a new mobile architecture for social applications. It aims to build a new analysis system by combining the latest navigation technologies and self-contained sensors to capture social contexts in real-time and use the system to study human movement and behavior in natural environments.

We abstract the social context as a Context Pyramid, as shown in Figure 2, where the raw data from diverse sensors is the foundation of the Context Pyramid. Based on the Raw Sensor Data, we can extract Physical Parameters such as position coordinates, acceleration, heading, angular velocity, velocity, and orientation. Features/Patterns of physical parameters are generated for further pattern recognition in the Simple Contextual Descriptors, which infer the simple context such as location, motion, and surroundings. Activity-Level Descriptors combine the simple contextual information into the activity level. On the top of the pyramid, Rich Context includes rich social and psychological contexts, which is ultimately expressed in natural language.

Figure 2. Context pyramid.


Meeting on Monday morning at FGI in room 309 with Jack and Jane. Meeting has been going on for 30 minutes. Jack is standing and giving presentation. Jane and I are sitting and listening with interests.
Rich Context
Having a work meeting with Navi group
Activity-Level Descriptors
Sitting, in room 309, with Jack, etc.
Simple Contextual Descriptors
Statistical properties
Features/Patterns
Para(t), e.g. position(t), acc(t), orientation(t), etc.
Physical Parameters
GPS messages, WiFi scanning info, Bluetooth queries, accelerometer output, audio stream, etc.
Raw Sensor Data

To implement the Context Pyramid, we break down the research into three modules as shown in Figure 3. In module I, we sense the social context with navigation and audio/visual sensors with output options such as position, motion, audio streams and visual contexts. The bottom three levels in the Context Pyramid are implemented in this module. Next, we analyze the social context and model human behavior in module II, which realizes the top three levels of the pyramid. Smartphone-based social applications ultimately use the human behavior models derived from module II, or the low level information from module I to demonstrate the use of sensing human behavior using indoor/outdoor seamless positioning technologies. Figure 4 gives two examples of mobile social applications based on the proposed architecture. On the left side is an application logging the location and motion of an employee in a workplace. It is an indoor social application using WiFi localization and motion sensors. On the right side is an application that interprets the commuting context of an employee, who works outdoors, based on location obtained from GPS and motion information from built-in sensors.

In order to implement cognitive applications, such as those shown in Figure 4, we combine the latest positioning technologies and smartphone sensors to capture human movements in natural environments and use the movement information to study human behavior. Three key technologies are applied in this research: ubiquitous positioning, motion recognition, and human behavior modeling, which will be described in the following sections.

Figure 3. Architecture of a social application.

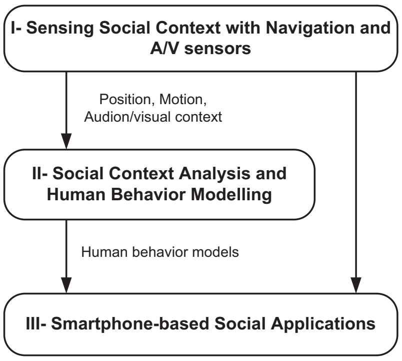
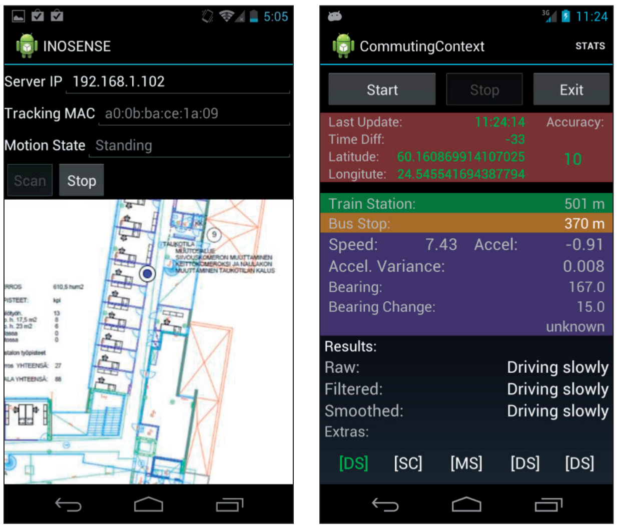

I- Sensing Social Context with Navigation and A/V sensors
Position, Motion, Audion/visual context
II- Social Context Analysis and Human Behavior Modelling
Human behavior models
III- Smartphone-based Social Applications

Figure 4. Application examples.



Server IP 192.168.1.102
Tracking MAC a0:0b:ba:ce:1a:09
Motion State Standing

Last Update: 11:24:14 Accuracy:
Time Diff: -33
Latitude: 60.160869914107025 10
Longitute: 24.545541694387794
Train Station: 501 m
Bus Stop: 370 m
Speed: 7.43 Accel: -0.91
Accel. Variance: 0.008
Bearing: 167.0
Bearing Change: 15.0
unknown
Results:
Raw: Driving slowly
Filtered: Driving slowly
Smoothed: Driving slowly
Extras:
[DS] [SC] [MS] [DS] [DS]

In order to implement cognitive applications, such as those shown in Figure 4, we combine the latest positioning technologies and smartphone sensors to capture human movements in natural environments and use the movement information to study human behavior. Three key technologies are applied in this research: ubiquitous positioning, motion recognition, and human behavior modeling, which will be described in the following sections.

3. Ubiquitous Positioning

Location as a simple contextual descriptor in the Context Pyramid is obtained using various positioning technologies. In this research, we integrate three families of smartphone-based positioning solutions, satellite-based, sensor-based, and network-based, to achieve the location capability both indoors and outdoors. For outdoors, positioning mainly relies on satellite-based technologies. Assisted with the heading and speed estimated from smartphone sensors, the satellite-based solution can also survive in the signal-deprived environments, such as urban canyons and tunnels [42]. As outdoor positioning solutions have been fully discussed in many publications [43,44], we mainly focus on indoor environments in this paper.

3.1. Indoor Outdoor Detection

Different positioning technologies are applied indoors and outdoors; therefore, to fulfill the seamless positioning function, an environment-aware approach is adopted for detecting the indoor and outdoor environments. The determination of indoor/outdoor status is performed using a combination of GPS and WiFi information. The outdoor case is recognized when the number of GPS satellites and their signal-to-noise ratio is sufficiently high. Conversely, the indoor case is recognized when the GPS signals are sufficiently weak, but WiFi signal strengths are high.

As defined in Equation (1), the probability of being present indoors combines the observations of GPS and WiFi:

$$P(X_1) = \omega \cdot P_g(X_1 \mid Y_g, Z_g) + (1-\omega) \cdot P_w(X_1 \mid Y_w, Z_w) \tag{1}$$

where $\omega \in [0,1]$ is the normalization weight of the indoor probability derived from GPS observation P_g (X_1 | Y_g, Z_g), which is estimated based on the GPS signal-to-noise ratio Y_g and the number of visible satellites Z_g. The value of ω is 0.5 by default. However, it is adjustable based on prior knowledge. For instance, when a user turns off WiFi on a smartphone, ω can be set as 1. The indoor conditional probability P_w (X_1 | Y_w, Z_w) is derived from WiFi observations including the RSSI of the strongest AP Y_w, and the number of visible APs Z_w . Probability lookup tables are generated for retrieving the probability based on the GPS and WiFi observations. The probability of being present outdoors can be calculated as follows:

$$P(X_2) = 1 - P(X_1) \tag{2}$$

Considering the battery capacity limitation of a smartphone, it is a wise option to turn off unnecessary navigation sensors or decrease the sampling rate of a sensor in the procedure of seamless positioning. For instance, we suggest using a lower WiFi scanning rate in outdoor environments and suspending GPS indoors.

3.2. Fingerprinting Based Wireless Positioning

For indoor positioning, we adopt the fingerprinting approach of WiFi positioning. Received signal strength indicators (RSSIs) are the basic observables in this approach. The process consists of a training phase and a positioning phase. During the training phase, a radio map of probability distributions of the received signal strength is constructed for the targeted area. The targeted area is divided into a grid, and the central point of each cell in the grid is referred to as a reference point. The probability distribution of the received signal strength at each reference point is represented by a Weibull function [6,9], and the parameters of the Weibull function are estimated with the limited number of training samples.

During the positioning phase, the current location is determined using the measured RSSI observations in real-time and the constructed radio map. The Bayesian theorem and Histogram Maximum Likelihood algorithm are used for positioning [45,46].

Given the RSSI measurement vector $\vec{O} = \{O_1, O_2 \ldots O_k\}$ from APs, the problem is to find the location l with the conditional probability $P(l|\vec{O})$ being maximized. Using the Bayesian theorem:

$$\arg\max_l [P(l \mid \vec{O})] = \arg\max_l [\frac{P(\vec{O} \mid l)P(l)}{P(\vec{O})}] \tag{3}$$

where $P(\vec{O}|l)$is the probability of observing RSSI vector given a location l, also known as the likelihood, $P(l)$ is the prior probability of a location l before observing , and $P(\vec{O})$ is the marginal likelihood which indicates the probability of obtaining a given RSSI measurement vector $\vec{O}$. In this study, $P(\vec{O})$ is constant for all l. Therefore, Equation (3) can be reduced to:

$$\arg\max_l [P(l \mid \vec{O})] = \arg\max_l [P(\vec{O} \mid l)P(l)] \tag{4}$$

We assume that the mobile device has equal probability to be located at each reference point, thus $P(l)$ can be considered as constant in this case. Using this assumption, Equation (4) can be simplified to:

$$\arg\max_l [P(l \mid \vec{O})] = \arg\max_l [P(\vec{O} \mid l)] \tag{5}$$

Now it becomes a problem of finding the maximum conditional probability of:

$$P(\vec{O} \mid l) = \prod_{n=1}^{k} P(O_n \mid l) \tag{6}$$

where the conditional probability $P(O_n|l)$ is derived from the RSSI distribution stored in the fingerprint database.

4. Motion Recognition

Motion, as another simple contextual descriptor in the Context Pyramid, can be detected by motion recognition methods. The possible motion states vary in different applications. Common motion states include sitting, standing, standing with tiny movements, fast walking, walking slowly, sharp turning, spot turning (a.k.a U-turning), gradient turning, running, using stairs, using an elevator, falling down, lying, and driving. The motion states can be further constrained in a particular use case. Given motion features, diverse classifiers can be applied for motion recognition. Feature selection and motion classification will be discussed in the following two subsections.

4.1. Feature Selection

This paper limits the use case to an office scenario and the applied motion states are defined as Table 1. In order to distinguish the above motion states, we currently retrieve the raw sensor data from accelerometers, gyroscope, and magnetometers built in a smartphone. The features listed in Table 2 are studied in this research. Raw data from a tri-axis accelerometer $\{a_x, a_y, a_z\}$, gyroscope $\{\omega_x, \omega_y, \omega_z\}$, and magnetometer $\{h_x, h_y, h_z\}$ of a smartphone are collected, and physical parameters such as acceleration a, linear acceleration $|a^l|$, horizontal acceleration a_h, vertical acceleration a_v, angular velocity $|\omega|$, heading h, and so on, are calculated from the raw sensor measurements.

Table 1. Motion state definition.

State	Definition
M1	Sitting.
M2	Normal walking.
M3	Fast walking.
M4	Standing, this might have some tiny movements.
M5	Sharp turning (heading change: $90° < \theta \leq 270°$).
M6	Gradient turning (heading change: $-90° < \theta \leq 90°$).

Table 2. Feature definition.

Features	Definition	Applied Physical Parameters	Raw Sensor Data
μ	Mean	$a_x, a_y, a_z, a_h, a_v, \lvert a \rvert, \lvert a^l \rvert, a_h^l, a_v^l,$ $\omega_x, \omega_y, \omega_z, \omega_h, \omega_v, \lvert \omega \rvert, h_x, h_y, h_z, h.$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, h_x, h_y, h_z.$
σ^2	Variance	$a_x, a_y, a_z, a_h, a_v, \lvert a \rvert, \lvert a^l \rvert, a_h^l, a_v^l,$ $\omega_x, \omega_y, \omega_z, \omega_h, \omega_v, \lvert \omega \rvert, h_x, h_y, h_z, h.$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, h_x, h_y, h_z.$
m	Median	$a_x, a_y, a_z, a_h, a_v, \lvert a \rvert, \lvert a^l \rvert, a_h^l, a_v^l,$ $\omega_x, \omega_y, \omega_z, \omega_h, \omega_v, \lvert \omega \rvert, h_x, h_y, h_z, h.$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, h_x, h_y, h_z.$
$IQR=Q_3\text{-}Q_1$	Interquartile range (IQR)	$a_x, a_y, a_z, a_h, a_v, \lvert a \rvert, \lvert a^l \rvert, a_h^l, a_v^l,$ $\omega_x, \omega_y, \omega_z, \omega_h, \omega_v, \lvert \omega \rvert, h_x, h_y, h_z, h.$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, h_x, h_y, h_z.$
$y_{skewness} = \dfrac{E(x-\mu)^3}{\sigma^3}$	Skewness	$a_x, a_y, a_z, a_h, a_v, \lvert a \rvert, \lvert a^l \rvert, a_h^l, a_v^l,$ $\omega_x, \omega_y, \omega_z, \omega_h, \omega_v, \lvert \omega \rvert, h_x, h_y, h_z, h.$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, h_x, h_y, h_z.$
$y_{kurtosis} = \dfrac{E(x-\mu)^4}{\sigma^4}$	Kurtosis	$a_x, a_y, a_z, a_h, a_v, \lvert a \rvert, \lvert a^l \rvert, a_h^l, a_v^l,$ $\omega_x, \omega_y, \omega_z, \omega_h, \omega_v, \lvert \omega \rvert, h_x, h_y, h_z, h.$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, h_x, h_y, h_z.$
$y_{diff} = \lvert y_t - y_{t-1} \rvert$	Difference of two successive measurements	h	h_x, h_y, h_z
f_{1st}	1st dominant frequency	$\lvert a \rvert, \lvert \omega \rvert$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z.$
f_{2nd}	2nd dominant frequency	$\lvert a \rvert, \lvert \omega \rvert$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z.$
$A_{f_{1st}}$	Amplitude of the 1st dominant frequency	$\lvert a \rvert, \lvert \omega \rvert$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z.$
$A_{f_{2nd}}$	Amplitude of the 2nd dominant frequency	$\lvert a \rvert, \lvert \omega \rvert$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z.$
$A_{scale} = \dfrac{A_{f_{1st}}}{A_{f_{2nd}}}$	Amplitude scale of two dominant frequencies	$\lvert a \rvert, \lvert \omega \rvert$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z.$
$A_{diff} = \lvert A_{f_{1st}} - A_{f_{2nd}} \rvert$	Difference between two dominant frequencies	$\lvert a \rvert, \lvert \omega \rvert$	$a_x, a_y, a_z, \omega_x, \omega_y, \omega_z.$

Thirteen features from the time domain and frequency domain are applied to the above physical parameters. The sequential forward selection (SFS) algorithm [47–49] is adopted for feature selection, and Decision Tree (DT), Linear Discriminant Analysis (LDA), and LS-SVM (Least Square-Support Vector Machines) are used as classifiers in the criterion function of SFS. The subset of features $\{\sigma^2_{a^l_h}, \sigma^2_{a^l_v}, \mu_{\omega_x}, \mu_{\omega_y}, \mu_{\omega_z}, \mu_{|\omega|}\}$ is selected for use in a SVM classifier, which achieves the highest accuracy rate of 92.9%. The algorithm details of LS-SVM classification are described in the below subsection.

4.2. Classification

A supervised learning method is adopted for motion recognition. Classification algorithms such as DT, LDA, and LS-SVM are investigated in this research. After comparing these classifiers, LS-SVM is finally applied in this work because of the high accuracy of the recognition rate. Using a least squares loss function and replacing the inequality constraints with equality constraints, LS-SVM tackles linear systems instead of solving convex optimization problems in standard support vector machines (SVM), which reduces the complexity of computation [50]. In the training phase, the LS-SVM classifier constructs a hyperplane in a high-dimensional space aiming to separate the data according to the different classes. This data separation should occur in such a way that the hyperplane has the largest distance to the nearest training data points of any class. These particular training data points define the so-called margin [51,52]. These parameters can be found by solving the following optimization problem having a quadratic cost function and equality constraints:

$$\arg\min J(\omega, e) = \arg\min\left(\frac{1}{2}\omega^T\omega + \frac{1}{2}\gamma\sum_{i=1}^{N} e_i\right) \tag{7}$$

subject to [51]:

$$y_i(\omega^T\phi(x_i) + b) \geq 1 - e_i, i = 1, \ldots, N \tag{8}$$

with $e = [e_1 \cdots e_N]^T$ being a vector of error variables to tolerate misclassifications, sign function $y \in \{-1,+1\}$, $\varphi(\bullet)$: $\mathbb{R}^d \rightarrow \mathbb{R}^{d_h}$ the mapping from the input space into a high-dimensional feature space of dimension d_h, ω a vector of the same dimension as $\varphi(\bullet)$, γ is a positive regularization parameter, determining the trade-off between the margin size maximization and the training error minimization. The term b is the bias. In this equation, the standard SVM formulation is modified using a least squares loss function with error variables e_i and replacing the inequality constraints with equality constraints [51,52]. The Lagrangian for the problem in Equations (7) and (8) is [15,52]:

$$L(\omega, b, e, \alpha) = J(\omega, e) - \sum_{i=1}^{N}\alpha_i\left(\ y_i\left[\omega^T\phi(x_i) + b\right] - 1 + e_i\right) \tag{9}$$

where $\alpha \in \mathbb{R}$ are the Lagrange multipliers, also support values.

Taking the conditions for optimality, we set:

$$\begin{cases} \dfrac{\partial L}{\partial \omega} = 0 \rightarrow \omega = \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i), \\ \dfrac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0, \\ \dfrac{\partial L}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma e_i, \\ \dfrac{\partial L}{\partial \alpha_i} = 0 \rightarrow \ y_i \left[\omega^T \varphi(x_i) + b\right] - 1 + e_i = 0, i = 1, ..., N. \end{cases} \tag{10}$$

Whereas the primal problem is expressed in terms of the feature map, the linear optimization problem in the dual space is expressed in terms of the kernel function [51,52]:

$$\begin{pmatrix} 0 & y^T \\ y & \Omega + \dfrac{1}{\gamma} I_n \end{pmatrix} \begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ 1_n \end{pmatrix} \tag{11}$$

where $y = [y_1 \cdots y_N]^T$, $\alpha = [\alpha_1 \cdots \alpha_N]^T$, $1_n = [1 \cdots 1]^T_{1 \times N}$ and $\Omega \in \mathbb{R}^{N \times N}$ is a matrix with elements $\Omega_{ij} = y_i y_j\ \varphi(x_i)^T \varphi(x_j)$, with $i, j = 1, ..., N$. Given an input vector x, the resulting LS-SVM classifier in the dual space is [50]:

$$y(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b \right) \tag{12}$$

where $K(x,x_i) = \varphi(x)^T \varphi(x_i)$ is a positive definite kernel matrix. The support values α_i are proportional to the error of the corresponding training data points. This implies that usually every training data point is a support vector and no sparseness property remains in the LS-SVM formulation. Note that high support values introduce a high contribution of the data point to the decision boundary [51]. The choice of the regularization parameter and the kernel hyperparameter δ in case of an RBF kernel, is out of the scope for discussion in this paper. Hospodar gives an example of the kernel parameters selection in [50].

5. Human Behavior Modeling Based on LoMoCo Model

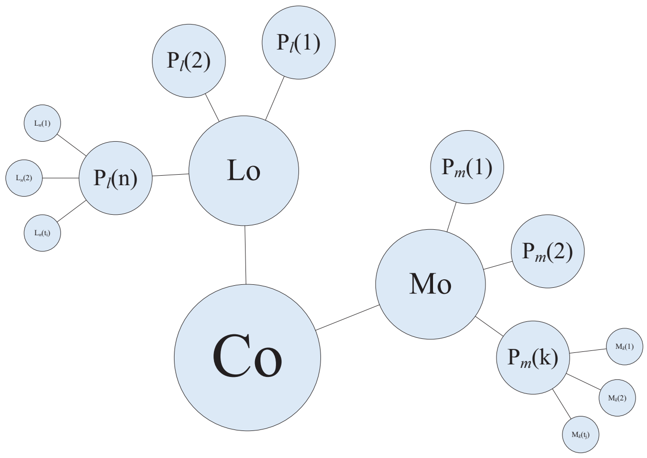
Modeling human behavior has great complexity, due to the wide range of activities that humans can undertake and due to the difficulties in systematically classifying these activities [15]. The approach taken in this research is to simplify the human behavior modeling using a Location-Motion-Context (LoMoCo) model which combines personal location information and motion states to infer a corresponding context based on Bayesian reasoning.

5.1. LoMoCo Model

Given a specific context, a person always performs movements with some particular patterns. For instance, an employee usually sits in a break room while taking a break. He/she most likely stands in front of a coffee machine and shortly walks back to the office in a context of fetching coffee. In this research, we determine a context based on a LoMoCo model shown in Figure 5. In the LoMoCo model, a context (Co) is represented by location patterns (Lo) and motion patterns (Mo). Assuming that all the target contexts occur in n significant locations, we denote $L_n(t_i)$ as a context that occurs at

L_n at the time epoch t_i. $P_l(n)$ denotes the density of the context that occurs at the location n. A location pattern (Lo) consists of the probabilities of all the possible locations. Similarly, motion patterns (Mo) include a set of probabilities for each possible motion state. $M_k(t_j)$ indicates that a context includes a motion state M_k of the time epoch t_j.

Figure 5. LoMoCo Model.



5.2. Bayes Inferring

In order to infer the context, the LoMoCo model in this paper is represented using Bayesian reasoning, which can not only determine the context but also provide with the probability of a determined class. The classifier of LoMoCo model is designed based on the Bayes rule and trained by supervised learning. In the training phase, we wish to approximate an unknown target function $P(Y|X)$, where Y is the context predefined, and $X=\{x_1,x_2 \dots x_k\}$ is a vector containing observed features which are all conditionally independent of one another, given Y. Applying Bayes' rule, we have:

$$P(Y=y_i \mid X) = \frac{P(X \mid Y=y_i)P(Y=y_i)}{\sum_{j=1}^{N} P(X \mid Y=y_j)P(Y=y_j)} \tag{13}$$

Further, we get:

$$P(Y=y_i \mid x_1 \dots x_k) = \frac{\prod_{n=1}^{k} P(x_n \mid Y=y_i)P(Y=y_i)}{\sum_{j=1}^{N}\prod_{n=1}^{k} P(x_n \mid Y=y_i)P(Y=y_j)} \tag{14}$$

where y_i denotes the *i*th possible context for Y, and the summation in the denominator is over all legal values of the context variable Y. In the training phase, we use the training data to estimate $P(X|Y=y_i)$ and $P(Y=y_i)$ which are utilized to determine $P(Y=y_i|X=X_z)$ for any new vector instance X_z. For the classification case, we are only interested in the most probable value of Y, so the problem becomes:

$$\arg\max_{y_i}[P(Y=y_i \mid x_1 \ldots x_k)] = \arg\max_{y_i}\left[\frac{\prod_{n=1}^{k} P(x_n \mid Y=y_i)P(Y=y_i)}{\sum_{j=1}^{N}\prod_{n=1}^{k} P(x_n \mid Y=y_i)P(Y=y_j)}\right] \tag{15}$$

which simplifies to the following because the denominator does not depend on a context y_i :

$$\arg\max_{y_i}[P(Y=y_i \mid x_1 \ldots x_k)] = \arg\max_{y_i}\left[\prod_{n=1}^{k} P(x_n \mid Y=y_i)P(Y=y_i)\right] \tag{16}$$

In the LoMoCo model, the feature vector is suggested using observations with location and motion state combined, where $X = \{P_l(1), P_l(2) \ldots P_l(n), P_m(1), P_m(2) \ldots P_m(k)\}$. In the case without motion or location observations, feature vector can be only location patterns where $X = \{P_l(1), P_l(2) \ldots P_l(n)\}$ or motion patterns where $X = \{P_m(1), P_m(2) \ldots P_m(k)\}$. $P_l(n)$ and $P_m(k)$ are respectively calculated as:

$$P_l(n) = \frac{\# D\{L(t_i) = n\}}{|D|}, 0 \le i \le D \tag{17}$$

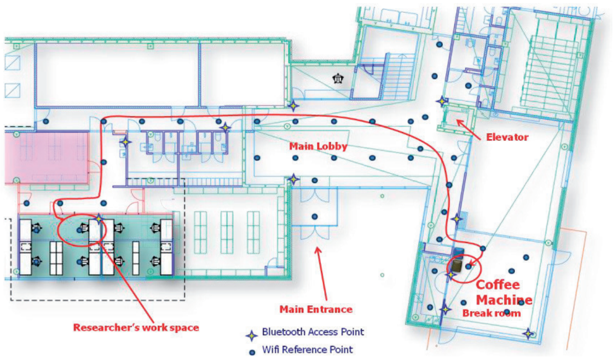
$$P_m(k) = \frac{\# D\{M(t_j) = k\}}{|D|}, 0 \le j \le D \tag{18}$$

where the $\# D\{c\}$operator returns the number of samples in the set D that satisfy the condition c, and $|D|$ is the total number of samples in the set D.

6. Experimental Results

In order to demonstrate the proposed approach, we set up a test environment on the first floor of the Finnish Geodetic Institute (FGI), as shown in Figure 6. Positioning tests and motion recognition tests were performed in this environment to validate the positioning algorithms and motion recognition methods proposed for determining the Simple Contextual Descriptors level in Figure 2. Then, an employee-centric experiment was designed to verify whether we can achieve the Activity-Level Descriptors layer of the pyramid in Figure 2 using the LoMoCo model. Taking into account the battery capacity limitation of a smartphone, we conducted a battery drain test at last.

Figure 6. Test environment.



6.1. Positioning Results

This section presents the results of the above mentioned ubiquitous positioning technologies. Because outdoor positioning performance using GPS has been thoroughly discussed in many publications, for instance [43], we will mainly focus on the indoor positioning performance in this section. The test was conducted in the FGI office where forty WiFi access points are distributed among all the three floors and thirty of them might be detected on the first floor. Among all reference points, at least one and at most fourteen access points can be simultaneously observed. An Android WiFi fingerprint collection application was developed on a Samsung Galaxy Nexus. Using that application, totally 43 reference points were selected for generating the radio map for the test area. The distance between two adjective reference points is around 3–5 meters. Taking account into the factors which might affect on RSSI measurements such as the variance of RSSI observations [45], inferences from other radio systems [53], and the disturbance of human body, sixty samples were collected for each reference point from four directions during approximately 1 minute. Each direction includes about 15 samples. During the positioning tests, a tester randomly walked throughout the test zone with a built-in audio recorder in the same phone to provide a positioning reference: the tester made a mark by speaking out the name of a reference point while passing by it. In total 560 samples were collected for verifying the positioning accuracy. The entire test area is classified into three types of space: open space, corridor, and semi-open space. Open space is a large space without obstacles, such as the main lobby and break room shown in Figure 6. The corridor environment refers to a narrow hallway where a person usually is oriented in one of only two directions. Semi-open space is an open space with some obstacles, such as furniture or office partitions.

Finally, the statistical analysis results are listed in Table 3. Positioning results indicate 1.9 meters errors in corridors, 2.7 meters errors in the semi-open space, and 3.5 meters errors in the main lobby and break room. The above accuracies are high enough for room-level activity recognition.

Table 3. Positioning results (Unit: Meter).

Environment	**Open Space**	**Corridors**	**Semi-open**
Mean error	3.5	1.9	2.7
RMSE	4.5	3.0	3.3
Maximum error	9.5	6.0	7.0
Minimum error	0	0	0

6.2. Motion Recognition Results

The proposed motion recognition method is verified by a set of dedicated tests. Note that a phone can be placed at different positions on a user's body, which impacts the sensor data patterns. In order to reduce the complexity, the tester always kept the phone in his pants pocket and the orientation of the phone was as shown in Figure 7. Provided with a sensor data collection application (developed by the authors), four testers were involved in the sensor data collection during five days. In the FGI office building, each tester performed six motion states which are listed in Table 1. For each tester, more than 1,200 samples were collected. Thirteen types of features were extracted from the built-in accelerometers, gyroscope, and magnetometers in a smartphone.

Figure 7. The phone in pants pocket.

Six motion states are detected by a Least Squares Support Vector Machines (LS-SVM) classification algorithm. The results indicate that the motion states are recognized with an accuracy rate of up to 92.9% for the test cases employed in this study. The confusion matrix in Table 4 shows that major confusions existed between sharp turning (M5) and gradient turning (M6) because these two motion states are processes depending on both the heading change and heading change rate. For example, if a user's heading changed 180 degrees in a second, the corresponding motion state will be determined as sharp turning. However, if the user changes his/her heading in more than two seconds, the motion state might be considered as gradient turning.

Table 4. Confusion matrix for the motion recognition from LS-SVM classifier (Unit: %).

	M1	**M2**	**M3**	**M4**	**M5**	**M6**
M1	99.5	0.5	0	0	0	0
M2	0	96.0	4.0	0	0	0
M3	0	0	100.0	0	0	0
M4	0	0	0	100.0	0	0
M5	0	0	0	16.7	64.8	18.5
M6	0	0	0	1.9	31.5	66.7

We also found that there are some misunderstandings between standing (M4) and sharp turning (M5). The reason is related to the training phase, where testers started a sharp turn while standing stationary, and also finished the sharp turning with a standing state. Thus, it was hard to label the sharp turning samples from the entire training data to only include the sharp turn time segment. As a result, even though a motion data set is labeled as a sharp turning state, it could include some standing states.

Despite the confusion in the turning states, the other motion states, such as sitting, normal walking, fast walking, standing, achieve a perfect success rate in the tests, and therefore can be used effectively for context determination.

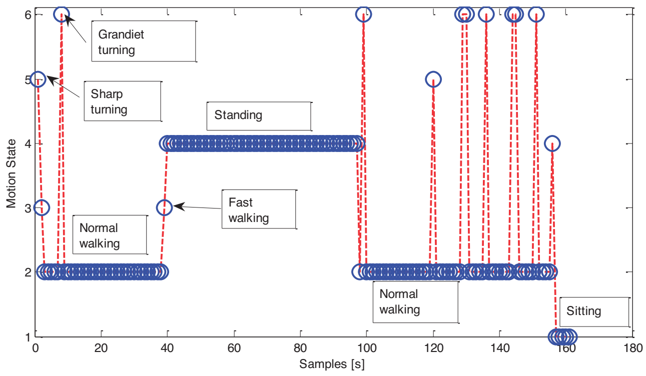
6.3. Human Behavior Modeling Results

Activity-Level Descriptors in the Context Pyramid vary because the activity definitions and scenarios are diverse. Each different activity has its own features. As a result, it is very difficult to develop a universal model to classify activities in the Activity-Level Descriptors layer. However, location and motion are two fundamental elements of human behavior, which can be used to infer some human activities. For instance, sitting in an office might be translated as working, standing beside of a coffee machine could be considered as fetching a drink. Therefore, we proposed the LoMoCo model in Section 5. In order to demonstrate the usability of this model, we narrow down the scope of human activities to an employee's behavior with dedicated contexts in a workplace scenario as shown in Figure 6. The goal of the tests is to determine the purpose of an employee using the break room after he/she left his/her office. To simplify the problem, we define six contexts/activities in the Activity-Level Descriptors:

1. C1: fetching coffee. The tester leaves his/her office and travels through the corridors and main lobby. Then, he/she fetches coffee from a coffee machine located in the break room. Finally, he/she returns to his/her office as long as his/her coffee is ready.
2. C2: fetching water. The tester leaves his/her office and travels through the corridors, main lobby, and break room. Then, he/she fetches water from a dispenser located in the kitchen. Finally, he/she returns to his/her office as long as his/her water is ready.
3. C3: taking a break. The tester leaves his/her office and travels through the corridors and main lobby. Then he/she sits in the break room for a while. Finally, he/she returns to his/her office after a break.
4. C4: having lunch. The tester leaves his/her office and travels through the corridors, main lobby, and break room. Then, he/she prepares his/her food in the kitchen and has his/her lunch in the break room. Finally, he/she returns to his/her office after lunch.
5. C5: working. The tester sits in his/her office in most of the time. However, this context might also include some brief standing, turning, walking motion states.
6. C6: undefined context. Contexts which are not defined in the above are classified as unknown context.

Figure 8 gives an example of the motion states sequence occurring in a fetching coffee context. The tester firstly left the office while performing some turnings, and walked to the coffee machine. He/she stood in front of the coffee machine while fetching coffee, and walked back to his/her office after the coffee was ready. The example ended up with the tester sitting back in the office.

Figure 8. Motion states in fetching coffee context (C1).



In this test scenario, we only require room-level accuracy location. Therefore, we use the ID of a location to where the estimated reference point belongs. We organize the reference points in the test area, as shown in Figure 9, into significant locations as shown in Table 5:

Figure 9. Graph of reference points.

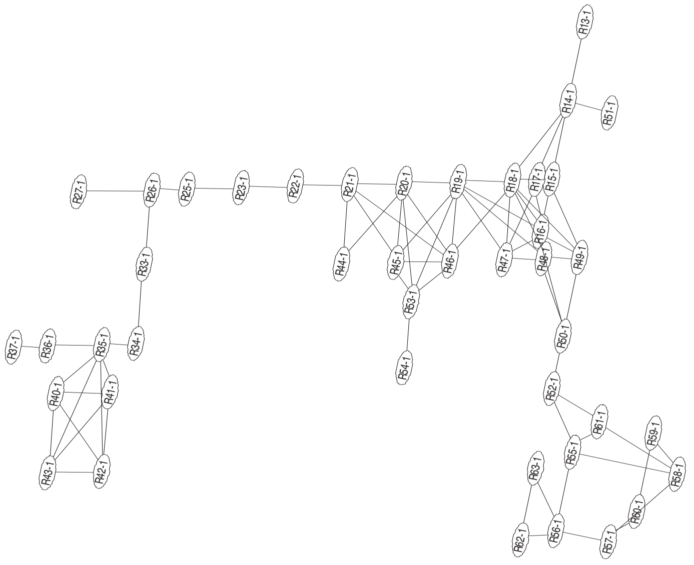


Table 5. Location definition.

ID	Location	Reference Points ID
L1	Office	R34-1~R37-1, R40-1~R43-1
L2	Corridors	R13-1~R14-1, R22-1~R27-1, R33-1, R49-1~R52-1
L3	Main lobby	R15-1~R21-1, R44-1~R48-1
L4	Break room	R55-1~R61-1
L5	Kitchen	R62-1~R63-1

Using Equation (17) in the LoMoCo model, the probability of each location is calculated for each context/activity. On the other hand, the probability of each motion state is also counted by employing Equation (18). During the training phase, twenty context samples covering four samples for each context except C6 were collected for the model training. In the testing phase, four testers performed sixty-seven contexts including fourteen C1, fifteen C2, ten C4, eight C3, and fifteen C5 context samples respectively, and five abnormal contexts which are not predefined in the LoMoCo model. The abnormal contexts included two contexts of fetching papers from a printer, two contexts of taking break in the lobby and the last one was using toilet. By applying the proposed LoMoCo model, we obtain the results as follows. Tables 6 and 7 show the results if only location features or motion features, respectively, are applied in the LoMoCo model. In the case of only location features applied, 85.5% of contexts can be correctly detected. 28.6% and 21.4% C1 contexts are mistaken as C3 and C2 respectively because those contexts have similar location patterns. Furthermore, as shown in Table 7, similar motion patterns introduce confusions between C1 and C2, C3 and C4, C3 and C5 as well. If we simultaneously take location features and motion features into account, as shown in Table 8, 90.3% of all the contexts can be correctly recognized. Abnormal contexts are classified as similar predefined contexts, for instance, the contexts of taking break in the lobby are recognized as C3, fetching a paper from a printer is labeled as fetching water or coffee. Using a toilet which is close to the office is labeled as the context of working.

Table 6. Confusion matrix for the context recognition with location features (Unit: %).

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
C1	50	21.4	28.6	0	0	0
C2	0	100	0	0	0	0
C3	10.0	0	90.0	0	0	0
C4	0	0	12.5	87.5	0	0
C5	0	0	0	0	100	0
C6	0	100	0	0	0	0

Table 7. Confusion matrix for the context recognition with motion features (Unit: %).

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
C1	28.6	64.3	7.1	0	0	0
C2	0	100	0	0	0	0
C3	0	0	100	0	0	0
C4	0	0	12.5	87.5	0	0
C5	0	0	6.7	0	93.3	0
C6	0	20.0	40.0	0	40.0	0

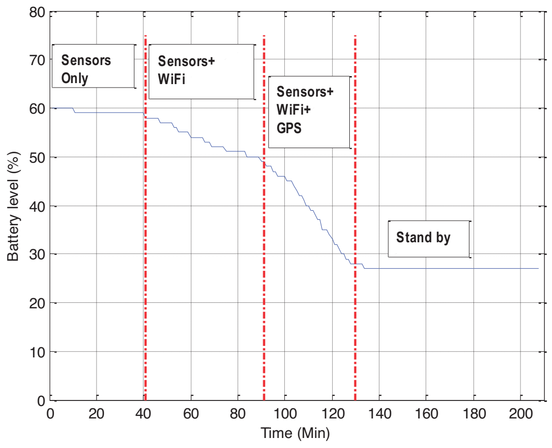
Table 8. Confusion matrix for LoMoCo model (Unit: %).

	C1	**C2**	**C3**	**C4**	**C5**	**C6**
C1	64.3	35.7	0	0	0	0
C2	0	100	0	0	0	0
C3	0	0	100	0	0	0
C4	0	0	12.5	87.5	0	0
C5	0	0	0	0	100	0
C6	20.0	20.0	40.0	0	20.0	0

6.4. Battery Drain Analyzing

Considering that the battery capacity is still limited, we conducted a 3.5 hours test to analyze the battery drain on a Samsung Nexus phone equipped with a 1,750 mAh Li-ion battery. A smartphone-based cognitive application as shown in the left image of Figure 4, which sampled the motion sensors around 90 Hz and scanned WiFi and GPS at about 1 Hz in the Raw Sensor Data layer of the Context Pyramid in Figure 2, was used for testing. The smartphone screen was kept off during the test. As shown in Figure 10, we started the test when 60% battery was left.

Figure 10. Battery drain on a smartphone.



After 41 minutes with only motion sensors enabled, 2% battery was consumed. The battery was drained even faster while WiFi scanning was on. Figure 10 indicates 10% battery used in 50 minutes. The most energy-consuming case was turning on motion sensors, WiFi, and GPS insight of a smartphone simultaneously and the battery drain rate was 27.5%·h^{-1} in such circumstance. Therefore, GPS is suggested turning off or lowering the sampling rate indoors. With a fully charged battery and without any extra applications running on a smartphone, the cognitive application would constantly work 8.3 hours if only motion sensors and WiFi are turned on.

7. Conclusions

This research investigates context sensing, modeling human behavior, and developing a new architecture for cognitive phone platform. We combine the latest positioning technologies and sensors to capture human movements in natural environments and use the movements to study human behavior. Contexts in this research are abstracted as a Context Pyramid which includes six levels: Raw Sensor Data, Physical Parameter, Features/Patterns, Simple Contextual Descriptors, Activity-Level Descriptors, and Rich Context. To achieve understanding of the Context Pyramid on a cognitive phone, three key technologies are implemented: ubiquitous positioning, motion recognition, and human behavior modeling. Preliminary tests indicate that we have successfully achieved the Activity-Level Descriptors level with a Location-Motion-Context (LoMoCo) model with a correct rate of 90.3%. Location accuracy of the proposed solution is up to 1.9 meters errors in corridor environments and 3.5 meters errors in open space. Test results also indicate that the motion states are recognized with an accuracy rate of up to 92.9%.

Despite the fact that the motion recognition solution proposed in this paper provides a high correct motion recognition rate, the motion definition and feature selection vary from case to case. For instance, even though it is easy to confuse sharp turning with gradient turning in motion recognition, it will not effect on the classification if we merge them as one turning state in some cases. Therefore, in the future, we will investigate the motion and feature selections to find out the most effective motion states definition and features for context classification. Undefined contexts are not able to handle in the proposed LoMoCo model yet. Therefore, we will improve the model to detect abnormal behaviors. In the current stage, we successfully reach the Activity-Level Descriptors for individuals. Social activities with a group of people will be studied in the near future. Additionally, in the next step of this research work, we will focus on more complex human behavior modeling to reach the Rich Context level. The psychological state and social media context will be considered in future work.

Acknowledgments

This work is a part of the INdoor Outdoor Seamless Navigation for SEnsing Human Behavior (INOSENSE) project, funded by the Academy of Finland. The authors would like to thank Ahsan Feroz and Tuomas Keränen from Aalto University, Finland, and Sampsa Salminen and Jari Takatalo from University of Helsinki for helping to collect the data for this research.

References

1. Hu, D.H.; Zhang, X.X.; Yin, J.; Zheng, V.W.; Yang, Q. Abnormal Activity Recognition Based on Hdp-Hmm Models. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, California, CA, USA, 11–17 July 2009; pp. 1715–1720.
2. Pei, L.; Chen, R.; Chen, Y.; Leppäkoski, H.; Perttula, A. Indoor/Outdoor Seamless Positioning Technologies Integrated on Smart Phone. In *Proceedings of the International Conference on Advances in Satellite and Space Communications*, Colmar, France, 20–25 July 2009; pp. 141–145.
3. Kraemer, I.; Eissfeller, B. A-GNSS: A different approach. *Inside GNSS* **2009**, *4*, 52–61.
4. Syrjärinne, J. Studies on Modern Techniques for Personal Positioning. Ph.D. Thesis, Tampere University of Technology, Tampere, Finland, March 2001.
5. Laura, K.; Perala, T.; Piché, R. Indoor Positioning Using Wlan Coverage Area Estimates. In *IEEE Proceedings of International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zurich, Switzerland, 15–17 September 2010; pp. 1–7.
6. Pei, L.; Chen, R.; Liu, J.; Kuusniemi, H.; Tenhunen, T.; Chen, Y. Using inquiry-based Bluetooth RSSI probability distributions for indoor positioning. *J. Glob. Position. Syst.* **2010**, *9*, 122–130.
7. Priyantha, N.B.; Chakraborty, A.; Balakrishnan, H. The Cricket Location-Support System. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, Boston, MA, USA, 6–11 August 2000; pp. 32–43.
8. Bahl, P.; Padmanabhan, V.N. Radar: An In-Building RF Based User Location and Tracking System. In *Proceedings of Infocom—Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Tel-Aviv, Israel, 26–30 March 2000; pp. 775–784.
9. Pei, L.; Chen, R.; Liu, J.; Tenhunen, T.; Kuusniemi, H.; Chen, Y. Inquiry-Based Bluetooth Indoor Positioning via RSSI Probability Distributions. In *Proceedings of the Second International Conference on Advances in Satellite and Space Communications (SPACOMM 2010)*, Athens, Greece, 13–19 June 2010; pp. 151–156.
10. Gomes, G.; Sarmento, H. Indoor Location System Using ZigBee Technology. In *Proceedings of Third International Conference on Sensor Technologies and Applications*, Athens/Glyfada, Greece, 18–23 June 2009; pp. 152–157.

11. Chen, L.; Pei, L.; Kuusniemi, H.; Chen, Y.; Kröger, T.; Chen, R. Bayesian fusion for indoor positioning using bluetooth fingerprints. *Wirel. Pers. Commun.* **2012**, *67*, 1–11.
12. Ni, L.M.; Liu, Y.; Lau, Y.C.; Patil, A.P. Landmarc: Indoor location sensing using active RFID. *Wirel. Netw.* **2004**, *10*, 701–710.
13. Pei, L.; Chen, R.; Liu, J.; Chen, W.; Kuusniemi, H.; Tenhunen, T.; Kröger, T.; Chen, Y.; Leppäkoski, H.; Takala, J. Motion Recognition Assisted Indoor Wireless Navigation on a Mobile Phone. In *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation*, Portland, OR, USA, 21–24 September 2010; pp. 3366–3375.
14. Liu, J.; Chen, R.; Chen, Y.; Pei, L.; Chen, L. iParking: An intelligent indoor location-based smartphone parking service. *Sensors* **2012**, *12*, 14612–14629.
15. Pei, L.; Liu, J.; Guinness, R.; Chen, Y.; Kuusniemi, H.; Chen, R. Using LS-SVM based motion recognition for smartphone indoor wireless positioning. *Sensors* **2012**, *12*, 6155–6175.
16. Ruotsalainen, L.; Kuusniemi, H.; Chen, R. Visual-aided *Two-dimensional pedestrian indoor navigation with a smartphone*. *J. Glob. Position. Syst.* **2011**, *10*, 11–18.
17. Mulloni, A.; Wagner, D.; Schmalstieg, D.; Barakonyi, I. Indoor positioning and navigation with camera phones. *Pervasive Comput.* **2009**, *8*, 22–31.
18. IndoorAtlas Ltd. Oulu, Finland. Available online: http://www.indooratlas.com (accessed on 22 October 2012).
19. Liu, J.; Chen, R.; Pei, L.; Chen, W.; Tenhunen, T.; Kuusniem, H.; Kröger, T.; Chen, Y. Accelerometer Assisted Wireless Signals Robust Positioning Based on Hidden Markov Model. In *Proceedings of the IEEE/ION Position, Location and Navigation Symposium (PLANS) 2010*, Indian Wells, CA, USA, 3–6 May 2010; pp. 488–497.
20. Liu, J.; Chen, R.; Pei, L.; Guinness, R.; Kuusniemi, H. Hybrid smartphone indoor positioning solution for mobile LBS. *Sensors* **2012**, *12*, 17208–17233.
21. Kuusniemi, H.; Liu, J.; Pei, L.; Chen, Y.; Chen, L.; Chen, R. Reliability considerations of multi-sensor multi-network pedestrian navigation. *Radar Sonar Navig. IET* **2012**, *6*, 157–164.
22. Poppe, R. Vision-based human motion analysis: An overview. *Comput. Vis. Image Understand* **2007**, *108*, 4–18.
23. Chung, T.-Y.; Chen, Y.-M.; Hsu, C.-H. Adaptive momentum-based motion detection approach and its application on handoff in wireless networks. *Sensors* **2009**, *9*, 5715–5739.
24. Fong, D.T.-P.; Chan, Y.-Y. The use of wearable inertial motion sensors in human lower limb biomechanics studies: A systematic review. *Sensors* **2010**, *10*, 11556–11565.
25. Yang, C.-C.; Hsu, Y.-L. A Review of accelerometry-based wearable motion detectors for physical activity monitoring. *Sensors* **2010**, *10*, 7772–7788.
26. Zeng, H.; Zhao, Y. Sensing movement: Microsensors for body motion measurement. *Sensors* **2011**, *11*, 638–660.
27. Musleh, B.; García, F.; Otamendi, J.; Armingol, J.M.; De la Escalera, A. Identifying and tracking pedestrians based on sensor fusion and motion stability predictions. *Sensors* **2010**, *10*, 8028–8053.
28. Kavanagh, J.J.; Menz, H.B. Accelerometry: A technique for quantifying movement patterns during walking. *Gait Posture* **2008**, *28*, 1–15.
29. Baek, J.; Lee, G.; Park, W.; Yun, B. Accelerometer signal processing for user activity detection. *Knowl. Based Intell. Inform. Eng. Syst.* **2004**, *3215*, 610–617.

30. Chen, W.; Fu, Z.; Chen, R.; Chen, Y.; Andrei, O.; Kröger, T; Wang, J. An Integrated GPS and Multi-Sensor Pedestrian Positioning System for 3D Urban Navigation. In *Proceedings of Joint Urban Remote Sensing Event*, Shanghai, China, 20–22 May 2009; pp. 1–6.
31. Yang, J. Toward Physical Activity Diary: Motion Recognition Using Simple Acceleration Features with Mobile Phones. In *Proceedings of the 1st International Workshop on Interactive Multimedia for Consumer Electronics (IMCE)*, Beijing, China, 19–23 October 2009; pp. 1–10.
32. Frank, K.; Nadales, M.J.V.; Robertson, P.; Angermann, M. Reliable Real-Time Recognition of Motion Related Human Activities Using MEMS Inertial Sensors. In *Proceedings of the 23rd International Technical Meeting of the Satellite Division of the Institute of Navigation*, Portland, OR, USA, 21–24 September 2010; pp. 2906–2912.
33. Susi, M.; Borio, D.; Lachapelle, G. Accelerometer Signal Features and Classification Algorithms for Positioning Applications. In *Proceedings of International Technical Meeting of the Institute of Navigation*, San Diego, CA, USA, 24–26 January 2011.
34. Pei, L.; Chen, R.; Liu, J.; Kuusniemi, H.; Chen, Y.; Tenhunen, T. Using Motion-Awareness for the 3D Indoor Personal Navigation on a Smartphone. In *Proceedings of the 24rd International Technical Meeting of the Satellite Division of the Institute of Navigation*, Portland, OR, USA, 19–23 September 2011; pp. 2906–2912.
35. Eagle, N.; Pentland, A. Reality mining: Sensing complex social systems. *Pers. Ubiquitous Comput.* **2006**, *10*, 255–268.
36. Adams, B.; Phung, D.; Venkatesh, S. Sensing and using social context. *ACM Trans. Multimed. Comput. Commun. Appl.* **2008**, *5*, 1–27.
37. Anderson, I.; Maitland, J.; Sherwood, S.; Barkhuus, L.; Chalmers, M.; Hall, M.; Muller, H. Shakra: Tracking and sharing daily activity levels with unaugmented mobile phones. *Mob. Netw. Appl.* **2007**, *12*, 185–199.
38. Choudhury, T.; Pentland, A. Sensing and Modeling Human Networks Using the Sociometer. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers (ISWC2003)*, Washington D.C., WA, US, October 2003; pp. 216–222.
39. Choudhury, T.; Philipose, M.; Wyatt, D.; Lester, J. Towards activity databases: using sensors and statistical models to summarize people's lives. *Data Eng. Bull.* **2006**, *29*, 49–58.
40. Lane, N.D.; Miluzzo, E.; Lu, H.; Peebles, D.; Choudhury, T.; Campbell, A.T. A survey of mobile phone sensing. *Commun. Mag.* **2010**, *48*, 140–150.
41. Campbell, A.; Choudhury, T. From smart to cognitive phones. *Pervasive Comput.* **2012**, *11*, 7–11.
42. Pei, L.; Chen, R.; Liu, J.; Liu, Z.; Kuusniemi, H.; Chen, Y.; Zhu, L. Sensor Assisted 3D Personal Navigation on A Smart Phone in GPS Degraded Environments. In *Proceedings of 2011 19th International Conference on Geoinformatics*, Shanghai, China, 24–26 June 2011; pp. 1–6.
43. Kaplan, E.D.; Hegarty, C.J. *Understanding GPS: Principles and Applications*; Artech House Publishers: Norwood, MA, USA, 2006.
44. Li, B.; Quader, I.J.; Dempster, A.G. On outdoor positioning with Wi-Fi. *J. Glob. Position Syst.* **2008**, *7*, 18–26.

45. Youssef, M.; Agrawala, A.; Shankar, A.U. WLAN Location Determination via Clustering and Probability Distributions. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, Fort-Worth, TX, USA, 23–26 March 2003; pp. 143–150.
46. Roos, T.; Myllymaki, P.; Tirri, H.; Misikangas, P.; Sievänen, J. A probabilistic approach to WLAN user location estimation. *Int. J. Wirel. Inform. Netw*. **2002**, *9*, 155–164.
47. Guyon, I.; Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
48. Hall, M.A. Correlation-Based Feature Selection for Machine Learning. Ph.D. Thesis, The University of Waikato, Hamilton, New Zealand, April 1999.
49. Jain, A.; Zongker, D. Feature selection: Evaluation, application, and small sample performance. *Pattern Anal. Mach. Intell.* **1997**, *19*, 153–158.
50. Hospodar, G.; Gierlichs, B.; Mulder, E.D.; Verbauwhede, I.; Vandewalle, J. Machine learning in side-channel analysis: A first study. *J. Cryptogr. Eng.* **2011**, *1*, 293–302.
51. Luts, J.; Ojeda, F.; Van de Plas, R.; De Moor, B.; Van Huffel, S.; Suykens, J.A.K. A tutorial on support vector machine-based methods for classification problems in chemometrics. *Anal. Chim. Acta* **2010**, *665*, 129–145.
52. Suykens, J.A.K.; Vandewalle, J. Least squares support vector machine classifiers. *Neur. Process. Lett.* **1999**, *9*, 293–300.
53. Pei, L.; Liu, J.; Guinness, R.; Chen, Y.; Kröger, T.; Chen, R.; Chen, L. The Evaluation of WiFi Positioning in a Bluetooth and WiFi Coexistence Environment. In *Proceedings of 2nd International Conference on Ubiquitous Positioning, Indoor Navigation and Location-Based Service (UPINLBS 2012)*, Helsinki, Finland, 2–5 October 2012.

PUBLICATION 4

R. E. Guinness, “Beyond where to how: A machine learning approach for sensing mobility contexts using smartphone sensors,” *Sensors*, vol. 15, no. 5, pp. 9962–9985, 2015.

Sensors **2015**, *15*, 9962-9985; doi:10.3390/s150509962

OPEN ACCESS
sensors
ISSN 1424-8220
www.mdpi.com/journal/sensors

Article

Beyond Where to How: A Machine Learning Approach for Sensing Mobility Contexts Using Smartphone Sensors †

Robert E. Guinness

Finnish Geospatial Research Institute, Geodeetinrinne 2, FI-02430 Masala, Finland;
E-Mail: robert.guinness@nls.fi; Tel.: +358-50-362-2796

† This paper is an extended version of a paper paper entitled “Beyond Where to How: A Machine Learning Approach for Sensing Mobility Contexts Using Smartphone Sensors” presented at 26th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS + 2013), Nashville, Tennessee, 16–20 September 2013.

Academic Editor: Vittorio M.N. Passaro

Received: 26 February 2015 / Accepted: 9 April 2015 / Published: 28 April 2015

Abstract: This paper presents the results of research on the use of smartphone sensors (namely, GPS and accelerometers), geospatial information (points of interest, such as bus stops and train stations) and machine learning (ML) to sense mobility contexts. Our goal is to develop techniques to continuously and automatically detect a smartphone user’s mobility activities, including walking, running, driving and using a bus or train, in real-time or near-real-time (<5 s). We investigated a wide range of supervised learning techniques for classification, including decision trees (DT), support vector machines (SVM), naive Bayes classifiers (NB), Bayesian networks (BN), logistic regression (LR), artificial neural networks (ANN) and several instance-based classifiers (KStar, LWLand IBk). Applying ten-fold cross-validation, the best performers in terms of correct classification rate (*i.e.*, recall) were DT (96.5%), BN (90.9%), LWL (95.5%) and KStar (95.6%). In particular, the DT-algorithm *RandomForest* exhibited the best overall performance. After a feature selection process for a subset of algorithms, the performance was improved slightly. Furthermore, after tuning the parameters of RandomForest, performance improved to above 97.5%. Lastly, we measured the computational complexity of the classifiers, in terms of central processing unit (CPU) time needed for classification, to provide a rough comparison between the algorithms in terms of battery usage requirements. As a result, the classifiers can be ranked from lowest to highest complexity (*i.e.*, computational cost) as follows: SVM, ANN, LR, BN, DT, NB, IBk, LWL and KStar. The instance-based classifiers take considerably more computational time than the non-instance-based classifiers, whereas the slowest non-instance-based classifier (NB) required about five-times the amount of CPU time as the fastest classifier (SVM). The above results suggest that DT algorithms are excellent candidates for detecting mobility contexts in smartphones, both in terms of performance and computational complexity.

Keywords: context awareness; smartphone sensors; machine learning; classification; mobility context; supervised learning

1. Introduction

Devices equipped with Global Navigation Satellite System (GNSS) receivers are approaching ubiquity with the inclusion of GPS receivers (or GPS + GLONASS receivers) in nearly all modern smartphones and increasingly in other highly mobile devices, such as tablets and so-called "smart watches". Although the GNSS receivers in consumer electronic devices tend to provide a less accurate positioning result than professional-grade receivers, the quality and reliability is adequate for many navigation and user tracking applications, as well as a growing number of location-based services. Certain issues continue to challenge researchers, such as obtaining reliable positioning in indoor and urban canyon environments, but it can be argued from some perspectives that the question of "where" has largely been solved by the wide availability of low-cost, low-power-consuming GNSS receivers.

In addition, nearly all smartphones are equipped with a variety of other sensors, such as accelerometers, gyroscopes and digital compasses, which could be used to understand the nature of the user's movement. This enables the smartphone to be aware of not only where the user is but how he or she is moving. This provides additional contextual information, which may be useful for a variety of different applications. A few examples envisioned within this study include detection that: "the user is riding a bus", "the user is driving a car", "the user is walking", *etc.* We use the term mobility context to refer to contextual information related to the user's mode of mobility.

The question of how a person is moving is related to the long-sought goal in mobile and ubiquitous computing of creating context awareness. It is certainly not the only relevant question in context awareness, but a solution that could reliably answer it would provide significant value in terms of enabling context-aware applications. For a recent review of context-aware computing, see [1].

There are many potential applications where mobility context could be used: an application that automatically searches bus or train timetables as the user walks towards a stop or station and accordingly plans his or her route, an application that tracks a user's carbon footprint as a result of using a car or other means of transportation, functionality in a smartphone to automatically respond to incoming phone calls with a message "Sorry, I'm driving..." whenever the user is driving or an application to track a user's physical activity (walking, running, *etc.*) for personal health monitoring. In addition, mobility context can be used in the navigation subsystem of a smartphone in order to improve the navigation solution. For example, when the smartphone detects that the user is walking, it could apply an optimized Kalman filter to fuse measurements from the various sensors and account for shortcomings of the GNSS receiver (e.g., in an urban canyon environment). The mobile computing literature provides many other examples of application concepts where mobility context plays a strong role (e.g., [2]).

Despite the great potential for applications of mobility context, a reliable solution to provide this type of information is not yet available commercially, although several companies have attempted such offerings. This depends, of course, on one's definition of reliable, but a review of user feedback to these commercial offerings suggests that users are not yet satisfied with the accuracy of existing solutions. In addition, a frequent complaint is regarding the increased energy usage of these solutions (*i.e.*, quickly draining the smartphone's battery). This suggests that the designers should take into account the computational complexity of the algorithm that determines the mobility context.

Several researchers have performed relevant studies on mobility context recognition. For example, [3] used a mobile phone with a GPS receiver and accelerometer to determine transportation modes of users, achieving 93.6% classification accuracy with the use of discrete hidden Markov models. They did not, however, distinguish between different types of motorized transport (e.g., car, bus, train, *etc.*), but grouped all of these into a single class. Bancroft *et al.* [4] used a foot-mounted device (with GPS receiver and inertial measurement unit) to determine different motion-related activities, such as walking, running, biking and moving in a vehicle, reporting less than 1% classification error during 99% of measurements. This study focused on applications for military personnel and first responders, where the use of specially-designed foot-mounted devices is feasible. Although foot-mounted accelerometers are gaining in popularity (e.g., Nike + iPod sensor), they are not yet ubiquitous in usage. Thus, solutions that rely on foot-mounted devices are of limited applicability in the current consumer market.

Accordingly, our long-term goal is to develop a solution that detects a wide range of mobility contexts (walking, running, biking, car, bus, train) using only devices that are truly ubiquitous in today's environment. Smartphones are an obvious choice, due to the rich set of sensors that they possess and their widespread use among the general public. According to market research firm Strategy Analytics, the number of smartphones in use worldwide surpassed one billion in the third quarter of 2012, and they estimate that this figure will double by 2015 [5]. Thus, if a solution for the mobility context can be developed that relies only on a smartphone, its potential impact is enormous.

This paper is an extension of a previously published conference paper [6], which was, to the authors' best knowledge, the first study to combine the use of GPS, inertial sensors (e.g., accelerometers), information from geographic information systems (GIS) and ML techniques in order to detect mobility contexts. Several studies have used GPS, GIS and accelerometers with the aim of understanding transport-related physical activity, and [7] provides a relatively recent review of this literature. A few relevant studies have been published in more recent years. For example, [8] used GPS, GIS and accelerometers to assess travel behaviors. No ML techniques, however, were used to classify the behavior into different mobility contexts. Furthermore, the study was not implemented using smartphone-based hardware. Elhoushi *et al.* [9] studied the use of GNSS and micro-electro-mechanical system (MEMS) sensors, together with ML techniques, to detect the mobility context. The authors report overall accuracy of nearly 95%; however, they do not specify which ML algorithm was used for classification. Other studies have combined GPS and accelerometer data with GIS information, but for other purposes, such as identifying the areas where children exerted the most physical activity [10]. What is consistently lacking in many existing studies involving GPS, accelerometer and GIS information is a systematic use and evaluation of available ML techniques for classification purposes, which is surprising given the success of ML techniques in many other disciplines.

There are several studies utilizing ML and accelerometer data. Jin *et al.* [11] detected several motion states using an armband-mounted accelerometer and a "fuzzy inference system", which can marginally be considered an ML technique. Other studies that used both accelerometer data and standard ML techniques to determine mobility or activity-related contexts include [12–16]. They are all lacking, however, in the use of GPS or GIS. There has been one study that used ML, GPS and GIS, but not accelerometers [17]. In a later work by Stenneth, however, accelerometers were also used [18]. From certain aspects, this is the most similar to our work compared to all other existing studies.

The main goal of this present study is to determine if a machine learning technique exists that, given information provided by GPS, inertial sensors and GIS, can produce a classifier having both high performance (in terms of recall rate) and low computational complexity. If this can be achieved, then it offers strong support for the feasibility of detecting the mobility context using smartphones.

This paper is organized as follows. First, we describe in greater detail the overall methodology used for creating a mobility context classifier. Next, we depict the experiments that we conducted in order to collect data from various mobility contexts using a smartphone, including a description of the application developed for this purpose. Thirdly, we discuss the specific procedures used to process and analyze the collected data and present the results of the experiment. Finally, we draw some conclusions and discuss our future work in this research topic.

2. Methodology

Since our goal is to detect mobility contexts, such as "user is walking" and "user is riding a bus", the main task is one of classification. One approach would be to use unsupervised learning techniques, such as clustering, in order to organize the data into groups (*i.e.*, classes) of high similarity without specifying the groups *a priori*. In our case, however, we desire to obtain groups that have a clear real-world interpretation, so it is more appropriate to define them according to our natural intuitions about the mobility context. Furthermore, we would like some measure of "performance" of our resulting classifier that is easy to interpret. An obvious choice is the rate at which the classifier produces the "correct" classification of the user's motion, *i.e.*, the recall rate. Thus, it is necessary to manually label the data instances with the correct class, leading to the methods of supervised learning.

The basic process used in this study is summarized in Figure 1, which follows the standard supervised learning approach. In the training phase, a function is generated that maps each of the input vectors to a target class:

$$f : \boldsymbol{x} \rightarrow y \tag{1}$$

where x is the input vector and $y \in Y = \{y_1, y_2, ...y_m\}$, *i.e.*, the set of all possible classes. The exact form of this function depends on the ML algorithm being used. In some cases, the algorithm produces a probabilistic model of the class distributions, given the input vector, *i.e.*, $p(y|x)$. The function is made deterministic by selecting the most likely class according to:

$$y = \underset{y \in Y}{\arg\max}\, p(y|\boldsymbol{x}) \tag{2}$$

Finally, after the function is learned, its performance is measured during the testing phase (using the testing set), where the values of y output from the function f are compared to the labeled values (*i.e.*, the "correct" classes). Note that the same function f is used in both the training and testing phases.

Because the performance will exhibit some variance, due to the finite size of the testing set, this process is repeated and the results averaged. Specifically, we use 10-fold cross-validation.

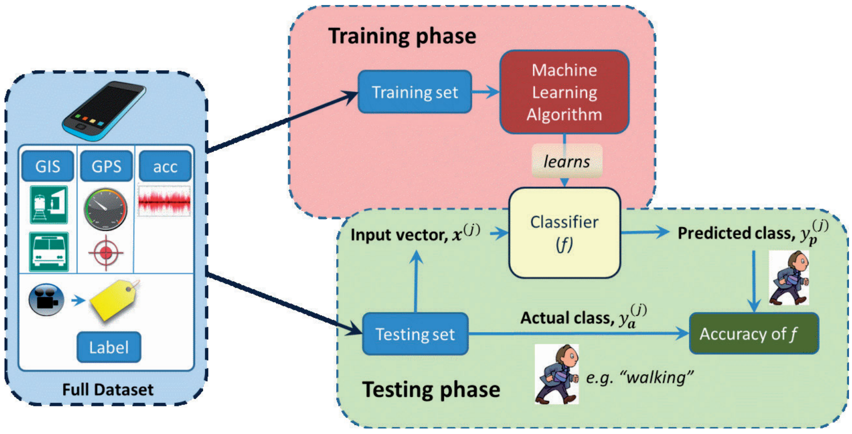


Figure 1. Overview of the supervised learning process, as used in this study.

3. Experiment Design and Data Collection

For this study, we implemented a smartphone application for the Android operating system capable of capturing accelerometer and GPS data. This application, called *CommutingContext*, also performed several data processing and information retrieval functionalities, which will be described in further detail below. The application was tested on a Samsung Galaxy Nexus smartphone, which is equipped with a SiRF StarIV GSD4t GPS receiver and Bosch BMA180 three-axis accelerometer [19]. The smartphone was running Android Version 4.1 ("Jelly Bean" release).

The data were collected over the course of one day during summer 2012 using two male test subjects, aged between 30 and 40 years and of good physical health. As this study is of a preliminary nature, no attempts were made to control for or analyze the possible effects of age, gender, height or other characteristics of the test subjects on the collected data. All data were collected in the western Uusimaa region of Finland.

The test subjects were asked to perform various mobility activities, including walking, running, riding a bus, riding a train and driving a car and to do so in the same manner as they naturally would. The only departures from normality were that the test subjects were asked to shake the phone at the start and end of each test run (used as a signal for later calibrating the time system), and they were also asked at various points during the tests to assume a "static" motion state (primarily after or before a "walking" motion state). Lastly, the subjects were instructed to place the phone in their pants pocket, in order to maintain some consistency in the smartphone's placement and orientation.

In total, fifteen test runs were performed, varying in length and mobility contexts. Some details of these test runs are presented in Table 1. Each of these test runs were either video + audio recorded or audio recorded by a dedicated device, operated by an observer. The purpose of this was to allow us to manually "label" the true mobility context as a function of time, providing a reference to measure our classification performance against. About halfway through the experiment, we switched from video + audio recording to audio-only recording, because we noted that this was sufficient for reference purposes and also due to the potential privacy concerns with recording video on public buses or trains.

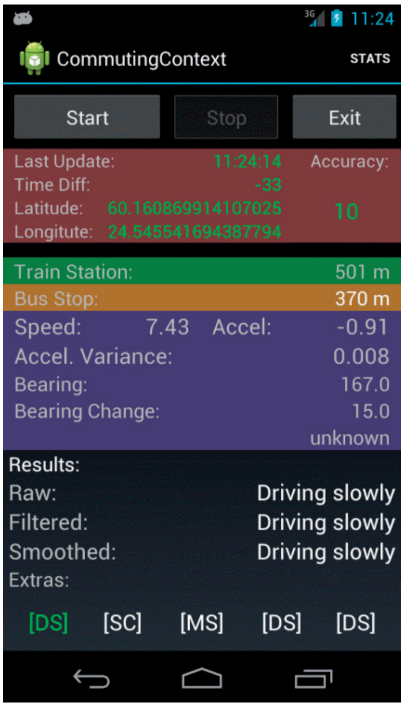
Table 1. Test runs comprising the dataset. W = walking; R = running; S = static; MS = moving slowly; RT = riding train; RB = riding bus; D = driving.

Test Subject	No. of Data Instances	Time Length	Motion Contexts
Subject 1	582	10:03	W, S
Subject 1	292	11:32	W, S
Subject 2	364	6:21	W, S
Subject 2	636	15:09	W, S, MS, RT
Subject 2	269	10:48	W, S
Subject 2	654	11:08	W, S, RT
Subject 2	583	9:46	W, S, MS
Subject 1	1266	21:42	W, S, MS, RB
Subject 1	278	4:39	W, S, R
Subject 2	176	2:56	W, S, R
Subject 1	293	4:53	W, S, R
Subject 1	765	16:38	W, S, RB
Subject 2	1017	17:11	W, S
Subject 1	593	10:41	W, S, D
Subject 1	730	12:34	W, S, D
Total	**8498**	**2:46:01**	

3.1. The CommutingContext Application

The main purpose of the CommutingContext application was to collect GPS and accelerometer data and to perform some basic data processing and information retrieval functions. A screenshot of this application is shown in Figure 2.

One of the information retrieval functions that the application performed was to look up train stations and bus stops in the vicinity of the user, using the position provided by the GPS and the smartphone's wireless data connection. Typically Universal Mobile Telecommunications System (UMTS) or Evolved High-Speed Packet Access (HSPA+) connections were available during the experiment. Two separate web-based services were used for these purposes: Google Places API for train stations [20] and the Reittiopas API, provided by the Helsinki Regional Transport Authority [21]. The primary reason for using Google Places API was its global availability. For bus stops, however, the Google Places API was not comprehensive enough for the areas where the experiment was to be performed. The Reittiopas API, on the other hand, is highly comprehensive and accurate, but is limited in scope to the areas serviced by the Helsinki Region Transport (HRT) system (known in Finland as Helsingin seudun liikenne (HSL)).



```
Last Update:          11:24:14   Accuracy:
Time Diff:                 -33
Latitude:   60.160869914107025       10
Longitute:  24.545541694387794
Train Station:                       501 m
Bus Stop:                            370 m
Speed:        7.43    Accel:         -0.91
Accel. Variance:                     0.008
Bearing:                             167.0
Bearing Change:                       15.0
                                   unknown
Results:
Raw:                        Driving slowly
Filtered:                   Driving slowly
Smoothed:                   Driving slowly
Extras:
  [DS]    [SC]    [MS]    [DS]    [DS]
```

Figure 2. Screenshot of the CommutingContext application.

Both of these web services return an array of station/stop objects for a given set of latitude/longitude coordinates, ordered from nearest to furthest away. The application then uses the Android API's built-in method for calculating the precise distance between the user and the nearest station/stop.

Another important function of the application was to perform data processing operations, especially with respect to the accelerometer data stream. Previous studies [14,22] have shown that an important feature for motion classification is the variance (σ^2) in the norm of the acceleration ($|a|$), defined by the following equations:

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3}$$

$$\sigma_t^2 = \sigma_{t-1}^2 + \overline{|a|}_{t-1}^2 - \overline{|a|}_t^2 + \frac{|a|_t^2 - |a|_{t-N}^2}{N} \tag{4}$$

where $\overline{|a|}_{t-1}$ and $\overline{|a|}_t$ represent the mean of $|a|$ at times $t-1$ and t, respectively, and N is the size of the window over which σ^2 is calculated. Because σ^2 is calculated recursively, the memory required for its calculation is bound by the window size, N. In our case, we used a window size of 50 accelerometer measurements, which corresponds to slightly more than a one-second window. Preliminary tests conducted during application development showed that the resulting acceleration signal produced from this feature was sufficiently responsive to motion changes, such as walking and stopping. Using the norm of acceleration also has the beneficial property that it is invariant under changes in the orientation of the smartphone.

In total, the application recorded in a log file a total of fourteen features. Due to the importance of GPS in this experiment, no data were recorded if there was an outage in the GPS solution. For debugging purposes, a subset of these features was displayed in the user interface (see Figure 2). In addition, the raw accelerometer data were recorded in a second log file for post-processing purposes. One additional feature, which will be described in detail below, was produced from this accelerometer data during post-processing.

4. Data Analysis

Now, we describe how the above methodology was implemented in this study. The first task was to manually label the 8498 data instances with the correct mobility context, provided by the reference recordings. A total of seven motion classes were defined, including: static, moving slowly, walking, running, driving a car, riding a bus and riding a train. The distribution of these classes in the dataset is shown in Table 2. Due to the differences between the time systems in the test smartphone and the reference recording device, a "shake" signal was used to perform time calibration. This shaking of the phone, which could be seen clearly in the acceleration data, allowed us to synchronize between the two time systems for labeling purposes. Each mobility context and the associated transitions could be clearly recognized in the video recordings, and the times of the transitions were noted. For the cases where audio recordings were used, transitions were verbally noted by the observer during the experiment; thus, the transition times were noted in a similar manner during post-processing.

Table 2. Distribution of the mobility classes.

Mobility Class	No. of Instances	Percentage
static	1144	13.5%
moving slowly	297	3.5%
walking	3443	40.5%
running	532	6.3%
driving a car	1135	13.4%
riding a bus	1482	17.4%
riding a train	465	5.5%

In addition, some other simple post-processing steps were performed in MATLAB, such as filling in labels between the mobility transitions and removing extraneous data from the beginning and end of the data files. We also removed data corresponding to circumstances not reflected in any of the classes, for example walking up/down stairs near a train station (these times were noted in the reference recordings).

Finally, the data were formatted in MATLAB into a form that could be used in Weka, the software used for the majority of our data analysis. The entire dataset is available via GitHub, see [24].

4.1. Selecting Features for Classification

The data contained a total of fifteen features, not including the class label. Four of these features were related to the time system (elapsed time, wall time, Unix time and GPS time). Although some mobility contexts could be correlated with respect to time of day (particularly for specific individuals, whose commuting activities follow some routine), we wish our final solution to perform well independent of such factors. Therefore, we removed these as classification features. Similarly, we removed latitude, longitude and heading as features for consideration, since we would not expect these to be invariant across all users. We considered the remaining eight features, described in Table 3, all to be candidates for use in classification.

Table 3. Description of the features.

Feature	Source	Description
speed	GPS	Speed from GPS, converted to km/h.
speedChange	GPS	Difference in speed from previous GPS measurement.
accelVariance	accelerometer	Variance in the norm of the acceleration (Equation (4)) over the moving window.
headingChange	GPS	Sum of absolute value of heading changes over the moving window.
accuracy	GPS	Accuracy rating from GPS, assumed to be meters of horizontalerror.
trainDistance	GPS/GIS	Distance to the nearest train station in meters.
busDistance	GPS/GIS	Distance to the nearest bus stop in meters.
1 HzPeak	accelerometer	Strength of 10-Hz peak in FFT of the accelVariance signal.

As mentioned above, one feature was not computed in real-time during the data collection, but instead was computed during post-processing. This is the "1 HzPeak" feature. As this feature requires a transformation to the frequency domain, we chose not to attempt to compute it in real-time for this research. Although modern smartphone processors are certainly capable of computing such features in real-time, using a fast Fourier transform (FFT) algorithm, for simplicity, we decided to perform this step in post-processing. It was computed by taking an FFT of the variance of the norm of the acceleration (as described in Section 3.1 above), using a sampling frequency of 125 Hz and operating on four-second windows of accelerometer data. After examining the spectrum plots for many segments of the training data, we decided to use as a feature the relative strength of the FFT peak at about 1 Hz. This feature p_r was computed as:

$$p_r = \frac{2 * p_{0.98Hz}}{p_{0.73Hz} + p_{1.22Hz}} \tag{5}$$

where $p_{0.98Hz}$, $p_{0.73Hz}$ and $p_{1.22Hz}$ are the peaks at 0.98 Hz, 0.73 Hz and 1.22 Hz, respectively. The feature p_r can be intuitively understood as a signal related to walking, as most people normally walk with a gait around 1 Hz in frequency. This is supported by previous research and also was evident in our data in spectrum plots corresponding to periods of walking.

4.2. Algorithm Performance Comparison Using Weka

Weka is an open-source, widely-used software framework for ML and data mining [23]. It was chosen because it supports a large number of ML algorithms, which have been tested and verified using many different reference datasets. It is also suitable for developing new ML algorithms, although all of the algorithms used in this study are standard ones distributed with the software package.

Table 4. Comparison of algorithm performance.

Classification Algorithm	Accuracy (% Correct)	Standard Deviation
Decision Trees		
RandomForest	96.51	0.59
NBTree	95.40	0.71
J48graft	94.93	0.77
J48	94.83	0.81
RandomTree	94.45	0.93
LMT	93.25	1.29
FT	92.66	1.02
BFTree	93.77	0.83
REPTree	93.58	1.06
LADTree	85.37	1.29
SimpleCart	94.00	0.83
Artificial Neural Networks		
MultilayerPerceptron	87.15	1.80
Bayesian Techniques		
NaiveBayes	81.47	1.15
BayesNet	90.89	0.93
Logistic Regression		
Logistic	83.41	1.05
Support Vector Machines		
SMO	80.24	1.02
Instance-Based Classifiers		
KStar	95.55	0.70
IBk(kNN = 5)	80.32	1.25
LWL	95.45	0.14
Baseline		
ZeroR	40.52	0.06

Included in the Weka software package (Version 3.6.8) is the Weka Experiment Environment, which allows one to perform repeated model training using different ML algorithms and employing standard performance testing techniques, such as cross-validation. We used this environment to perform classification on our dataset using a wide range of ML algorithms. Each algorithm was used to train and test a model using 10-fold cross-validation. Because performance varies depending on how the folds are partitioned, this procedure was repeated 10 times for each algorithm, in order to obtain adequate statistics regarding classification accuracy. The full set of algorithms we employed, grouped by type, is listed in Table 4, along with the key performance metric, classification accuracy (also known as recall rate).

In Table 4, we use naming conventions consistent with the Weka software package. These sometimes differ from naming conventions used in the ML literature. For example, J48 is a java implementation of C4.5, which was originally developed by Quinlan (see [25]). Due to space limitations, we will not attempt to describe all of these algorithms in detail, but refer the reader to [26] or the Weka software documentation for details [23]. Several of the algorithms will be described further in the Discussion Section.

Here, we will mention only one algorithm, ZeroR, which provides a baseline for comparing to the other algorithms. This is a trivial algorithm that simply classifies all of the testing instances as the most frequent class in the training set (in our case, the "walking" class). Thus, all other algorithms should perform at least as well as ZeroR (as is the case here).

It is important to note that all of the algorithms included in the Weka software package assume that they will be applied to independent and identically distributed (i.i.d.) data. In fact, in nearly all of the ML literature, it is assumed that the data of interest is i.i.d. In our application domain, however, this is clearly not the case. Time series data from different mobility contexts are expected to be highly correlated in time. This issue is handled in Weka by randomizing the order of the data instances, which obscures the time dependence. See the Discussion section for further implications of this issue.

4.3. Feature Analysis

Because ML algorithms can sometimes suffer poor performance due to the "curse of dimensionality", we next endeavored to determine if better performance can be achieved by using a subset of the seven features described above. Feature analysis is also of particular importance in this research domain because different features have different costs, in terms of the amount of energy required to produce them. Thus, feature analysis provides a means of multi-objective optimization, e.g., when the goal is to achieve adequate performance while minimizing energy requirements.

In order to perform feature analysis, we performed a combination of greedy forward feature selection (GFFS) and greedy backward feature selection (GBFS). We first re-ran the analysis for five algorithms (chosen because of their good performance and diversity) using each feature individually (*i.e.*, in each analysis, only a single feature was utilized). The results, presented in the top section of Table 5, indicate that speed, acceleration variance (accelVariance), distance to train station (trainDistance), speed change (speedChange) and distance to bus stop (busDistance) are among the most important features individually. In a similar fashion, we re-ran the analysis using all of the features minus a single feature (varying the missing feature each time). These results, shown in the second sub-section of Table 5, show that, for most of the algorithms and features, the classifiers do not degrade significantly when a single feature is removed. In fact, for the case of RandomForest with speedChange removed, the resulting classifier performed slightly better, and when headingChange was removed, there was a negligible change in the performance. Furthermore, note that speedChange, although among the top individually performing features, actually degrades the performance when a larger set of features is used.

Table 5. Feature selection analysis. LR, logistic regression; NB, naive Bayes.

Feature set	RF	J48	MP	LR	NB	SV
Individual features						
speed	72.9	73.8	71.4	67.7	71.5	55.5
accelVariance	64.2	70.7	62.2	57.4	55.6	57.3
trainDistance	61.6	69.0	56.1	51.7	53.9	50.7
speedChange	50.7	50.7	44.8	40.5	44.0	40.5
busDistance	49.7	56.9	45.6	43.4	43.3	42.9
accuracy	47.1	47.0	46.7	43.9	43.9	44.0
1HzPeak	46.3	51.6	49.8	45.0	43.6	40.5
headingChange	41.3	41.1	49.1	40.5	39.7	40.5
All features, except						
speedChange	97.7	95.9	90.1	87.1	79.7	85.0
headingChange	97.1	95.5	87.8	85.5	81.6	81.9
accelVariance	96.5	94.3	85.6	81.6	81.2	74.3
accuracy	96.2	94.3	85.9	83.6	79.5	78.9
speed	96.1	94.2	85.2	79.8	74.1	75.5
busDistance	96.1	94.8	86.5	82.7	82.0	78.7
1HzPeak (all time-domain features)	96.5	94.8	87.1	83.4	81.5	80.2
trainDistance	94.3	92.9	83.1	82.1	78.8	79.4
All time-domain features, except						
speedChange	96.7	94.9	86.9	83.5	81.4	80.2
headingChange	96.5	94.8	86.9	83.4	81.4	80.4
accelVariance	96.1	93.5	83.8	78.3	78.7	73.5
accuracy	95.6	93.6	85.1	81.8	78.0	77.6
busDistance	95.1	93.8	84.6	80.8	79.5	77.0
speed	94.6	92.1	83.0	77.6	72.9	73.1
trainDistance	93.3	92.0	82.4	81.1	76.7	77.8
Other subsets						
{av, s, t, b, acc, 1hp}	97.6	96.0	90.2	87.1	79.5	85.0
{av, s, t, b, acc}	96.5	94.9	86.6	83.5	81.2	80.2
{av, s, t, b, 1hp}	96.3	94.6	85.5	83.5	78.6	79.1
{av, t, b, acc, 1hp}	96.1	94.5	82.5	79.9	70.1	75.7
{av, s, t, b}	93.8	93.8	84.5	81.8	77.5	77.6
{av, s, h, acc, sc, hc, 1hp} (no GIS)	93.8	92.4	84.8	80.6	79.4	76.3
{av, s, t}	92.4	92.4	81.0	77.4	76.3	72.0
{b, t, 1hp}	89.6	88.4	67.4	62.3	55.2	57.4
{av, s, b}	89.4	89.4	80.8	79.0	73.4	75.0
all	**97.1**	**95.4**	**87.8**	**85.5**	**81.9**	**81.8**

Next, we re-ran the five classifiers using only time-domain features and removing a single feature among the seven remaining features. The reason we were interested in these results is because, in some applications, it may be too costly in terms of energy to generate frequency-domain features. We wanted to better understand the performance without the 1 HzPeak feature. To see the effects of removing individual time-domain features, the results in the third sub-section of Table 5 can be compared to the second to last row in the second sub-section, where all time-domain features were used.

Lastly, we re-ran the analysis using various small subsets of the features. A sampling of these subsets is shown in the last sub-section of Table 5. None of these subsets were the best overall performer, but one subset ({av, s, t, b, acc, 1hp}) was nearly as good and surpassed nearly all of the classifiers that used all available features. Note also the subset containing all but the GIS-related features (*i.e.*, {av, s, h, acc, sc, hc, 1hp}) contains all of the features, except the two GIS-related ones. Its performance was about 3.4% less than when using all features. This highlights the usefulness of GIS in this domain. This difference would most likely have been more significant if the dataset had consisted of more instances of "riding a bus" and "riding a train" (see Table 2).

Although the results in Table 5 do not represent an exhaustive feature selection procedure, they offer some support that, with the possible exceptions of speedChange and headingChange, the features used in this study are useful for classification and have minimal redundancy.

4.4. Parameter Tuning

Most ML algorithms specify one or more parameters that can be tuned in order to optimize performance or other characteristics, such as the tendency to overfit the training data. The Weka platform supplies the user with default parameters for each algorithm, usually based on some rule of thumb or generally accepted set of values. Not all algorithms, however, have a set of universally applicable parameter settings and should ideally be tuned according to training data. We selected the RandomForest algorithm for further parameter tuning, because it performed the best in most of the above evaluations. For parameter tuning, we used as our input data the feature set containing all features, except speedChange. This choice was made because the feature analysis presented above showed that this was the best set of features using the default parameter settings for RandomForest.

In the Weka implementation of RandomForest, there are three parameters that are user-defined: (1) the max depth of the decision trees; (2) the number of attributes (*i.e.*, features) used for random selection during the learning process; and (3) the number of trees grown. Early analysis showed that the recall rate was not very sensitive to the first parameter. We tested the performance at increasing values for this parameter and found that somewhere in the range of 20 to 30, the performance leveled off and remained constant for higher values. It is also possible to set this parameter such that no constraint is placed on tree depth, and this is the setting we used in further analysis.

The default value in Weka for the second parameter (number of attributes used in random selection), which we call F, is $\lfloor log_2 M + 1 \rfloor$, where M is the total number of attributes available. For our dataset, this parameter would default to $F = 3$. We wanted to determine if this was indeed the best setting for this parameter or if better performance could be achieved with another value. Regarding the last parameter, the number of trees grown (denoted by K), it is generally accepted that the generalization error converges to a minimum as K increases. We wanted to see at which value an increase in K provides negligible performance benefit, since increasing this also increases the complexity of classification.

In order to avoid the problem of overfitting from the parameter tuning process, we created a hold-out test set of 10% from the original dataset using random sampling without replacement. As a result, the class distribution of the hold-out test set is similar, but not identical to the class distribution of the entire dataset. With the remaining 90% of data, we performed training and evaluation of the RandomForest classifier using 10-fold cross-validation, while varying the parameters F and K using a grid search approach. The results are shown in Figure 3. From the contour lines, we can see that for a small choice of K (<10), there is no clear best choice of F. However, for $K > 10$, the best choice of F is clearly two. This can be seen from the contour lines, because for all other values of K, a greater number of trees is needed to achieve the same level of performance as for $F = 2$. We have highlighted one particular point on this curve, where $F = 2$ and $K = 30$. The performance at this point is about 97.444%, which, compared to the point $F = 2$ and $K = 200$, is only 0.2% worse in relative terms. Finally, we set $F = 2$ and re-trained the RandomForest classifier using the full 90% training data, but varying the value of K, and evaluated the resulting performance using the 10% hold-out test set. The resulting performance was 97.291% correctly classified samples. The purpose of the analysis using the hold-out test set is to provide an estimate of how the classifier will perform on new, unseen data; however, to produce an estimate of good confidence, a large amount of labeled data is needed. Given that this hold-out set only contains 849 data samples and, for some classes, only a few tens of samples, the main value of this estimate is that it lends support to the fact that the parameter tuning process did not introduce significant overfitting. Taking together, the parameter tuning results and the hold-out set results provide good evidence that a classification performance of around 97% can be achieved, although we caution the reader that results from two test subjects moving about in one geographic region do not necessarily generalize to a larger, global population of smartphone users.

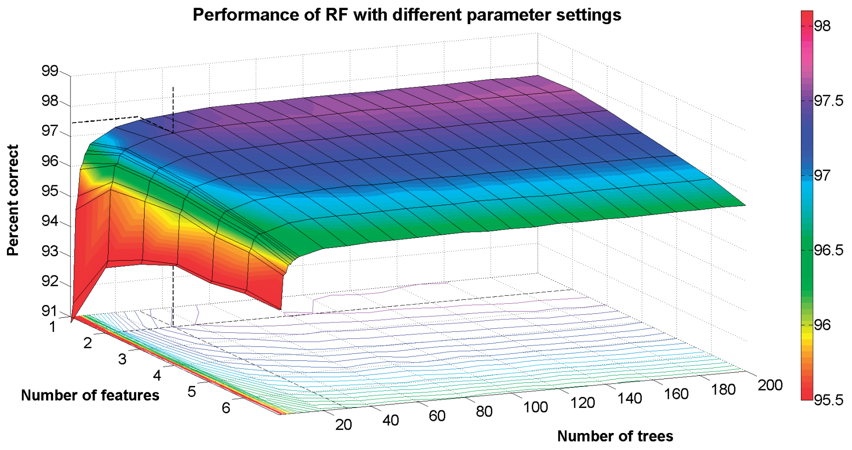


Figure 3. Performance of Random Forest as a function of parameter settings.

Lastly, the results from the parameter tuning analysis suggest that for the types of applications described in this paper, the choice of K could be a user-selectable parameter (e.g., $10 \leq K \leq 30$), since higher values will improve the performance, but also increase the computational complexity of the classifier. Clearly, however, the results in Figure 3 show that increasing K to values above about 30 will only provide minuscule improvements in performance, and it is a case where the law of diminishing returns holds strong.

4.5. Computational Comparison of Algorithms

In order for a classifier to be feasibly implemented in a smartphone, its underlying classification algorithm must be relatively simple; otherwise, classification cannot be performed in real-time or would require an unreasonable amount of resources, draining the limited battery power of a smartphone. For this reason, we next analyzed the computational complexity of the classifiers. We expect there to be a direct linear relationship between computational complexity and battery usage, although this must be confirmed in future studies (see [27]).

First, an important distinction must be made regarding the operations that we are analyzing. As described above, ML algorithms necessarily involve two steps: model training and testing (in our case, classification). In this analysis, the only portion that we are interested in is testing, because this is what would be implemented in a smartphone in order to perform real-time classification. Model training, on the other hand, is likely to be performed on a PC (or server) and the resulting models later transferred to a smartphone. For most ML algorithms, model training requires vastly more computations than testing/classification.

Measuring the relative complexity of the classification algorithms is straightforward, because Weka can measure and save the CPU time used during the testing portions of each analysis run. Although these times are likely to vary with the system and platform being used, they are suitable for purposes of comparing the different algorithms. These tests were performed on a PC with an Intel Core i5-2450M processor with a 2.5-GHz clock speed. The results are presented in Table 6.

The times represent the average amount of CPU time used to classify 1/10th of the samples in the dataset (about 850 instances). To aid the reader in understanding the presented values, consider a system that classifies the user's mobility context at a rate of 1 Hz. If RandomForest were used, then the classification part of the computation would require about 10 µs of CPU time each second, whereas the classifier based on IBkwould require nearly 1 ms of CPU time each second. While both may be feasible in principle to operate at a rate of 1 Hz, clearly the IBk classifier would consume significantly more energy than the RandomForest classifier. Of course, the presented values were measured on a PC rather than a smartphone, so the actual values in a smartphone-based implementation will differ from those presented. The ratios between different classifiers in terms of CPU time, however, should remain relatively consistent over similar implementations on different processors.

The CPU time measurement procedure was repeated 100 times (10 repetitions of 10-fold cross-validation). Note that the standard deviation of many of these measurement times is high compared to the actual times, especially for the faster algorithms. This is due to limitations in the precision of the CPU time measurement provided by Weka. In the future, we aim to measure the CPU time for classification using larger datasets and directly on various smartphone CPUs. The presented results suffer from our current software limitations, but the main value at this stage in our result is to provide a comparison between different algorithms.

From the above results, we can conclude that most decision tree-based algorithms are very fast compared to other algorithms. SMOand MultiLayerPerceptronare also very fast algorithms for the testing portion. Instance-based classifiers, on the other hand, are extremely slow for testing. This will be discussed in further detail in the Discussion Section. Although the absolute speeds will certainly vary under different processors and architectures, we have no reason to believe that the relative speed of these different classifiers will differ significantly from one CPU to another. Nonetheless, this claim should be confirmed in future work.

It is also true that future implementations of, for example, instance-based classifiers could be parallelized and optimized for graphics processing units (GPUs), which are common in many modern smartphones. Recent research results have shown a ten- to twenty-fold speedup in classification utilizing GPUs and parallel versions of the k-nearest neighbors (kNN) algorithm [28]. A detailed consideration of parallelization, however, is beyond the scope of the present study.

Table 6. Comparison of computational time.

Classification Algorithm	**CPU Time (ms)**	**Standard Deviation**
Decision Trees		
RandomForest	7.96	7.84
NBTree	19.3	7.05
J48graft	3.74	6.70
J48	1.25	4.25
RandomTree	1.40	4.49
LMT	117	6.96
FT	762	54.5
BFTree	1.72	4.90
REPTree	1.09	4.00
LADTree	1.09	4.00
SimpleCart	1.25	4.25
Artificial Neural Networks		
MultilayerPerceptron	3.28	6.39
Bayesian Techniques		
NaiveBayes	13.26	6.79
BayesNet	6.55	8.05
Logistic Regression		
Logistic	4.21	6.96
Support Vector Machines		
SMO	2.65	5.90
Instance-based Classifiers		
KStar	7.21×10^4	2.74×10^4
IBk (kNN = 5)	751	16.9
LWL	8.60×10^5	7.89×10^4

4.6. Error Analysis

Lastly, we measured the types of errors that occurred for the best classifier, *i.e.*, RandomForest, using all features, except speedCahnge. Classification errors are typically presented in the form of a confusion matrix, where the true class labels and predicted labels comprise the rows and columns of the matrix, respectively. This result is presented in Table 7 below, where the bold values show the correctly predicted class label and the non-bold values show the various classification errors.

Table 7. Confusion matrix for the RandomForest algorithm applied to the dataset.

		Predicted Label						
		Walk	**Static**	**Slow**	**Train**	**Bus**	**Run**	**Car**
Actual Label	**Walking**	**98.86**	0.572	0.145	0.000	0.032	0.383	0.006
	Static	3.024	**93.81**	1.792	0.079	0.420	0.236	0.638
	Moving Slowly	2.929	7.946	**87.85**	0.000	0.875	0.000	0.404
	Riding Train	0.000	0.323	0.000	**97.87**	1.742	0.022	0.043
	Riding Bus	0.297	0.499	0.115	0.803	**97.10**	0.007	1.174
	Running	6.184	0.959	0.000	0.019	0.019	**92.82**	0.000
	Driving Car	0.009	1.445	0.000	0.326	2.326	0.000	**95.89**

Here, we see that the most common errors include the following incorrect classifications: "moving slowly" as "static", "running" as "walking", "static" as "walking", "moving slowly" as "walking" and "driving car" as "riding bus". Note that the percentages are out of all instances within the respective class, not out of the whole dataset. The accuracy/error percentages by class give a notion of how the algorithm will perform when the smartphone is engaged in a particular mobility context. For example, "walking" is detected with the highest accuracy, whereas "moving slowly" is detected with the least accuracy, in terms of the correct classification rate.

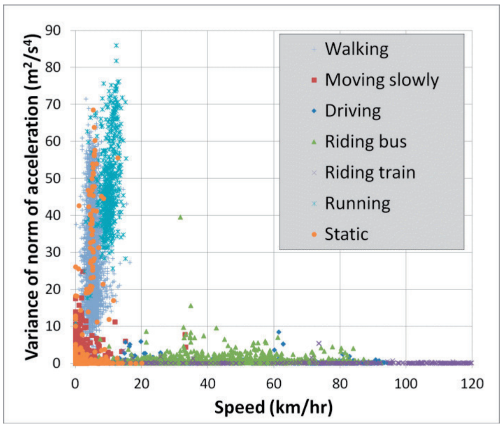


Figure 4. Plot of accelVariance *vs.* speed for all mobility classes.

Lastly, a simple way to obtain a better understanding of the cause of errors is to use two-dimensional plots of the features. Since we have seen above that speed and accelVariance are among the most important features, we created plots to see the distribution of the classes with respect to these two variables. Figure 4 below shows such a plot for the entire dataset. For greater clarity, Figure 5 shows only the classes "walking" and "running". Recall from Table 7 that 6.2% of the "running" instances were confused with the "walking" class. This is not surprising in light of Figure 5, and we will discuss this error type further in the section below.

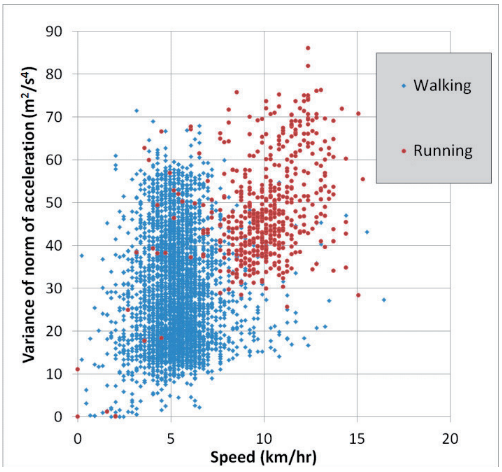


Figure 5. Same plot for only "walking" and "running" classes.

5. Discussion

The main goal of this study was to determine if an ML algorithm exists that can produce a classifier having both high performance (in terms of class prediction rate) and low computational complexity. The results show that several existing ML algorithms achieved performance above 95% accuracy, but the only ones that also have relatively low computational complexity (for classification) are decision-tree algorithms. In particular, the RandomForest and NBTree algorithms performed well. RandomForest, however, surpassed NBTree, both in terms of performance and complexity.

RandomForest is, in fact, an ensemble classifier, meaning it uses multiple classifiers, and the final classification is performed by "voting" among the classifiers. For an overview of the method of random forests, see [29]. Specifically, the Weka implementation is similar to the algorithm referred to as "Forest-RI" in [29]. RandomForest creates an ensemble of such trees using randomly-selected attributes.

There is one difference between Forest-RI and the Weka implementation of RandomForest. Weka uses REPTreeas the base classifier; The name of this algorithm comes from "reduced-error pruning", which means the decision tree is iteratively "pruned" (*i.e.*, decision nodes are removed) and then tested to see the effect on its performance [30]. Breiman's algorithm uses unpruned trees.

As mentioned above, many of the algorithms used in this study contain one or more parameters that can affect the classification performance. In all cases, except one, we used the default parameters provided by Weka and did not attempt to optimize the parameter settings for performance. The exception is for IBk, which is a classifier based on the kNN algorithm. We tried several values (1, 5, 10, 100) for the k parameter and report in this paper only the best performer (k = 5). Further attempts at parameter optimization of ML algorithms would likely require more labeled data, and as such, it is not emphasized in this study.

Next, we note that two instance-based classifiers also performed well (KStar and LWL) in terms of accuracy. In terms of computational complexity, however, these two algorithms were the worst performers. This is due to the fact that instance-based classifiers essentially work by memorizing the training data. During testing, the algorithms perform classification by searching through the training instances to find the closest match. They are sometimes known as "lazy learners" because they do not make any inferences until the testing phase. As such, they are not particularly suitable for smartphone-based processing, unless a very efficient look-up scheme could be devised.

In general, the models produced by decision-tree algorithms are intrinsically simple, because they consist of simply a set of if-then-else statements, which can easily be implemented in any programming language. The Weka API even has the ability to output a java source representation of the decision trees that it builds. Of course, the size of the tree (*i.e.*, number of nodes) will affect its computational complexity. Our results show, however, that even an ensemble of decision-trees, like RandomForest, is not significantly more complex than, for example, Bayesian techniques or logistic regression. Although the analysis of computational complexity was performed on a PC, the result in relative terms should be similar if all of the classifiers were implemented on a smartphone. Our plan to demonstrate this will be discussed in the section on Future Work.

As mentioned above, all the algorithms available in Weka make assumptions that the data will be i.i.d., which is not strictly the case with this dataset. This does not prevent the algorithms from being used with time series data, but it does mean that the algorithms cannot take advantage of the time dependence existing in the dataset [31].

One simple way to exploit the inherent time dependence would be to consider at each epoch the latest *n*-outputs of the base classifier (e.g., RandomForest) and then produce an ensemble classifier that chooses the majority from the *n* individual outputs. This would in effect "smooth" the classifier results in the same way that a low-pass filter smooths a noisy signal. The disadvantage of this approach is that the resulting ensemble classifier would not react as quickly to changes in the mobility context, since it would rely on a memory of recent context states.

In practice, it is rather challenging to perform this kind of analysis using the techniques of machine learning and the available software tools, although it could be achieved if explicitly planned during the data collection or built into the analysis software. In general, machine learning techniques call for randomized sampling from the labeled data to segregate the data into training and test data, as described above. Since the smoothing technique described above requires consecutive data samples, one would either need to have a method of randomly selecting many small sets of n consecutive data samples to segregate the data into training and test sets or one would need to break the violationof random sampling altogether. In the case of the latter, it would be very important to ensure that the data collection is done in a way that some small time series of data from within the whole dataset is representative of the whole dataset, especially in terms of class distribution. This is perhaps impractical, however, and we plan to implement the former technique in our future research.

There exist some other statistical models that utilize time dependence more deeply, such as hidden Markov models (HMMs) or conditional random fields (CRFs), which could provide better results than the ML algorithms examined in this paper. Use of these models for detecting mobility context will be another subject of future research.

Finally, we discuss some potential sources of the classification errors in our results. One likely source is due to error in the speed measurements provided by GPS. For example, we observed that about 27% of the data instances with the "static" class label have a non-zero speed measurement. In fact, more than 17% have a speed measurement >2 km/h. This level of error in speed measurements is probably largely the reason for confusion between the "static", "moving slowly" and "walking" classes. We attempted to augment the GPS speed measurements using the acceleration data, but due to the large noise levels, this proved unsuccessful. What is clear, however, is that if the error of speed measurements at low speeds could be improved, either by next-generation GNSS receivers, GNSS-IMU integration or some other means, then the classification results will also improve.

One of the most frequent classification errors was when "running" was confused with "walking". As discussed in the section above, this is due to a significant overlap between these two mobility classes in terms of the features speed and accelVariance. This is perhaps not surprising, because when the user is jogging slowly, the speed is very similar to that of fast walking. The average speed for all walking instances is about 5.34 km/h with nearly 8% of instances having speeds greater than 7 km/h. This can be compared to running, where the average speed is 10.1 km/h, and more than 6% of speed measurements were less than 7 km/h.

In Figure 5, there is an even greater overlap in terms of accelVariance. It is possible that other features of the accelerometer may be able to better distinguish between these two classes, for example the 1 HzPeak feature. Another possibility would be to collect data from additional inertial sensors, namely MEMS gyroscopes, which are available in many smartphones.

6. Conclusions

This research demonstrated that, by combining data from GPS, accelerometers and GIS with existing ML algorithms, one can build a highly-performing classifier for detecting mobility contexts of smartphone users. Results from evaluating the performance of various ML algorithms using our full feature set ranged from 80.2% to 96.5% in terms of the correct classification rate. Results from measuring the computational complexity of the classification algorithms suggested that many of these classifiers can feasibly be implemented in a smartphone, although future research must verify these preliminary results.

The only part of the process that would likely need to be implemented separate from the smartphone is the training (model learning) phase, which can be done using a standard PC with the resulting model then being transferred to a smartphone. Since training is performed off-line in supervised ML, it is not required that it be implemented in a smartphone.

Furthermore, our analysis showed that decision-tree-based algorithms and, in particular, the RandomForest algorithm are ideally suited for this type of application. The best performance was achieved when this algorithm was used with six of our seven features (*i.e.*, all features, except speedChange), which resulted in an average accuracy of 97.7%. Although RandomForest is not the least computationally complex of all of the algorithms analyzed, it is feasible to implement on modern smartphones. This conclusion is supported by the fact that it required on average less than 8 ms of CPU time to classify 850 instances, using a PC.

Finally, the most common classification errors were confusion between the "static", "moving slowly" and "walking" classes, as well as incorrect classification of "running" as "walking" and "driving car" as "riding bus". In terms of the features used in this study, these are the classes that are most challenging to correctly detect. All mobility classes were correctly classified more than 90% of the time with the exception of "moving slowly". For this class, correct classification was only achieved 87.9% of the time with nearly 8% of the error going to the "static" class and nearly 3% going to the "walking class". This is not surprising, given that "moving slowly" was defined somewhat as a bridge between these two classes.

7. Future Work

We hope that this work will serve as a baseline for future efforts for detecting mobility contexts. We examined only the ML algorithms included in the Weka software package, so future work will include extending our analysis to other existing ML algorithms, such as those based on hidden Markov models and conditional random fields. Furthermore, we have begun preliminary work on a novel algorithm that combines aspects of decision trees and hidden Markov models, which we plan to develop further.

In addition, we plan to implement several of the algorithms considered in this paper on the Android OS and to measure the CPU usage on several different smartphones, in order to more definitively demonstrate the feasibility of a purely smartphone-based solution.

We also plan to investigate other features that could be used as features for classification, including other frequency domain features, as well as from additional sensors, such as gyroscopes. With regards to gyroscopes, one possibility for the future is to use gyroscope data to reduce the error of the heading measurements, which may result in the headingChange feature being of greater value in future studies. It is also possible that other GIS type information could be used as features for classification. For example, OpenStreetMap offers a web service that allows retrieval of road segments or other "nodes" from its database [32]. These could be very useful for improving the detection of mobility contexts.

Lastly, it is clear that more mobility context data should be collected, including data from a wider range of test subjects, more diverse environments and additional mobility contexts (e.g., cycling). It should not be taken for granted that the performance results achieved from two test subjects will generalize to a larger, more diverse population. Another interesting topic of study would be to investigate the effects of placing the smartphone in different locations other than the user's pocket, for example in a backpack or handbag.

Acknowledgments

This work was supported in part by the INOSENSE project (Indoor Outdoor Seamless Navigation for Sensing Human Behavior), funded by the Academy of Finland. The author wishes to thank Ling Pei for his assistance with data collection, as well as Ruizhi Chen and Heidi Kuusniemi for supervision and mentoring support. Thanks also to the Finnish Geospatial Research Institute for organizational support.

Conflicts of Interest

The author declares no conflict of interest

References

1. Dey, A. Context-Aware Computing. In *Ubiquitous Computing Fundamentals*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2010; pp. 321–352.
2. Campbell, A.; Choudhury, T. From Smart to Cognitive Phones. *IEEE Pervasive Comput.* **2012**, *11*, 7–11.
3. Reddy, S.; Mun, M.; Burke, J.; Estrin, D.; Hansen, M.; Srivastava, M. Using mobile phones to determine transportation modes. *ACM Trans. Sens. Netw.* **2010**, *6*, 13:1–13:23.
4. Bancroft, J.B.; Garrett, D.; Lachapelle, G. Activity and environment classification using foot mounted navigation sensors. In Proceedings of the 2012 Indoor Positioning and Indoor Navigation (IPIN) Conference, Sydney, NSW, Australia, 13–15 November 2012.
5. Mawston, N. Worldwide Smartphone Population Tops 1 Billion in Q3 2012. Available online: http://blogs.strategyanalytics.com/WDS/post/2012/10/17/Worldwide-Smartphone-Population-Tops-1-Billion-in-Q3-2012.aspx (accessed on 27 April 2015).
6. Guinness, R.E. Beyond Where to How: A Machine Learning Approach for Sensing Mobility Contexts Using Smartphone Sensors. In Proceedings of the 26th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2013), Nashville, TN, USA, 16–20 September 2013.
7. Duncan, M.J.; Badland, H.M.; Mummery, W.K. Applying GPS to enhance understanding of transport-related physical activity. *J. Sci. Med. Sport* **2009**, *12*, 549–556.
8. Oliver, M.; Badland, H.M.; Mavoa, S.; Duncan, M.J.; Duncan, S. Combining GPS, GIS, and accelerometry: Methodological issues in the assessment of location and intensity of travel behaviors. *J. Phys. Act. Health* **2010**, *7*, 102–108.
9. Elhoushi, M.; Georgy, J.; Korenberg, M.; Noureldin, A. Robust Motion Mode Recognition for Portable Navigation Independent on Device Usage. In Proceedings of the 2014 IEEE/ION Position Location and Navigation Symposium (PLANS), Monterey, CA, USA, 5–8 May 2014.
10. Quigg, R.; Gray, A.; Reeder, A.I.; Hold, A.; Waters, D.L. Using accelerometers and GPS units to identify the proportion of daily physical activity located in parks with playgrounds in New Zealand children. *Prev. Med.* **2010**, *50*, 235–240.
11. Jin, G.; Lee, S.B.; Lee, T.S. Context awareness of human motion states using accelerometer. *J. Med. Syst.* **2008**, *32*, 93–100.

12. Ravi, N.; Dandekar, N.; Mysore, P.; Littman, M.L. Activity recognition from accelerometer data. In Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, USA, 11–13 July 2005; Volume 20, pp. 1541–1546.
13. Frank, K.; Nadales, M.; Robertson, P.; Angermann, M. Reliable Real-Time Recognition of Motion Related Human Activities Using MEMS Inertial Sensors. In Proceedings of the 23rd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2010), Portland, OR, USA, 21–24 September 2010; pp. 2919–2932.
14. Pei, L.; Chen, R.; Liu, J.; Chen, W.; Kuusniemi, H.; Tenhunen, T.; Kröger, T.; Chen, Y.; Leppäkoski, H.; Takala, J. Motion recognition assisted indoor wireless navigation on a mobile phone. In Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010), Portland, OR, USA, 21–24 September 2010; pp. 3366–3375.
15. Pei, L.; Guinness, R.E.; Chen, R.; Lui, J.; Kuusniemi, H.; Chen, Y. Human Behavior Cognition Using Smartphone Sensors. *Sensors* **2013**, *13*, 1402–1424.
16. Susi, M.; Renaudin, V.; Lachapelle, G. Motion Mode Recognition and Step Detection Algorithms for Mobile Phone Users. *Sensors* **2013**, *13*, 1539–1562.
17. Stenneth, L.; Wolfson, O.; Yu, P.S.; Xu, B. Transportation mode detection using mobile phones and GIS information. In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Chicago, IL, USA, 1–4 November 2011; pp. 54–63.
18. Stenneth, L. Human Activity Detection Using Smartphones and Maps. Ph.D. Thesis, University of Illinois at Chicago, Chicago, IL, USA, 2013.
19. El-Shiemy, N. The Promise of MEMS to LBS and Navigation Applications. Available online: http://www.cwins.wpi.edu/workshop12/presentation/Application_panel/ElSheimy.pdf (accessed on 27 April 2015).
20. Google. Google Places API Developer's Guide. Available online: http://bit.ly/1vBPd20 (accessed on 26 February 2015).
21. Helsinki Regional Transport Authority. Reittiopas API Developer's Guide. Available online: https://www.hsl.fi/en/helsinki-regional-transport-authority (accessed on 27 April 2015).
22. Pei, L.; Lui, J.; Guinness, R.E.; Chen, Y.; Kuusniemi, H.; Chen, R. Using LS-SVM Based Motion Recognition for Smartphone Indoor Wireless Positioning. *Sensors* **2012**, *12*, 6155–6175.
23. Weka project. Weka Documentation. Available online: http://bit.ly/1AaQEQf (accessed on 26 February 2015).
24. Guinness, R. Mobility Context data for 2013 ION Paper. Available online: http://bit.ly/MLdata2013 (accessed on 24 April 2015).
25. Quinlan, J.R. *C4.5: Programs for Machine Learning*; Morgan Kaufmann: San Mateo, CA, USA, 1993; Volume 1.
26. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA data mining software: An update. *ACM SIGKDD Explor. Newsl.* **2009**, *11*, 10–18.
27. Murmuria, R.; Medsger, J.; Stavrou, A.; Voas, J.M. Mobile Application and Device Power Usage Measurements. In Proceedings of the 2012 IEEE Sixth International Conference on Software Security and Reliability, Gaithersburg, MD, USA, 20–22 June 2012; pp. 147–156.

28. Nikam, V.B.; Meshram, B.B. Parallel kNN on GPU architecture using OpenCL. *Int. J. Res. Eng. Technol.* **2014**, *3*, 367–372.
29. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32.
30. Elomaa, T.; Kääriäinen, M. An analysis of reduced error pruning. *J. Artif. Intell. Res.* **2001**, *15*, 163–187.
31. Dietterich, T.G. Machine learning for sequential data: A review. In Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Windsor, ON, Canada, 6–9 August 2002; pp. 15–30.
32. OpenStreetMap. OpenStreetMap API Documentation. Available online: http://bit.ly/181w1zw, 2015 (accessed on 26 February 2015).

PUBLICATION 5

R. E. Guinness, J. Saarimäki, L. Ruotsalainen, H. Kuusniemi, F. Goerlandt, J. Montewka, R. Berglund, and V. Kotovirta, "A method for ice-aware maritime route optimization," in *Position, Location and Navigation Symposium–PLANS 2014*, IEEE/ION, pp. 1371–1378, 2014.

A method for ice-aware maritime route optimization

Robert E. Guinness, Jarno Saarimäki, Laura Ruotsalainen, Heidi Kuusniemi
Department of Navigation and Positioning
Finnish Geodetic Institute
Kirkkonummi, Finland
firstname.lastname@fgi.fi

Floris Goerlandt, Jakub Montewka
Department of Applied Mechanics
Aalto University
Espoo, Finland
firstname.lastname@aalto.fi

Robin Berglund, Ville Kotovirta
Department of Solutions for natural resources and environment
VTT Technical Research Centre of Finland
Espoo, Finland
firstname.lastname@vtt.fi

***Abstract*—We present a method for *ice-aware* maritime route optimization. Our aim is to increase the safety and efficiency of maritime transport under icy conditions. The proposed method is based on the A* algorithm, developed by Hart et al. It uses a model of maritime navigation, consisting of (1) a sea spatial model, (2) ship maneuverability model, (3) sea ice model, and (4) ship performance model. The sea ice model, which provides a snapshot of the sea ice conditions, is based on previous work by the Finnish Meteorological Institute. The ship performance model, based on previous work by Kotovirta et al., estimates ship transit speed as a function of ice conditions and ship design parameters. The main novelties in this paper are the application of the A* algorithm to maritime route optimization and development of an associated cost function that takes into account ice conditions and available icebreaker assistance. We present preliminary results based on the method, using the Baltic Sea as a case study. Generated routes are compared with historical routes under the same ice conditions. Areas of future work and needed enhancements are briefly discussed.**

***Keywords*— routing; ice navigation; maritime navigation; route optimization**

I. Introduction

Conventionally, navigation and routing functions of vessels are the responsibility of the vessel's captain and his or her designated crew, who rely on their extensive experience and training to navigate the vessel safely and efficiently. Increasingly, various data sources, such as electronic charts, radar imagery, and Automatic Identification System (AIS) information, are used to assist the captain in planning the vessel's route and in real-time navigation to adjust the route based on the evolving conditions at sea. In challenging maritime environments, such as in winter conditions at high latitudes, these various data sources play a critical role [1]. As the demand for maritime transport increases, especially in northern and Arctic regions, the need for systems that specifically address the problem of ice navigation is quickly rising.

Despite the great success of automated route planning and navigation for land vehicles, such as car navigators or online driving directions services, maritime navigation and route planning remains a mostly manual task, which requires extensive training and great vigilance to be carried out safely and efficiently. Various software systems have been developed to assist the crew in routing and navigation functions, such as the Jeppesen Vessel and Voyage Optimization Solution (VVOS), as well as onboard Decision Support Systems (DSS), which mostly aim to assist the captain in mitigating risks to the vessel [2]. A large gap still remains, however, between the available information sources and the ability to utilize them in an efficient manner. Previous research has attempted to narrow this gap, most notably the European project, Ice Ridging Information for Decision Making in Shipping Operations (IRIS) [3]. Most of the current commercial solutions and systems, however, are not applicable or optimized for navigation in ice-covered waters.

Furthermore, there are many documented instances where poor navigation choices at sea, either due to lack of available information or poor training, have led to suboptimal routes or even unsafe routes to be used in ice-covered waters. For example, in March 2010 around fifty vessels were stuck in Baltic Sea ice, including six passenger ships, requiring extensive icebreaker assistance to free the stuck ships. During this episode a large passenger ship, MS Amorella, collided with another passenger ship, MS Finnfellow [4]. Countries with high latitude sea borders, such as Finland, Sweden, and Russia, expend significant resources to maintain and operate icebreaker fleets, in order to assist vessels that become stuck in sea ice, as well as to produce channels in the ice for easier transit (see e.g. [5]).

The goal of the present work is develop a method that takes into account available information about sea ice conditions and icebreaker operations, in order to determine optimal routes, both in terms of minimizing crossing time and minimizing risk of ships getting stuck or damaged due to severe ice conditions. Due to the Baltic Sea being one of the busiest waterways in the world that experiences significant ice coverage every winter, this paper will focus on examples from this region.

The paper is organized as follows: We first review related work especially in the areas of maritime route optimization and ice navigation. In Section III, we describe (1) the model we used for the sea environment, including a model of sea ice conditions, (2) the assumptions we made about how ships can maneuver in the sea environment, and (3) our model of ship transit speeds as a function of ice conditions. In Section IV, we describe icebreaker operations, in order to elucidate the role they play in route planning. Section V presents the route optimization method, which utilizes all the elements discussed

This work was supported in part by the Finnish Funding Agency for Innovation (Tekes) under the ARCSAT project.

in Sections III-IV. In Section VI, we discuss results of the route optimization method, including preliminary validation of the results based on historical route information. Finally, we conclude our work in Section VII, summarizing the main benefits of our proposed method and describing areas of future work that still remain.

II. Related Work

Research related to maritime route optimization can be traced back primarily to a seminal 1960 paper by Hanssen and James, which describes a system developed by the United States Hydrographic Office to optimize transoceanic crossings, utilizing predictions of wind, waves and current [6]. In the years following, many different route optimization methods have been proposed, however, few of them are applicable to navigation in ice-covered waters. One notable exception is [3]. This method described in this paper employed a cost function based on ship speed and navigation restrictions, such as fairways and shallow water. The authors' ship speed estimation technique took into account ice conditions based on three ice parameters, including level ice thickness, ridged ice thickness, and ice concentration. They evaluated the use of three different optimization techniques for minimizing this cost function, in order to compute an optimal route, including Powell's method, the polytope method, and simulated annealing. The method the authors chose for prototype implementation (Powell's method), however, cannot guarantee to find a global minimum of the cost function [7]. In other words, an initial guess of the route is supplied, and the method will find a local minimum of the cost function with respect to the initial guess.

III. Development of a Maritime Model

The purpose of this section is to describe the model of the sea and ship motion that was used in this study. The model can be divided up into four parts: (1) sea spatial model, (2) ship maneuverability model, (3) sea ice model, and (4) ship performance model.

A. Sea Spatial Model

The main difference between a car navigation application and a maritime navigation application is that a car is assumed to only travel on roadways, whereas a ship can generally travel anywhere where the sea depth is greater than the ship's draught and in any direction within this area, provided only that it stays within such area. This degree of freedom makes the route planning problem more complex because there is an infinite number of possible routes between any two given points. Despite this fact, we note that similar routes will have similar crossing times. Thus, we can divide the sea area into a finite number of points and use this set of points as an approximate model for the actual sea.

Let S represent a set of points in a discrete model of the sea and $m = |S|$ represent the number of points in this set. For each point $p \in S$, there will be associated information, such as its latitude and longitude, the sea depth at that point, as well as ice-related information, as will be discussed below. The complete set of points S and this associated metadata are the primary components of the model of the sea. We refer to this part of the modeling process as a *discretization of the sea*.

There are a number of different ways to discretize a given sea area, but in this work we chose the most simple approach, which is to define a uniform grid of points with a suitable grid spacing covering the whole sea area. The drawback of this approach is that if the grid spacing is small (i.e. a dense grid of points), there will be a large number of points, which leads to more computation required for the routing algorithm. If the grid spacing is large, then the required computation will be less, but the resulting model will not accurately represent the actual sea. That is, there will only be a limited number of positions and directions in which the ship can travel. Fig. 1 shows an example grid applied across the Baltic Sea, where the grid points are shown in red. In this example, the grid spacing is relatively sparse (~20 km between each grid point). The actual grid spacing used in this work was approximately 1 km. This value, however, can be adjusted according to the requirements of the end user.

Also shown in Fig. 1 is the concept of a *depth mask*, which is a binary image representing where a ship can and cannot travel based on the depth of the sea. If a given ship has a draught of r, then the value of the depth mask is defined at each coordinate in the image as follows:

$$D_{ij} = \begin{cases} 1 & \text{for } d_{ij} > r \\ 0 & \text{for } d_{ij} \le r \end{cases} \tag{1}$$

where D_{ij} is the value of the depth mask at coordinate (i, j), d_{ij} is the depth of sea at coordinate (i, j), and r is the draught of the ship. If desired a safety factor can be applied, such that:

$$r = (1 + \varepsilon)r_{\text{actual}} \tag{2}$$

where r_{actual} is the ship's actual draught and ε is a positive value representing the safety factor. In Fig. 1, a value of r = 10m was used. The white regions represent the areas where the depth mask has a value of 1, and the black regions represent a value of 0.

Note that because the depth mask is a function of a ship's draught, the resulting model of the sea is customized to a specific ship or class of ships. In practice, any implementation of this method should pre-compute a set of depth masks appropriate for the set of ships for which the routing software will be used. Thus, the software could apply the most suitable depth mask as a lookup table prior to start of route optimization.

B. Modeling Ship Maneuverability

To model how a ship can move among the grid points described above, we define a set of neighbors $\mathcal{N}$ and assume a ship can move from its current position (i.e. grid point) to one of its neighboring grid points defined by this set. For example, if the set of neighbors were defined as $\mathcal{N}$={north, south, east, west}, then the set of vectors from the current position to each of the neighbors (which we refer to as the set of direction vectors $\mathcal{V}$) would form the four cardinal compass directions. As a result, the ship could only move in these four cardinal directions. Since this is obviously not sufficient to accurately describe the real maneuverability of a ship, additional neighbors must be added to the set.

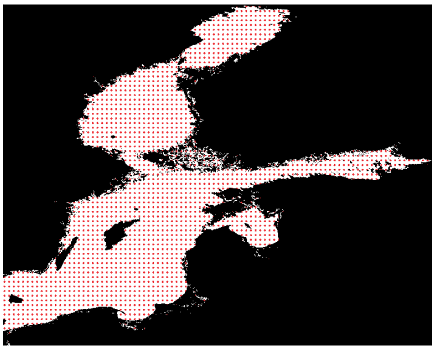


Fig. 1. Example of a depth mask for the Baltic Sea with a threshold value *r* set to 10 m. The black regions represent unnavigable areas. The red dots represent nodes in the discrete model of the sea. In the figure, a node spacing of about 20 km is shown, but in practical implementations a smaller spacing is generally used.

Let $n = |\mathcal{N}| = |\mathcal{V}|$ represent the number of neighbors defined in the ship maneuverability model. In this work, we used $n = 56$ and defined the sets $\mathcal{N}$ and $\mathcal{V}$ as shown in the Appendix (i.e. Table III)[1]. Note that because a uniform rectangular grid of points was used, the length of each of the vectors in set $\mathcal{V}$ is not necessarily the same. In fact, in defining these sets, the aim was to keep the difference in compass directions between the direction vectors as close to equal as possible. Fig. 2 illustrates the sets $\mathcal{N}$ and $\mathcal{V}$ defined and used in this work, where the red squares represent the neighbors (relative to the central square drawn in green) and the blue arrows represent the direction vectors. The average angle between adjacent direction vectors is 6.43° with a minimum separation of 4.97° and maximum of 8.13°.

Thus, the set defined by $\mathcal{N}$ represents a discretization of the maneuverability of a ship at sea, similar to how the set of grid points S discretized the sea into a finite set of points. The choice of n is a tradeoff between having an accurate model and the computational complexity of the routing algorithm which will use the model. Similar to the choice of grid spacing, n can be adjusted according to the requirements of the end user.

Together the sets S and $\mathcal{N}$ constitute a graph, consisting of m nodes and a set of edges between the nodes, defined according to $\mathcal{N}$. In other words, $\mathcal{N}$ determines the connectivity of the graph. Because nodes on the edge of the sea boundary do not necessarily have n valid neighbors, the exact number of edges depends on the dimensions of the sea area. The upper bound for the number of edges, however, is m*n. Fig 3. shows a portion of the graph structure used in this work (i.e. n = 56). The blue lines show the edges drawn between nodes in the graph (which are shown in red). The figure has been zoomed in to cover only nine points because when zoomed further out, it quickly becomes difficult to discern the edges separately.

[1]Note that the set $\mathcal{V}$ is uniquely defined by the set $\mathcal{N}$. As such, a reference to $\mathcal{V}$ implies the corresponding set $\mathcal{N}$ and vice versa.

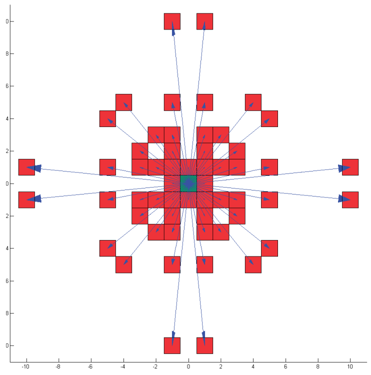


Fig. 2. The set of neighbors $\mathcal{N}$ and direction vectors $\mathcal{V}$ used to model ship maneuverability.

C. Modeling Sea Ice

The ice data were obtained from the hindcasts generated using the HELMI multicategory sea-ice model, developed at the Finnish Meteorological Institute; for details see [8]. The model resolves ice velocity, internal ice stress, ice concentration and ice thickness. Thickness is resolved for seven categories: five level ice categories, plus rafted ice and ridged ice. The ice model is discretized in a curvilinear coordinate system called a c-crid, a common solution when there are both fields of velocities and velocity-dependent properties to be solved. The grid has 415 nodes from west to east and 556 nodes from south to north. The SW lower corner coordinates are 56.74° N 16.72° E, NE corner coordinates 65.99° N 30.48° E, and the increment is 1/30 degrees eastwards and 1/60 degrees northwards. This is approximately 1 NM in both directions at 60°N.

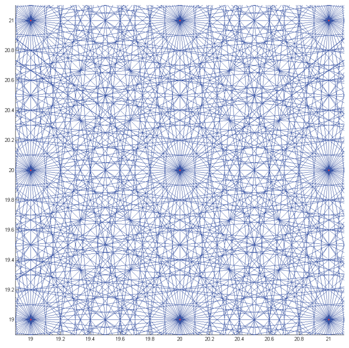


Fig. 3. The blue lines represent connections between nodes which are valid paths for a ship to travel. Only nine nodes are shown in the figure, whereas in an area the size of the Baltic Sea, there may be tens of thousands of nodes, depending on the chosen grid size.

Ice motion is determined by the momentum balance equation, which takes into account the Coriolis force, wind and water stresses, sea surface tilt term and internal friction of ice, which is the divergence of internal stress tensor. The magnitude of internal friction is used as the principal model variable to describe compression. It is to be noted that the viscous-plastic rheology does not describe elastic stresses and the internal stress arises from the interactions of moving ice. Forces arising in a static ice are included by assuming a negligibly slow viscous creep. Roughly, the internal friction term can be interpreted to describe the forces arising when ice floes are pushed and sheared against each other, or broken and heaped into ridges. Thus it is a good descriptor for the interaction between dynamical ice cover and an ice-going ship. This is manifested as ice forces against the ship hull and as the closing of channels, or other phenomena that navigators associate to compressive ice conditions. The internal friction magnitude has typical values ranging from 0 to 10 N/m^2. The magnitude acts as a proxy for ice compression, scaled to semi-empirical compression numeral 0-4, where 0 means no compression and 4 stands for extreme severe compression, see Table I. However, to estimate the actual local forces additional scaling arguments must be taken into account such as floe size and other ice cover geometry.

TABLE I. ICE COMPRESSION CONVERSION

Ice pressure obtained from the HELMI model [10^4 N]	Practical scale	
0 – 500	no significant compression	0
500 – 1000	mild pressure	1
1000 – 2000	moderate pressure	2
2000 – 3000	severe pressure	3
>3000	extreme severe pressure	4

D. *Modeling Ship Performance in Ice*

This chapter provides an outline of the adopted approach for estimating ship performance in ice conditions, which is used in the cost function of the path planning algorithm.

Various advanced theoretical approaches have been developed to estimate vessel performance in ice conditions, including analytical models [10-11] and simulation-based calculation procedures [12-13]. These typically require detailed information regarding the ship hull design, while being limited in application in real-world ice conditions as the mentioned models only consider level ice. Methods based on full-scale onboard data have been developed [14] and recently, a probabilistic model based on a combination of data from the Automatic Identification System (AIS) and ice information has been developed [15, 23].

In the current approach to vessel performance, an approach presented in Kotovirta et al. is applied [3]. While it has the limitation of only accounting for level ice, ice ridges and pack ice, its computational efficiency is beneficial in the context of route optimization. The methodology is based on a semi-empirical model by Lindqvist [16], as modified in La Prairie et al. [17] and Riska et al. [18]. In ridged ice fields, the ship transit speed $v_{i,eq}$ (m/s) is determined by numerically solving following force balance equation, where ice resistance and propulsion power are in equilibrium:

$$T_{\mathrm{net}}(v) = R_{\mathrm{tot}}\left(h_{\mathrm{i}}, h_{\mathrm{eq}}, v_{i,\mathrm{eq}}\right) \tag{3}$$

Here, T_{net} is the net available trust (kN), R_{tot} the total ice resistance (kN), h_i the level ice thickness (m) and h_{eq} the mean thickness of the ice rubble (m). h_i and h_{eq} are derived from the sea ice model (see Section IIC). The net available trust T_{net} is obtained from following simplified formula, see [19]:

$$T_{\mathrm{net}}(v) = \left(1 - \frac{1}{3}\frac{v}{v_{\mathrm{ow}}} - \frac{2}{3}\left(\frac{v}{v_{\mathrm{ow}}}\right)^2\right) T_{\mathrm{pull}} \tag{4}$$

Here, v is the average ship speed (m/s), v_{ow} the ship speed in open water (m/s) and T_{pull} the bollard pull force (kN), i.e. the available trust when a vessel is stationary with engines running:

$$T_{\mathrm{pull}} = K_{\mathrm{e}}\left(P_{\mathrm{s}} D_{\mathrm{p}}\right)^{2/3} \tag{5}$$

with propulsion power P_s, propeller diameter D_p and quality efficient for bollard pull K_e.

While not fully physically correct, the total ice resistance R_{tot} is taken as a sum of resistance components stemming from breaking through level ice R_i and passing through ridge rubble R_r:

$$R_{\mathrm{tot}}(v) = R_{\mathrm{i}}(h_{\mathrm{i}}, v) + R_r\left(h_{\mathrm{eq}}, v\right) \tag{6}$$

Riska et al. [18] assume a linear relation for the relation between level ice thickness and speed:

$$R_{\mathrm{i}} = C_1 + C_2 v \tag{7}$$

with:

$$C_1 = f_1 \frac{1}{2\frac{T}{B}+1} B L_{\mathrm{par}} h_{\mathrm{i}} + (1 + 0.021\varphi)(f_2 B h_{\mathrm{i}}^2 + f_3 L_{\mathrm{bow}} h_{\mathrm{i}}^2 + f_4 B L_{\mathrm{bow}} h_{\mathrm{i}}) \tag{8}$$

and:

$$C_2 = (1 + 0.063\varphi)\left(g_1 h_{\mathrm{i}}^{1.5} + g_2 B h_{\mathrm{i}}\right) + g_3 h_{\mathrm{i}}\left(1 + 1.2\frac{T}{B}\right)\frac{B^2}{\sqrt{L}} \tag{9}$$

The resistance in ridges R_r is calculated as:

$$R_{\mathrm{r}} = 4C_3 h_{\mathrm{eq}}^2\left[B + 2C_\psi h_{\mathrm{eq}}\right](\mu_{\mathrm{h}}\cos\varphi_2 + \sin\psi\sin\alpha_2) + C_4 L_{\mathrm{par}} h_{\mathrm{eq}}^2 + C_5\left[\frac{LT}{B^2}\right]^3 h_{\mathrm{eq}} A_{\mathrm{wf}} F_{\mathrm{n}}^2 \tag{10}$$

Here, μ_h is the friction between hull and ice, φ_2 the angle with buttock line at B/4 with the horizontal, α_2 the waterline entrance angle at B/4, L_{par} the length of the parallel midbody, L, B and T the vessel length, width and draft, A_{wf} the bow waterplane area, and F_n the Froude number. All necessary constants are given for the considered vessel in Table II.

The effect of ice concentration on ship performance in ice, focusing on attainable speed and hull loads, has been studied using a simulation model of ship dynamics in ice, applying analytical formulae for ice resistance [20]. The adopted approach in the route optimization is based on empirical formulae and a set of simplifying assumptions, see [3].

TABLE II. SHIP PERFORMANCE CONSTANTS

Constant	Value	Unit
f_1	230	N/m^3
f_2	4580	N/m^3
f_3	1470	N/m^3
f_4	290	N/m^3
g_1	18900	$N/(m^{2.5}/s)$
g_2	670	$N/(m^3/s)$
g_3	1550	$N/(m^{3.5}/s)$
C_3	850	N/m^3
C_4	42	N/m^3
C_5	1300	N/m^3
C_ψ	$0.047\psi - 2.115$	-
L	193.7	m
L_{par}	136	m
B	30.2	m
T	12.0	m
μ_h	0.05	-
φ_2	0.7156	rad
α_2	0.6981	rad
A_{wf}	496.79	m^2

With an ice concentration C less that C_0, the vessel is assumed to manoeuver around ice floes and sail at the open water speed v_{ow}. For ice concentrations higher than C_1, the ship operates at $v_{i,eq}$, the speed through the ridged ice field. Between C_0 and C_1, the resulting transit speed v_{tr} is assumed as a linear combination of v_{ow} and $v_{i,eq}$:, assuming C_0=70% and C_1=95%:

$$v_{\mathrm{tr}} = \begin{cases} v_{\mathrm{ow}} & C \le C_0 \\ \frac{(C_1 - C)v_{\mathrm{ow}} + (C - C_0)v_{\mathrm{i,eq}}}{(C_1 - C_0)} & C_0 < C < C_1 \\ v_{\mathrm{i,eq}} & C \ge C_1 \end{cases} \quad (11)$$

While the vessel performance is largely based on semi-empirical formulations and does not account for effects of compression, which can be significant [21], the approach leads to a reasonable estimate of total transit time [3, 17].

IV. WINTER MARITIME SHIPPING IN THE BALTIC SEA

In spite of the sea ice in wintertime, most of the ports around the Baltic Sea are kept open all year around for shipping. This is possible thanks to specially built icebreakers (8 Finnish, 9 Swedish, 2 Estonian, 3 larger and several smaller Russian icebreakers operate in the Northern Baltic Sea) that assist the merchant ships through the ice field.

Cooperation between the nations around the Baltic Sea enables more efficient use of the icebreakers, as many ship routes have common parts irrespective of the port of destination. Especially in the Bay of Bothnia and the Sea of Bothnia, the Finnish and Swedish icebreakers organize their assistance activities as a joint fleet and the operating costs are shared between the Finnish and Swedish governments based on actual assistance provided and the port of destination of the assisted ships.

To ensure efficient and safe navigation, the ships bound for ports surrounded by ice have to fulfill some minimum requirements regarding capability to navigate through the ice, both from a safety (e.g. hull strength) and efficiency point-of-view (e.g. machine power). These requirements have been formulated into ice class rules defined by classification societies. The Ice Class notation given by a Classification society can be mapped to an Equivalent Finnish/Swedish ice class using mapping tables issued by the Finnish and Swedish authorities. The port-specific ice class restrictions (Ice Restrictions for short) are determined by the maritime Administration in the country of the destination port and are updated depending on the ice conditions and forecasts. These ice restrictions are published on the Internet and also included in the daily ice charts issued by the national Ice Services.

A ship coming from the southern part of the Baltic Sea, bound for a Swedish or Finnish port having a valid ice restriction, is obliged to contact a coordinating point announcing, among other things, the port of destination of the ship. The coordinating point then checks that the ship fulfils the ice class requirements and tells the ship the coordinates of the next ice waypoint and also which icebreaker to contact. Often, when the ship is considered capable of navigating independently for part of the voyage (not necessarily having to be escorted by an icebreaker), the ship is given several ice waypoints defining a coarse route through the ice field. The details on how to navigate through the ice field between the ice waypoints, which may be separated by tens of nautical miles, are left to the ship itself. The waypoints are often communicated to the ship over VHF, but sending the ice waypoints in digital form to the ship using e-mail or an AIS addressed message, is becoming more and more common. Icebreakers give lowest priority to ships which do not follow these waypoints, the result of which is that ships in general follow them whenever there is a significant chance of requiring icebreaker assistance.

The ice waypoints are determined by the icebreakers and registered in the Icebreaker information system, IBNet, maintaining the overall set of valid points. The waypoints are updated when needed as a consequence of changing ice conditions. Some of the waypoints may be valid for weeks, but often they are adjusted with 2 -3 days intervals. Satellite images (especially Synthetic Aperture Radar images), ice drift forecasts, ship speed monitoring (i.e. how ships have been able to proceed through the ice field) and ice observations from other icebreakers are used as background information. This information is then combined by the experienced icebreaker captain into a set of waypoints that are registered in the IBNet system and communicated to the ships. A tool that could simulate different routing alternatives for ships with varying capabilities could help the icebreakers to determine and adjust the ice waypoints as part of an optimal (safe and efficient) winter navigation system.

V. Route Optimization

The classic algorithm for route optimization, also known as pathfinding, using graph structures is called A* [22]. A* finds an optimal path according to a *cost model* and subject to the constraints imposed by the graph structure. In our case the graph structure includes the set of points S resulting from the discretization of the sea, as described in Section IIIA, plus the set of edges connecting the grid points, as described in Section IIIB.

For the details of the A* algorithm, we refer to reader to [Hart et al., 1968]. In short the algorithm begins at the origin point, computes the cost to travel to that point's neighbor nodes (according to the cost model), and then sorts these neighbor nodes in priority order according to the following heuristic:

$$F_{ij} = G_{ij} + H_{ij} \tag{12}$$

where F_{ij} is the value used for sorting, G_{ij} is the cost to get to the node with coordinates (i, j), and H_{ij} is a heuristic estimate of the cost to from node (i, j) to the destination. After computing the value of F_{ij} for each neighbor node, the algorithm chooses the lowest F_{ij} among all the nodes (i, j) on the so-called "open list" and then computes F_{ij} for the neighbor nodes with respect to the chosen point (i, j), known as the current parent node.

The algorithm repeats this process recursively until the destination node has been reached. As each neighbor node is "visited" by the algorithm, the cost to get to that node from the current parent node must be saved (i.e. G_{ij}), as well as the coordinates of that parent node. Because most nodes have multiple parent nodes from which they can be visited, occasionally the algorithm will revisit nodes already on the open list. As G_{ij} is recalculated, if it is less than the stored value of G_{ij}, the old value as well as the coordinates of the parent node are replaced with the current values. This is because this new path to the node (i, j) represents the shortest path to that node among all attempted paths. In this way, the algorithm is storing a set of candidate sub-paths between the origin and the visited nodes in S. After the algorithm reaches the destination node, it can trace back through the stored parent nodes to find the best overall path between the origin and the destination.

Next, we define the cost model used by the function G_{ij}. It is a function of both the geometry (i.e. distance between nodes in the graph), the ice conditions (i.e. estimated speed that the ship can travel under given ice conditions, as derived in Section IIIC), and the distance to the nearest icebreaker waypoint (as discussed in Section V), namely:

$$G_{ij} = \sum_{k=\text{origin}}^{(i,j)} \left(\frac{d_{k\to\text{next}}}{v(k, d_{\text{IB}})} \right) \tag{13}$$

representing a Riemann sum estimating the time it takes to travel from the origin to the destination; (i,j) are the coordinates of the point immediately before the destination point; the summand contains $v(k, d_{ib})$, the estimated ship speed at point (a, b) along the route and at distance d_{IB} from the icebreaker waypoints, and $d_{k\to\text{next}}$, the length of the line segment between point k to point k+1. The function $v(k, d_{\text{IB}})$ is defined as follows:

$$v(k, d_{\text{IB}}) = \begin{cases} v_{\text{tr}}(k) & \text{for } d_{\text{IB}} > c \\ v_{\max} & \text{for } d_{\text{IB}} \le c \end{cases} \tag{14}$$

where v_{tr} is defined according to Eq. 11 and $v_{\max}$ is the maximum transit speed, normally corresponding to nominal speed of the modeled ship in open water conditions. To determine this value at any given point (a, b), we linearly interpolated the two-dimensional grid of ship transit speeds described in Section IIIB. d_{ib} is defined as the shortest distance between the point (a, b) and the line formed by the icebreaker waypoints, and c is a threshold value set to control how close to the icebreaker waypoints the ship should come before it is to receive icebreaker assistance. In this study, we used a value for c of approximately 2 km.

The remaining element to define is the heuristic function H_{ij}. We used the following:

$$H_{ij} = \frac{\sqrt{(x_i - x_{\text{dest}})^2 + (y_j - y_{\text{dest}})^2}}{v_{\max}} \tag{15}$$

where x_i is the x-coordinate of the node (i, j), x_{dest} is the x-coordinate of the destination, y_j is the y-coordinate of the node (i, j), y_{dest} is the y-coordinate of the destination, and $v_{\max}$ is the maximum. We used a local Cartesian coordinate system in order to speed up computation.

VI. Validation of Results and Discussion

Although a ship's crew has great freedom in choosing its route to their intended destination, in practice, especially in the relatively narrow Baltic Sea, ships follow more restricted pathways. Under ice conditions, ships need to follow icebreaker instructions (i.e. waypoints), and thus they have even less route options. Despite these constraints, there is much to be gained by optimizing the route based on the best available knowledge about the ice conditions. We see that the Baltic Sea area is ideal for development and testing of *ice-aware* route optimization, as it is one of the busiest ice covered sea areas, and therefore there are statistical data about ship transits available for validation, as well as model and observational data about the ice conditions.

Nonetheless, one of the challenges in this research topic is the difficulty of validating whether the chosen route optimization algorithm actually produces optimal results. That is, if the goal is to find a route that minimizes the crossing time, how can one show that the generated route gives the minimum time among all possible routes? Earlier work has shown that if a well-defined (i.e. admissible) heuristic H_{ij} is used, then A* will find the globally optimal path between the origin and destination [24]. H_{ij} is considered admissible as long as it never overestimates the actual cost to get to the destination from a given node (i, j). Because the Euclidean distance is always shorter than the geodetic distance and because the ship can never travel faster than $v_{\max}$, H_{ij} is a conservative estimate of the shortest crossing time between the node (i, j) and the destination. For this reason, H_{ij} always underestimates the actual minimum cost to travel from node (i, j) to the destination, and is therefore admissible.

The question remains, however, does the cost function used to generate the route accurately estimate the cost (in terms of crossing time) of traveling along a given route? This question is not easily answered, and it is where historical maritime traffic data can play a key role. As many ships are equipped with an Automatic Identification System (AIS) transponder, which broadcasts the position, heading, and speed of the ship among other information. AIS data is received not only by other ships in the vicinity of a broadcasting ship but also receivers designed to monitor all the maritime traffic in a given area. This data is then made available by various commercial providers, such as AIS Live, or by governmental authorities.

Fig. 4 below compares the generated optimal route with historical AIS routes under the same ice conditions (i.e. same day) as used to generate the route. The sea area shown is primarily the Gulf of Finland with the Finnish coast on the top half of the images and the Estonian coast on the bottom. The AIS routes are shown in yellow, and the generated route in blue. The icebreaker waypoints are shown with a blue star-shaped marker. The calculated transit speeds (i.e. Eq. 11) are indicated by the background color of the sea areas, where the highest speeds are shown in magenta and the lowest speeds shown in cyan. It is apparent that there is significant difference between the historical routes and the generated routes. At this phase in our research, it is not possible for us to say which one of the two routes for each case is more optimal. We can only make a few conjectures and identify aspects requiring further development.

In both cases presented in Fig. 4 the ship represented by the AIS data took a more direct route to the first icebreaker waypoint but spent more time under heavy ice conditions. One possibility is that the crew was not aware of area of open water near the Estonian coast. Another possibility is that they simply believed that the more direct route to the first waypoint would be faster. It is also possibility is that the more direct route actually was faster, compared to the generated route. If this were the case, then it would follow that our cost function is inaccurate. The only way we could determine whether this is the case, however, would be if there was additional AIS data for a similar ship over the same time period, where the ship took this more southerly route. Unfortunately, we have not yet identified a suitable set of AIS data to disambiguate these various possibilities.

One deficiency of the currently implemented method is that it places no constraints on the number and severity of course changes in the calculated route. Ship crews generally would prefer fewer course changes, especially if there is no significant advantage to making such changes. In addition, ships have physical constraints on the rate of course change. It has been suggested that a penalty could be added to the cost function for course changes, the value of which would be proportional to the magnitude of the course change. This would result in straighter routes to be generated, balancing the benefits of changing course with the operational and physical constraints they impose. This approach has not yet been implemented, but it is planned in future work.

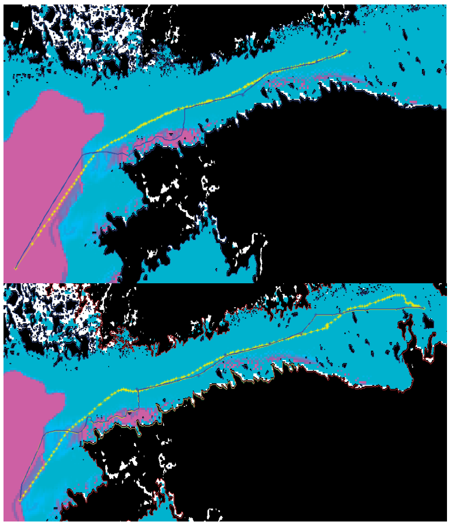


Fig. 4. Comparison of generated routes with historical routes under the same ice conditions. Historical routes, based on AIS data, are shown in yellow with generated routes shown in blue. The cyan and magenta coloring represent transit speed with magenta indicating speeds near those of open water sailing.

VII. Conclusions

This paper has demonstrated the feasibility of ice-aware maritime route optimization. That is, it appears that existing route optimization algorithms, namely the A* algorithm, can be used to minimize the costs associated with sailing through ice-covered waters. Furthermore, the effect of icebreaker assistance can be taken into account by adjusting the cost associated with sailing in the areas between a set of waypoints published by the icebreaker service.

Further work is needed to validate that the results produced using the presented method are indeed optimal. Validation efforts should include further analysis of historical data, as well as simulator-based studies and actual testing of routes at sea. The goal of minimizing crossing time is just one aspect among several that must be taken into account in maritime route planning with others including operational efficiency, safety, and reliability. If the ice information or the cost function used to generate the route are faulty, then the algorithm is prone if not likely to generate suboptimal routes. Thus, it is clear that results can be improved through further work to improve the quality of ice forecasts and further development of models of ship performance in ice. For example, the HELMI sea-ice model provides greater detail than is used in the current model of ship transit speed. Future work should focus on incorporating additional factors, such as ice compression, in order to more accurately predict ship speed under compressive ice conditions. Lastly, future work should also compare the results generated with the presented method to results produced using other algorithms, such as Powell's method.

VIII. Appendix

The following table shows the complete set of neighbors $\mathcal{N}$ and direction vectors $\mathcal{V}$ previously described in Section IIIB. The coordinates reference the relative position from the central node, e.g. (1,-1) is one node to the east and one node to the south from the central node. See Fig. 2 for an illustration.

TABLE III. Set of neighbors and direction vectors employed

#	Coordinates (x,y)	Compass Angle (deg)	#	Coordinates (x,y)	Compass Angle (deg)
1	(0,1)	0	29	(3,2)	56.310
2	(0,-1)	180	30	(3,-2)	123.690
3	(1,0)	90	31	(-3,2)	303.690
4	(-1,0)	270	32	(-3,-2)	236.310
5	(1,1)	45	33	(4,5)	38.6598
6	(1,-1)	135	34	(4,-5)	141.340
7	(-1,-1)	225	35	(-4,5)	321.340
8	(-1,1)	315	36	(-4,-5)	218.660
9	(1,2)	26.565	37	(5,4)	51.340
10	(1,-2)	153.435	38	(5,-4)	128.660
11	(-1,2)	333.435	39	(-5,4)	308.660
12	(-1-,2)	206.565	40	(-5,-4)	231.340
13	(2,1)	63.435	41	(5,1)	78.690
14	(2,-1)	116.565	42	(5,-1)	101.310
15	(-2,1)	296.565	43	(-5,1)	281.310
16	(-2,-1)	243.4349	44	(-5,-1)	258.690
17	(3,1)	71.565	45	(1,5)	11.310
18	(3,-1)	108.435	46	(1,-5)	168.690
19	(-3,1)	288.435	47	(-1,5)	348.690
20	(-3,-1)	251.565	48	(-1,-5)	191.310
21	(1,3)	18.435	49	(1,10)	5.711
22	(1,-3)	161.565	50	(1,-10)	174.289
23	(-1,3)	341.565	51	(-1,10)	354.289
24	(-1,-3)	198.435	52	(-1,-10)	185.711
25	(2,3)	33.690	53	(10,1)	84.289
26	(2,-3)	146.310	54	(10,-1)	95.711
27	(-2,3)	326.310	55	(-10,1)	275.711
28	(-2,-3)	213.690	56	(-10,-1)	264.289

References

[1] Bowditch, N., The American Practical Navigator, National Imagery and Mapping Agency, 2002.

[2] U. Nielsen and J. Jensen, "A novel approach for navigational guidance of ships using onboard monitoring systems," Ocean Engineering, Vol. 38, No. 2-3, 2011, pp. 444-455.

[3] Kotovirta, V., Jalonen, R., Axell, L., Riska, K., & Berglund, R., "A system for route optimization in ice-covered waters," Cold Regions Science and Technology, Vol. 55(1), 2009 pp. 52–62.

[4] BBC News, "Dozens of ships freed from Baltic Sea ice", 5 March 2010. Available: http://bbc.in/1ejqwZJ

[5] Baltic Icebreaking Management, Baltic Sea Icebreaking Report 2009-2010, Available: http://bit.ly/1cpegvx.

[6] G. Hanssen and R. James, "Optimum ship routing", Journal of Navigation, Vol. 13, No. 3, July 1960, pp. 253-272.

[7] M. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," Computer Journal, Vol. 7, No. 2, 1964, pp. 155–162.

[8] Haapala, J., Lönnroth, N., & Stössel, A., "A numerical study of open water formation in sea ice," Journal of Geophysical Research: Oceans, Vol. 110, No. C9, 2005.

[9] Lehtiranta, J., Lensu, M., & Haapala, J., Ice model validation on local scale. Helsinki, 2012.

[10] Neagle, J. N., "Ice resistance prediction and motion simulation for ships operating in the continuous mode of icebreaking," Doctor of Philosophy Thesis, The University of Michigan, Ann Arbor, MI, 1980.

[11] Valanto, P., "The resistance of ship in level ice," Transactions of the Society of Naval Architects and Marine Engineers, Vol. 109, 2001, pp. 53–83.

[12] Liu, J., Lau, M., & Williams, M. F. "Numerical implementation and benchmark of ice-hull interaction model for ship manoeuvring simulations", NRC Publications Archive, Vancouver, BC, Canada, 2008, pp. 215–226.

[13] Sawamura, J., Tsuchiya, H., Tachibana, T., & Osawa, N., "Numerical modeling for ship maneuvering in level ice," Proceedings of 20th International Symposium on Ice (IAHR) Lahti, Finland. 2010, pp. 1–12.

[14] Dick, R. A., Prior, A. D., & Peirce, T. H., "Resistance and propulsion in ice using system identification techniques," Transactions of the Society of Naval Architects and Marine Engineers, Vol. 103, 1995, pp. 237–254.

[15] Montewka, J., Kujala, P., Goerlandt, F., Lensu, M., & Haapala, J., "Modelling a ship performance in dynamic ice. Part I - transforming data into information," Scientific Journal of Warsaw University of Technology, Transport, Vol. 95, 2013, pp. 359–368.

[16] Lindqvist, G., "A straightforward method for calculation of ice resistance of ships," Luleå University of Technology, 1989, pp. 722–735.

[17] La Prairie, D., Wilhelmson, M., & Riska, K., "A transit simulation model for ships in Baltic Ice conditions," Memo No. 200, Espoo, Finland: Helsinki University of Technology, 1995.

[18] Riska, K., Wilhelmson, M., Englund, K., & Leiviskä, T., "Performance of merchant vessels in ice in the Baltic," Espoo, Finland: Helsinki University of Technology, 1997.

[19] Juva, M., & Riska, K., "On the power requirement in the Finnish-Swedish ice class rules" Research Report No. 53 of the Winter Navigation Research Board, Espoo, Finland: Helsinki University of Technology, 2002.

[20] Krupina, N., "Results of time-domain stochastic simulation of ice loads on ship hull from broken ice," Increasing the safety of icebound shipping - Vol. 2, Scientific Report No. 302 of the Helsinki University of Technology Ship Laboratory, Espoo, Finland: Helsinki University of Technology, 2007.

[21] Kaups, K., "Modeling of the ship resistance in compressive level ice," Master Thesis, Espoo, Finland: Aalto University, 2011.

[22] Hart, P. E., Nilsson, N. J., and Raphael, B. "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, Vol. 4, No. 2, 1968, pp. 100–107.

[23] Montewka, J., Sinclair H. "Modelling a ship performance in dynamic ice. Part II - transforming information into knowledge," Scientific Journal of Warsaw University of Technology, Transport, Vol. 95, 2013, pp. 369–382.

[24] R. Dechter and J. Pearl. "The Optimality of A*," In: L. Kanal and V. Kumar, eds., Search in Artificial Intelligence. Springer-Verlag, 1988.

Suomen Geodeettisen laitoksen julkaisut:
Veröffentlichungen des Finnischen Geodätischen Institutes:
Publications of the Finnish Geodetic Institute:

97. G.P. Arnautov, Ye N. Kalish, A. Kiviniemi, Yu F. Stus, V.G. Tarasiuk, S.N. Scheglov: Determination of absolute gravity values in Finland using laser ballistic gravimeter. Helsinki 1982. 18 pages.
98. Leena Mikkola (editor): Mean height map of Finland. Helsinki 1983. 3 pages. 1 map.
99. Mikko Takalo and Jaakko Mäkinen: The Second Levelling of Finland for Lapland. Helsinki 1983. 144 pages.
100. Jussi Kääriäinen: Baseline Measurements with invar wires in Finland 1958-1970. Helsinki 1984. 78 pages.
101. Raimo Konttinen: Plate motion studies in Central Asia. Helsinki 1985. 31 pages.
102. Raimo Konttinen: Observation results. Angle measurements in 1979-1983. Helsinki 1985. 30 pages.
103. J. Kakkuri, T.J. Kukkamäki, J.-J. Levallois et H. Moritz: Le 250^{e} anniversaire de la mesure de l'arc du meridien en Laponie. Helsinki 1986. 60 pages.
104. G. Asch, T. Jahr, G. Jentzsch, A. Kiviniemi and J. Kääriäinen: Measurements of Gravity Tides along the "Blue Road Geotraverse" in Fennoscandia. Helsinki 1987. 57 pages.
105. Jussi Kääriäinen, Raimo Konttinen, Lu Qiankun and Du Zong Yu: The Chang Yang Standard Baseline. Helsinki 1986. 36 pages.
106. E.W. Grafarend, H. Kremers, J. Kakkuri and M. Vermeer: Adjusting the SW Finland Triangular Network with the TAGNET 3-D operational geodesy software. Helsinki 1987. 60 pages.
107. Matti Ollikainen: Astronomical determinations of latitude and longitude in 1981-1983. Helsinki 1988. 37 pages.
108. Markku Poutanen: Observation results. Angle measurements in 1967-1973. Helsinki 1988. 35 pages.
109. Jussi Kääriäinen, Raimo Konttinen and Zsuzsanna Németh: The Gödöllö Standard Baseline. Helsinki 1988. 66 pages.
110. Jussi Kääriäinen and Hannu Ruotsalainen: Tilt measurements in the underground laboratory Lohja 2, Finland, in 1977-1987. Helsinki 1989. 37 pages.
111. Mikko Takalo: Lisäyksiä ja korjauksia Suomen tarkkavaaitusten linjastoon 1977-1989. Helsinki 1991. 98 sivua.
112. Raimo Konttinen: Observation results. Angle measurements in the Pudasjärvi loop in 1973-1976. Helsinki 1991. 42 pages.
113. Raimo Konttinen, Jorma Jokela and Li Quan: The remeasurement of the Chang Yang Standard Baseline. Helsinki 1991. 40 pages.
114. Jussi Kääriäinen, Raimo Konttinen and Markku Poutanen: Interference measurements of the Nummela Standard Baseline in 1977, 1983, 1984 and 1991. Helsinki 1992. 78 pages.
115. Juhani Kakkuri (editor): Geodesy and geophysics. Helsinki 1993. 200 pages.
116. Jaakko Mäkinen, Heikki Virtanen, Qiu Qi-xian and Gu Liang-rong: The Sino-Finnish absolute gravity campaign in 1990. Helsinki 1993. 49 pages.
117. Raimo Konttinen: Observation results. Geodimeter observations in 1971-72, 1974-80 and 1984-85. Helsinki 1994. 58 pages.
118. Raimo Konttinen: Observation results. Angle measurements in 1964-65, 1971, 1984 and 1986-87. Helsinki 1994. 67 pages.
119. Jorma Jokela: The 1993 adjustment of the Finnish First-Order Terrestrial Triangulation. Helsinki 1994. 137 pages.
120. Markku Poutanen (editor): Interference measurements of the Taoyuan Standard Baseline. Helsinki 1995. 35 pages.
121. Jorma Jokela: Interference measurements of the Chang Yang Standard Baseline in 1994. Kirkkonummi 1996. 32 pages.
122. Olli Jaakkola: Quality and automatic generalization of land cover data. Kirkkonummi 1996. 39 pages.
123. Matti Ollikainen: Determination of orthometric heights using GPS levelling. Kirkkonummi 1997. 143 pages.
124. Tiina Kilpeläinen: Multiple Representation and Generalization of Geo-Databases for Topographic Maps. Kirkkonummi 1997. 229 pages.
125. Jussi Kääriäinen and Jaakko Mäkinen: The 1979-1996 gravity survey and the results of the gravity survey of Finland 1945-1996. Kirkkonummi 1997. 24 pages. 1 map.
126. Zhitong Wang: Geoid and crustal structure in Fennoscandia. Kirkkonummi 1998. 118 pages.
127. Jorma Jokela and Markku Poutanen: The Väisälä baselines in Finland. Kirkkonummi 1998. 61 pages.
128. Markku Poutanen: Sea surface topography and vertical datums using space geodetic techniques. Kirkkonummi 2000. 158 pages
129. Matti Ollikainen, Hannu Koivula and Markku Poutanen: The Densification of the EUREF Network in Finland. Kirkkonummi 2000. 61 pages.
130. Jorma Jokela, Markku Poutanen, Zhao Jingzhan, Pei Weili, Hu Zhenyuan and Zhang Shengshu: The Chengdu Standard Baseline. Kirkkonummi 2000. 46 pages.
131. Jorma Jokela, Markku Poutanen, Zsuzsanna Németh and Gábor Virág: Remeasurement of the Gödöllö Standard Baseline. Kirkkonummi 2001. 37 pages.
132. Andres Rüdja: Geodetic Datums, Reference Systems and Geodetic Networks in Estonia. Kirkkonummi 2004. 311 pages.
133. Heikki Virtanen: Studies of Earth Dynamics with the Superconducting Gravimeter. Kirkkonummi 2006. 130 pages.
134. Juha Oksanen: Digital elevation model error in terrain analysis. Kirkkonummi 2006. 142 pages. 2 maps.
135. Matti Ollikainen: The EUVN-DA GPS campaign in Finland. Kirkkonummi 2006. 42 pages.
136. Annu-Maaria Nivala: Usability perspectives for the design of interactive maps. Kirkkonummi 2007. 157 pages.
137. Xiaowei Yu: Methods and techniques for forest change detection and growth estimation using airborne laser scanning data. Kirkkonummi 2007. 132 pages.
138. Lassi Lehto: Real-time content transformations in a WEB service-based delivery architecture for geographic information. Kirkkonummi 2007. 150 pages.
139. Pekka Lehmuskoski, Veikko Saaranen, Mikko Takalo and Paavo Rouhiainen: Suomen Kolmannen tarkkavaaituksen kiintopisteluettelo. Bench Mark List of the Third Levelling of Finland. Kirkkonummi 2008. 220 pages.
140. Eija Honkavaara: Calibrating digital photogrammetric airborne imaging systems using a test field. Kirkkonummi 2008. 139 pages.
141. Markku Poutanen, Eero Ahokas, Yuwei Chen, Juha Oksanen, Marita Portin, Sari Ruuhela, Heli Suurmäki (editors): Geodeettinen laitos – Geodetiska Institutet – Finnish Geodetic Institute 1918–2008. Kirkkonummi 2008. 173 pages.