DISSERTATION

Probabilistic Models in Building Automation: Recognizing Scenarios with Statistical Methods

Submitted at the Faculty of Electrical Engineering and Information Technology, Vienna University of Technology in partial fulfillment of the requirements for the degree of Doctor of Technical Sciences

under supervision of

Prof. Dr. Dietmar Dietrich
Institute of Computer Technology
Vienna University of Technology
1040 Wien

and

Prof. Dr. Brian M. O'Connell
Department of Computer Science
Department of Philosophy
Central Connecticut State University
New Britain, CT 06050

by

Dipl.-Ing. Dietmar Bruckner
Matr.Nr. 9925419
Kreuzberg 97, 3922 Großschönau

12. Jänner 2007 _____

additional thesis reader(s)

Dr. Brian Sallans
Smart Systems
Business Unit Intelligent Sensors and Neuroinformatics
Austrian Research Centers GmbH – ARC
1220 Wien

Kurzfassung

Gebäudeautomationssysteme finden speziell in den letzten Jahren eine immer größere Verbreitung sowohl in Nutzbauten als auch im Wohnbau. Die technischen Entwicklungen in den Bereichen Sensorik, Aktuatorik und Embedded Systems führen zu immer leistungsfähigeren und komplexeren Systemen. Diese Systeme erleben eine ständige Verbesserung in ihrer Fähigkeit zur Observierung von Aktivitäten in Gebäuden, was zu einer Ausweitung ihres Einsatzgebietes führt. Leider sind Automationssysteme mit einer großen Zahl von Parametern mit gängigen Methoden schwer zu beschreiben, bzw. kann man die Geschehnisse, die zu der aktuellen Situation geführt haben - den Kontext - schwer herauslesen.
In dieser Arbeit wird untersucht, inwiefern statistische Methoden geeignet sind, in (zukünftigen) Gebäudeautomationssystemen zum Erkennen fehlerhaften Verhaltens oder sogar zum Finden von semantischer und kontextueller Information aus den Sensordaten eingesetzt zu werden. Eine auf hidden Markov Modellen basierende, hierarchische Modellstruktur wird als Rahmenstruktur präsentiert. Dieser Rahmen kann mit beliebigen statistischen Modellen ausgestattet werden. Diese Modelle werden vorgestellt, verglichen und bewertet.
Die unteren Schichten in der hierarchischen Modellstruktur werden verwendet, um die Sensordaten selbst zu bewerten, während die oberen Schichten verwendet werden können, um semantische Interpretationen der Geschehnisse im Gebäude vorzunehmen.
In dieser Arbeit werden drei Implementierungen von Teilaspekten des Gesamtsystems gezeigt: In Kapitel 7 wird gezeigt, dass alleine die Verwendung von einfachen statistischen Modellen zur Beschreibung des Sensorverhaltens Vorteile für den Benutzer des Gebäudeautomationssystems hinsichtlich Treffsicherheit von Alarmen bringt. Weiters wird das Ergebnis des vorhandenen, regelbasierten Systems mit dem des modellbasierten Systems verglichen. Die zweite Implementierung zeigt, dass die Kombination von einfachen Modellen im Rahmenmodell eines hidden Markov Modells (HMM) verwendet werden kann, um das Verhalten des Gesamtsystems zu modellieren. Eine konkrete Anwendung dieser Idee - die Überwachung der Verkehrssituation in einem Tunnel - wird in Kapitel 8 vorgestellt. Weiters wird erläutert, wie ein HMM in Kombination mit entsprechenden Modellen für die Sensoren dazu verwendet werden kann, Muster in Sensordaten zu finden. Diese Muster bilden sich in der Modellstruktur in einer Weise ab, die es einem Menschen ermöglicht, Zustände des Modells - oder Kombinationen von Zuständen - semantisch zu interpretieren. Anhand eines Beispiels in einem Bürogebäude werden alle notwendigen Überlegungen und Algorithmen in Kapitel 6 präsentiert. Diese Modelle können als Basis für zukünftige - die Situation des menschlichen Benutzers erkennende - Systeme dienen.

Abstract

Building automation systems have also seen widespread distribution in private residences over the past few years. The ongoing technological development in the field of sensors, actuators as well as embedded systems leads to more and more complex and larger systems. These systems allow ever-better observations of activities in buildings with a rapid growing number of possible applications. Unfortunately, control systems with lots of parameters are hard to describe - and from a context-deriving view - hard to understand with standard control engineering techniques.

This thesis investigates how statistical methods can be applied to (future) building automation systems to recognize erroneous behavior and to extract semantic and context information from sensor data. A hierarchical model structure based on hidden Markov models is proposed to establish a framework. This framework can be equipped with any statistical model. Examples of models are given and their selection is justified. The lower levels of the model structure are used to observe the sensor values themselves whereas the higher levels provide a basis for the semantic interpretation of what is happening in the building.

In this work three implementations of different aspects of the overall system are presented: The advantages of utilizing simple statistical models to describe sensor values of single sensors in building automation systems are illustrated in chapter 7. This chapter also includes a comparison of the outcome of both systems, the traditional one and the statistical-model-based one. Further on, the combination of several models for single sensors within the framework of a HMM can be used to achieve an overview of the whole system behavior. An application of that idea is discussed in chapter 8. Finally, the HMM in combination with appropriate models for emissions can be used to find patterns in sensor values. This is done in a way that a human can interpret the model structure of the HMM and find states - or combinations of states - which have a semantic meaning. These models can be a basis for future context aware systems. The necessary ideas and algorithms are depicted in chapter 6 with the illustration of a concrete example located in an office building.

Vorwort

Das Gebäude. Wir wohnen in ihm, schlafen in ihm, essen in ihm, arbeiten in ihm. Wir verbringen unser Leben größtenteils innerhalb seiner Wände. Wir nehmen seine schützende Nestwärme gerne an. Und doch fordern wir noch mehr.
Genauso wie der Mensch selbst und alle Elemente der Natur macht auch das Gebäude eine evolutionäre Entwicklung durch. Wohin genau die Entwicklung geht, kann heute niemand abschätzen. Was wir aber erwarten, können wir ziemlich genau in Worte fassen: Das Gebäude verändert sich vom bloßen Dach über dem Kopf als Schutz vor dem Wetter in Richtung einer erweiterten, multifunktionalen Körperhülle. Beispiele aus der Biologie zeigen uns: wir sind nicht die Ersten. Einige Tiere haben ihren Bau bereits perfekt angepasst. Es gibt biologische Klimaanlagen, genau regulierte Luftfeuchtigkeit und -qualität, variable, dem Lebensabschnitt angepasst Raumgrößen und vieles mehr. Dies mag noch wenig spektakulär klingen, gibt es doch kein Bürogebäude mehr ohne diese Ausstattung. Doch wir wollen noch mehr.
Wir wollen die Funktion des Gebäudes von der Erweiterung der Körperhülle zum Lebensraum ausdehnen. Diese Erweiterung impliziert Zusatzfunktionalitäten wie eine Regulierungsfunktion verschiedener lebenswichtiger Parameter, Schutz vor äußeren Einflüssen inklusive Selbstheilungsfähigkeit und vieles mehr. Um diese Funktionen gewährleisten zu können, benötigt das Gebäude ein gewisses Bewusstsein seiner selbst im Sinne von Zuständen und Szenarien auf der Seite der Datenerfassung und die Möglichkeit der Beeinflussung von Parametern des System- oder Gebäudezustandes auf der Seite der Steuerung. Die Erkennung von Szenarien in Gebäuden erfordert die Entwicklung neuer Methoden abseits der Regelungstechnik um riesige Datenmengen in Echtzeit zu verarbeiten.
Diese Arbeit stellt einen Ansatz der Erfassung von Szenarien und Zuständen mit Hilfe statistischer Methoden - im Gegensatz zu vordefinierten Regeln - vor. Dabei wird das Gebäude und die Vorgänge in seinem Inneren im Betrieb möglichst vollständig sensorisch erfasst und mit Methoden der Statistik und des maschinellen Lernens ein Modell dieser Vorgänge erstellt. Das fertig gelernte Modell erlaubt die Erkennung von aktuellen Szenarien auf Basis der zum Lernen Verwendeten und damit auch eine Vorhersagemöglichkeit für die Aktivitäten in naher Zukunft.
In Verbindung mit Projekten, die "Symbole grounden" - also versuchen, reale Sensorwerte durch Regeln in einer Weise zu interpretieren, dass das technische System die gleichen "Symbole" zur Beschreibung der aktuellen Situation verwendet wie ein Mensch es tun würde - bildet das Ergebnis dieser Arbeit eine Basis für eine Fülle von Applikationen im Bereich des intelligenten Gebäudes.
Der Aufbau der Arbeit wurde folgendermaßen gewählt:
Kapitel 1 gibt eine Einführung in aktuelle Entwicklungen in der Gebäudeautomation, speziell in Richtung "intelligente Umgebung". Weiters wird das ARS-Projekt des Instituts vorgestellt und meine eigenen Ideen, Informationen aus Sensordaten zu gewinnen, in diesem Kontext beleuchtet. Die Grenzen herkömmlicher Systeme werden aufgezeigt, die Ziele dieser Arbeit beschrieben und auf mögliche soziale Implikationen von computerwissenschaftlichen Anwendungen eingegangen.
Leser mit Erfahrungen in den Bereichen Statistik, statistische Modelle und HMMs im Besonderen können die Kapitel 2 und 3 normalerweise überspringen. In Kapitel 2 werden Methoden der Wahrscheinlichkeitstheorie, im Speziellen das Bayes'sches Theorem, Wahrscheinlichkeitsdichteschätzung und Lernmethoden am Bespiel der mehrdimensionalen Gaussverteilung präsentiert. Die Anwendung von statistischen Modellen in der Gebäudeautomation wird beschrieben und begründet.

Das nächste Kapitel, Kapitel 3, beschäftigt sich mit hidden Markov Modellen (HMM): Ein Markov-Prozess kann mit Modellen verschiedenen Abstraktionsgrades beschrieben werden, die wichtigsten davon werden präsentiert. Weiters werden die Algorithmen zum Ermitteln der Wahrscheinlichkeit und zum Neuabschätzen der Parameter eines HMM vorgestellt und ein Ausblick auf kompliziertere Modelle aus der Familie der Segment-Modelle gegeben.
Um mögliche Konfigurationen zukünftiger intelligenter Umgebungen zu diskutieren, meine Sichtweise derselben darzulegen, und um später auf konkrete Anwendungen referenzieren zu können, beschreibe ich in Kapitel 4 zwei konkrete Projekte aus dem Bereich ubiquitäre Computertechnik (ubiquitous computing): SENSE und SEAL.
Am Anfang von Kapitel 5 wird die Architektur eines Kontext-sensitiven Systems inklusiver der notwendigen Begriffe einerseits, und die interne Darstellung von Szenarien mit Basismodellen - um von den Sensordaten zu einer semantischen Repräsentation zu gelangen - in so einem System andererseits, beschrieben. Je nach vorhandenem Wissen über das Verhalten des Systems gibt es verschiedene Möglichkeiten, diese Repräsentation zu adaptieren, einige Varianten werden präsentiert.
In dem Fall, dass kein Vorwissen vorausgesetzt werden kann, muss das Modell vollautomatisch konstruiert werden. Dieser Fall wird in Kapitel 6 gezeigt. Ausgehend von zu treffenden Vermutungen wird die komplette Prozedur zur Konstruktion des Modells vorgestellt und am Beispiel eines binären Bewegungsmelders werden die Beschreibungen und Pseudo-Codes der Algorithmen durchdiskutiert. Schließlich wird die fertige Modellstruktur interpretiert: Wie hat das System die täglichen Abläufe der Menschen in einem Bürogebäude wahrgenommen?
In Kapitel 7 wird der Vergleich eines auf statistischen Modellen basierten Automationssystems mit einem herkömmlichen, regelbasierten System gezeigt.
Ein System, dass HMMs für die Beschreibung des Systemverhaltens verwendet, wird in Kapitel 8 gezeigt: Der Verkehrsfluss in einem Tunnel wird mit Hilfe von HMMs modelliert.
Im letzten Kapitel, Kapitel 9, wird der Inhalt der Arbeit zusammengefasst und abgegrenzt und ein Ausblick auf zukünftige Entwicklungen in diesem Forschungsgebiet gegeben.

Preface

Buildings. We live in them, sleep in them, eat in them, work in them. We spend most of our lives inside their walls. We appreciate their protection and security. However, we demand much more.

As with humans and all elements of nature the building goes through an evolutionary development. Where this development will go, nobody can predict today. But we can state our expectations: The building changes from a simple roof above our heads to protect us from weather influences towards an extended, multi functional body-cover. If we look at what nature has already invented, we see, we are not the first. Some animals adapt their den perfectly to their needs: there exists biological air conditioning, well controlled humidity and air quality, room sizes adapted to period-of-lives and much more. This may not sound spectacular: isn't it already standard in new office buildings? However, we demand much more.

We want our building to not only be an extended body-cover, but to become our habitat. This extension implies lots of additional functionalities like the ability to adjust several vital parameters, protect against outer influences including self-healing, and much more. To ensure these additional functionalities the building needs a certain awareness of itself. This awareness includes knowledge of state parameters and scenarios on the data acquisition side and the possibility of influencing particular parameters of the system and building state on the control side. Scenario recognition in buildings requires new methods for processing large amounts of data in real time aside from standard control engineering technologies.

This work presents an approach to gathering information about scenarios and system states based on statistical methods as opposed to methods based on pre-defined rules. It is therefore necessary to collect as much data about the building and the procedures inside it as possible with sensors. With methods from statistics and machine learning a model of these procedures is then automatically created. This model enables recognition of current scenarios based on the knowledge of previously seen ones and offers therefore also the possibility of predicting activities in the near future.

In combination with projects that "ground semantic symbols"[1] the result of this work builds a basis for many applications in the field of the intelligent building.

The thesis is organized in the following way:

Chapter 1 gives an introduction to current developments in building automation, specifically intelligent environments. Also the ideas of the institute's ARS project and my own approach to gain information out of sensor data because of the limitations of traditional systems are presented. The goal of the thesis is described and one section deals with social implications of computer science.

Readers with experience in statistics, statistical models and the HMM in particular can skip chapter 2 and 3. Chapter 2 introduces methods from probability theory, in particular Bayesian inference, probability density estimation and learning methods, illustrated with the multivariate Gaussian distribution. The application of probabilistic models in building automation is explained and justified.

The next chapter, chapter 3, is dedicated to the hidden Markov model HMM: A Markov process can be described with more or less abstract models, the most important ones are presented. The probability computing and parameter re-estimation algorithms of the HMM are presented and an outlook to more complicated segmental models is given.

[1] semantic symbols: Symbols like "desk", "human", etc.; concepts, that a human can interpret
grounded symbols: such symbols that are derived from real sensor data

To discuss possible configurations of future intelligent environments, to share my understanding thereof, and to have the possibility for later referencing, two particular projects in the field of ubiquitous computing (SENSE and SEAL) are presented in chapter 4.
The first sections of chapter 5 introduce my overall view of a context aware system including the necessary terms and describe possible internal representations of a scenario including basic models to bridge from the sensor data to a semantic representation. Depending on the prior knowledge about the behavior of the system and the user, the representation has to be adapted, several variations are depicted.
In the case that no prior knowledge is available, the model has to be constructed in a fully automated fashion. Chapter 6 starts with discussing necessary presumptions followed by a description of the model constructing procedure. With the example of a binary motion detector sensor the completed generation of the model is discussed, including the descriptions and pseudo-codes of all necessary algorithms. Finally, the structure of the model and its interpretation is given: Which parts of the daily routine of office occupants in this example can be recognized?
In chapter 7 a concrete case study shows the results of a comparison of a statistical-model-based system to a standard building automation system.
A system that makes use of HMMs for overall system description is presented in chapter 8: The traffic situation inside a tunnel is modeled with the use of HMMs.
The last chapter, chapter 9, summarizes the work and gives an outlook to future developments in this particular field of research.

Acknowledgments

Writing this dissertation would not have been possible without the support of several mentors. I would like to thank Prof. Dietmar Dietrich and Prof. Brian M. O'Connell for their guidance, feedback and comments.
In particular, I would like to thank Dr. Brian Sallans for his continuous support in all questions regarding statistical methods.
Further, I would like to thank all members of the ARS team at the ICT and the (former) members of the semantic networks group at ARC for their fruitful discussions.

My deep gratitude goes to my parents, who always supported me on my way.

And, last but not least, all my effort goes towards a satisfied life with my partner in life, Eva.

Contents

Abbreviations

AAL	Ambient Assisted Living
HVAC	Heating, Ventilation and Air Condition
FA	Forward Algorithm
FBA	Forward Backward Algorithm
VA	Viterbi Algorithm
ARS	Artificial Recognition System
pdf	probability density function
EM	Expectation Maximization
HMM	Hidden Markov Model
MC	Markov Chain
MDP	Markov Decision Process
MP	Markov Process
GUI	Graphical User Interface
HSMM	Hidden Semi Markov Model
USB	Universal Serial Bus
PC	Personal Computer
RFID	Radio Frequency Identification
SEAL	Smart Environment for Assisted Living
SENSE	Smart Embedded Network of Sensing Entities
SM	Segmental Model
BWA	Baum-Welsch Algorithm
BAS	Building automation system
IST	Information Society Technologies
FP6,7	Framework Programmes 6 and 7 of the European Commission
ISO	International Organization for Standardization
OSI	Open Systems Interconnection
ICT	Institute of Computer Technology
ICT	Information and Communication Technologies
iid	independent and identically distributed
LON	LonWorks fieldbus
MIB	Management Information Base

Chapter 1

Introduction and State of the Art

The motivation for the thesis lies in the fascinating vision of the intelligent environment. This vision drives the academic research area of ubiquitous computing (Wei91) and its industry-driven clone - pervasive computing (HMNS03), also referred to as ambient intelligence. All of these areas refer to abstract statements and propose implementations foreseen for the distant future.
The common view in these communities is that computers will not only become cheaper, smaller and more powerful, they will disappear and hide or become integrated in normal and everyday objects (Mat04; Hai06). Technology will become invisible and embedded into our surroundings. Smart objects communicate, cooperate and virtually amalgamate without explicit user interaction or commands to form consortia for offering or even fulfilling tasks for a user. They are capable of not just sensing values, but deriving context information about the reasons, intentions, desires and beliefs of the user. This information may be shared over networks - one of which is the world-wide available internet - and used to compare and classify activities, find connections to other people and/or devices, look up semantic databases and much more. The uninterrupted information flow makes the world a global village and allows the user to access his explicitly or implicitly posed queries anywhere anytime.

1.1 Putting this work into context

The vision of ambient intelligence poses many requirements across the whole field of information and communication technology and allied fields (LMB+03). Ambient intelligence requires development in the areas of sensors, actuators, power supplies, communications technology, data encryption and protection, privacy protection, data mining, artificial intelligence, probabilistic pattern recognition, chip design and many others not stated here. Each research group and even researcher has her own view of what ambient intelligence will be or wil approximate. The research is considered to split into three basic research areas - or existing research projects can be grouped into projects investigating three basic methods - which reflect fundamentally different approaches in establishing ubiquitous computing environments (EBM05):

Augmented Reality is understood as an additional virtual layer over the physical environment. This means that the user has some device that displays or presents this additional

information (Lag00) while the user moves through the covered area. Computation is not hidden, but (at least the result) is explicitly presented to the user. The additional information provides the user with location based-services. Examples are among others historical information, advertisement or context information to known user interests.

Intelligent Environments enhance objects with additional sensors, actuators and/or processors. Actual objects therefore develop a behavior, something that was previously impossible and that can be shared across the system (RHC+02). These objects consist of a physical part and an informational part and are sometimes labeled eGadgets - extrovert Gadgets (MK03) or Smart-Its (GSB02). Examples can be smart lamps, chairs, walls or any other object in a house that can provide the system with information about the user's behavior in order to let the system decide on which action to offer.

Distributed Mobile Systems are seen as being closest to the original ubiquitous computing scenarios. They involve multiple mobile devices, share functionality and therefore provide ubiquitous computing power (LS03; WSA+95). These categories also encompass wearable computing systems and other mobile context-aware systems.

Some of the aforementioned approaches and their promise for a more comfortable, safe and secure life are already implemented and presented to the public in demonstration houses. A selection of existing demonstration houses and the related publications are:

- "InHaus" Duisburg[1], Germany (BSS02)
- "House_n" Cambridge[2], US (Int02)
- "Futurelife" Hünenberg[3], Switzerland (AH03)
- "The Adaptive House" Boulder[4], US (Moz98)
- "SmartHOME" München[5], Germany (BRT02)
- "Adhoco" Winterthur[6], Switzerland (LGWA05)
- "TRON Intelligent House" Nishi Azabu[7], Japan (Sak99)
- "Smart Medical Home" New York[8], US (ABD+01)

These examples give a broad range of actual implementations of the vision of intelligent environments. TRON was perhaps the most adventurous of all demonstration houses due to its cost and complexity at that time. However, it was never intended that houses like TRON are built for normal users. The other mentioned examples all have their special focus. The SmartHOME project provides the framework for several projects doing research on sensors, mainly gas sensors. House_n also provides a framework, it is the name of a whole lab doing

[1] www.inhaus-zentrum.de
[2] architecture.mit.edu/house_n
[3] www.futurelife.ch
[4] www.cs.colorado.edu/~mozer/house
[5] smarthome.et.unibw-muenchen.de/de
[6] www.adhoco.com/index.php?site=4&sub=2&lang=en
[7] tronweb.super-nova.co.jp/tronintlhouse.html
[8] www.futurehealth.rochester.edu/smart_home

research on materials, sensors, context awareness and location based services in the house, while the Smart Medical Home provides a concept and prototype testing environment for health care. The inHaus project is targeted to end-users and its main research interest lies on interconnection of several household appliances, while the Futurelife house shows results of such research to a larger public. Adhoco, a company, has built a demonstration to show their building automation systems that help elderly people to live longer independent in their homes. Last, but not least, the Adaptive House is also targeted to end-users, but in a way that the user should not program all the devices by hand, but rather the house should lern the behavior of the user and adapt its actions accordingly.
From nearly twenty years ago researchers show where they belief are the advantages of this technology. Currently, one could say the time is ripe for communications technology, embedded sensors and systems, and building automation control systems to promote intelligent homes, but where are they? In particular there are two main reasons why the distribution of intelligent building automation systems is still behind expectations. First, the vendors of the devices are not prepared (or willing) to offer household appliances' communication abilities in an open [9] fashion. Some of them offer vendor-specific solutions (see for example *serve@Home* by Siemens, `www.servehome.de` and *Miele@home* by Miele `www.miele.de/de/haushalt/produkte/180.htm`) focused on traditional branches, and others just wait until preferences of the customers become clear.
The second reason lies in the structure of the industry itself. There are basically three types of companies. The ones that are specialized in IT-services and have gained knowledge about computing and networks; vendors of household appliances or consumer electronics that have experience in user needs; and last but not least there are the service providers. For offering solutions to the customer it will be necessary to either establish alliances between members of those groups or to join (parts of) companies to originate new providers capable of dealing with the whole spectrum of challenges in intelligent environments.

Recently, governments and also the European Commission realized the potential of intelligent environments, especially when applied as health care and security systems. Therefore, the European Commission started an article-169-initiative labeled Ambient Assisted Living AAL [10]. According to their website AAL addresses a topic of major concern in all European countries:
"Aging societies are a common phenomenon in all European countries. The concept of "Ambient Assisted Living" aims at prolonging the time people can live in a decent way in their own home by increasing their autonomy and self-confidence, the discharge of monotonously everyday activities, to monitor and care for the elderly or ill person, to enhance the security and to save resources."

Additionally, within the European research framework programs, AAL has started to play a role in the IST priority of FP6 and is now - when FP7's structure will be fixed - of even more concern.

[9] What open means or an open standard is, is still subject of discussion, (see for example `www.itu.int/ITU-T/othergroups/ipr-adhoc/openstandards.html`, `ec.europa.eu/idabc/servlets/Doc?id=19528` and `www.oio.dk/files/040622_Definition_of_open_standards.pdf`) whereby the latter includes a definition for an ideal open standard, which basically claims free of charge access for everyone at every time. I see the most promising potential for the widespread distribution of ambient intelligent systems when applying the ideal open standard philosophy for communications on the one hand and vendor competition for devices on the other.

[10] `www.aal169.org`

1.2 History

The Institute of Computer Technology (ICT) developed its own view of ambient intelligence coming from research in fieldbuses and their applications. The problem was that, due to the complexity of modern building automation systems, previously existing design tools such as the ISO/OSI model and the automation pyramid were limited. The introduction of profiles for fieldbus systems was one step to address that issue, but these methods were still insufficient. Therefore, a group at the ICT studied what kind of methods nature provides in dealing with such problems: The ARS project (Artificial Recognition System) was born (Die00). Citing the ARS introduction[11]:
"We need methods to design and to integrate sensor and actuator networks with hundred thousands of nodes. We have to maintain them in a cost-efficient way. We have to develop powerful, highly flexible control systems, to give answers for future demands in the automation area. In this sense we address themes like energy saving, geriatric health care, facility management, efficiency improvement in hospitals, and many others. The decisive point is that complex scenarios must be compiled of many diverse sensors and the corresponding technical intelligence, to be able to react in an adequate way."
Nature's answer - or, more accurately, the result of nature's development process over the ages - is the brain, and in its highest developed form the human brain. It receives huge amounts of information from diverse sensors and classifies, identifies, weighs and reacts on it within split seconds. Research on how the human brain works has been done by psychologists, psychoanalysts and neurologists for more than a hundred years. Especially the last 20 years have brought to light amazing results and models that are rich enough in detail to be of interest for computer scientists for applying them to complex control applications such as future building automation systems (BAS) (Die00).
The ARS project has now reached a point where first pre-defined scenarios are detected (PP05; Rus03; Fue03; Pra06). Additionally, a psychoanalyticaly-inspired model of the human [decision making + memorizing + reacting, or simply: thinking] process (DRT+01) is on its way to being implemented and used to revolutionize agent behavior and interaction. Although a great deal of investigations and progress have been achieved, there are still unanswered questions. On the level of scenario recognition, there is discussion on:

- How to define and then store a "scenario" consisting of different values from diverse sensors describing a behavior over time?
- How to broaden that definition to be able to recognize particular scenarios with (first of all, slightly) changing sensor values, termination, etc.?
- How to enhance existing definition sets of scenarios to generate new definitions for similar or even not so similar event flows?

On a more practical level interesting for each type of new BAS, general questions in ever more complex BA systems arise. These questions concern initialization, automatic detection of type, location or erroneous behavior of units, etc.

[11] `http://ars.ict.tuwien.ac.at`

1.3 Problem Description

Modern BAS allow ever-better observations of activities in buildings with rapidly growing numbers of possible applications. Unfortunately, control systems with large numbers of parameters are hard to describe and - important for symbol processing applications - hard to understand with traditional control engineering technologies. In addition, they are not able to give higher semantic information like recognized scenarios. These considerations redound to new approaches in building automation data processing.
One possible attempt is based on statistical methods.
The goal of this work is to investigate in more complex statistical methods like time series models and hidden Markov models for their use in building automation systems. The scrutinized models shall in a hierarchical structure detect erroneous sensors on their lowest level of generalization and detect scenarios and reflect system's status on their top levels.
The focus in investigations is to be put on unsupervised learning. A combination of learning methods with projects that use rules and pre-defined scenarios is not foreseen here.

1.4 Motivation

Today's building sensor and control systems are primarily based upon the processing of sensor information using predefined rules. The user or operator defines, for example, the range of valid temperatures for a room by a rule - when the temperature value in that room is out of range (e.g. caused by a defect), the system reacts (for example, with an error message). More complicated diagnostics require an experienced operator who can observe and interpret real-time sensor values.

This work will focus on "security and care systems" where the system includes observations of people and interpretation of scenarios. Such a system could for example be used to understand the daily routine in an office or care institution. In this case, either rules would have to be defined that are few and general in character, or a large number of possibilities would have to be specified. However, as systems are targeted more and more towards the end-user in home environments, both possibilities (rule-based systems and expert users) become problematic. The security and care system would require comprehensive prior knowledge of possible operating conditions, ranges of values and possible dangerous conditions. This knowledge will either not be readily available or will be difficult for an unsophisticated user to input. It is also not affordable to have each system customized by an experienced operator during a longer initialization phase in the particular environment. This situation can be seen as forecasting in an unrestricted environment in contrast to systems like energy distribution, where the global consumption of energy can be forecasted very accurately (see (Pal01) for example) with statistical methods and negotiating agents.

Future building automation or habitation assistance systems for care and security applications will act based on their awareness of system status (see (Rus03), (Fue03)) and user behavior and preferences. Therefore, efficient and reliable scenario detection, pattern recognition and tracking systems and algorithms have to be designed. Automation system awareness is a complex and interdisciplinary field. One step towards inferring the intelligence of systems is to define and recognize simple recurring scenarios with slightly different sensor values. In this context we divide a scenario into semantic symbols or concepts. A semantic concept

represents a system or user event that caused the automation system to change its belief in what is happening. For example, there are different symbols for a normal daily routine in an office environment and an afternoon that is interrupted by a meeting.

One goal of this work is to accomplish the task of reliably condensing simple semantic concepts out of sensor data with probabilistic methods without the need of pre-programmed conditions, user-entered parameters, or experienced operators. The system observes sensor data over time, constructs a model of "normal sequences", and compares and classifies newly arriving sensor values. The result is a system that can produce concepts with semantic meaning of sensor readings, with minimal manual configuration of the system. Further, if sensor readings vary or drift over time, the system can automatically adapt itself to the new "normal" conditions, adjusting its parameters accordingly.

1.5 Possible Implications

Computer science and artificial intelligence have been research areas for many decades. The implications of research results in natural science as a whole and computer science and its spheres in particular has influenced our lives on so many levels that even our understanding of ourselves and the understanding of our place in nature has changed radically. In (Wei76) the author tries to face the past, to understand how this change in our view of the world has happened. He insists that natural science is *only one* approach to understanding the universe. In former times, art and literature were considered essential tools of making sense of mankind's place in the world. Now, science is the only legitimate method of understanding, and art has been reduced to the role of entertainment. We are obsessed with scientific procedure and thoughts, he insists.
The reason for him to write the book were the reactions to his own work on ELIZA, a natural-language processing computer program (Wei66). Weizenbaum created ELIZA in order to demonstrate natural-language processing in computer systems and the limitations thereof.
ELIZA mimics a Rogerian psychotherapist. She passed a restricted Turing test for machine intelligence. The idea of the program - taken from basics of contemporary American psychiatrists - is to confront the vis-á-vis with slight modulations of his words in order to let her speak freely. A dialogue of a young woman with ELIZA looked like this:

young woman: Men are all alike.

eliza: In what way?

young woman: They're always bugging us about something specific or other.

eliza: Can you think of a specific example?

young woman: Well, my boyfriend made me come here.

eliza: Your boyfriend made you come here?

young woman: He says I'm depressed much of the time.

eliza: I am sorry to hear you are depressed.

young woman: It's true. I'm unhappy.

eliza: Do you think coming here will help you not to be unhappy?

young woman: I need some help, that much seems certain.

eliza: What would it mean to you if you got some help?

young woman: Perhaps I could learn to get along with my mother.

eliza: Tell me more about your family.

young woman: My mother takes care of me.

eliza: Who else in your family takes care of you?

young woman: My father.

eliza: Your father.

young woman: You are like my father in some ways.

eliza: What resemblance do you see?

young woman: You are not very aggressive but I think you don't want me to notice that.

eliza: What makes you think I am not very aggressive?

young woman: You don't argue with me.

eliza: Why do you think I don't argue with you?

young woman: You are afraid of me.

eliza: Does it please you to believe I am afraid of you?

young woman: My father is afraid of everybody.

eliza: What else comes to mind when you think of your father?

young woman: Bullies.

eliza: Does that have anything to do with the fact that your boyfriend made you come here?

Though ELIZA was seen as capable of carrying on a "human-like" conversation, Weizenbaum never intended ELIZA to be a substitute for human interaction. He was appalled when psychiatrists suggested that the program might be an acceptable substitute for human therapy (CWG66). Even Weizenbaum's own secretary, who was aware that ELIZA was a machine, conversed with the computer in a very personal matter. Horrified, Weizenbaum began work on the philosophical problem presented by the mechanization of human characteristics and talents.
In fact, Weizenbaum is not the only one to raise questions about social and ethical implications of technology, but the questions and expectations of ELIZA's users might give us hints on the reactions on future ubiquitous computing systems.

During the ARS - and related - projects, those implications also play a role. SEAL for example (see section 4.2) will address what may appear as a rather narrow social problem: extending the period which elderly or needy people can stay in their homes. Such a perspective, however, dramatically underestimates the large number of social problems associated with elderly care. Europe's current tendency towards early hospitalization creates significant challenges not only at the financial level, but has also impacts on the service levels for younger people, etc. Early hospitalization is often associated with dramatically deteriorating standards of living for the elderly, including loneliness and depression.
In Austria - as in many EU countries - 80% of the health care services are still performed by family members. Another social impact of SEAL will thus be to improve the co-ordination of the health care services of these "non professional" helpers (family members, friends, etc.) by involving them in the care necessities following a timetable. This not only means increased freedom for these family members, but increased flexibility for work organization and an increase in independence.

Chapter 2

Probability Density Estimation

For finding similarities within sensor data, statistics provides us with a number of methods. For the scope of this thesis, I have choosen to use methods from pattern recognition, probability density estimation and machine learning. Among other things, these fields provide methods for describing data from a data source and adapting a (probabilistic) model's structure suitable for modeling different types and combinations thereof. This chapter gives an overview of the necessary terms and algorithms for the above mentioned topics.

To give an introduction in statistical methods in building automation, I will discuss the example of a temperature sensor located in an office. The sensor is attached to a sensor node (mote) that has a programmable hysteresis, in our case we use $\pm 0.1°C$. If the sensor recognizes a temperature difference greater than that, it launches a message that is transported via the network to a data sink mote. If the temperature stays stable, the mote sends "keep alive" messages with the current temperature every 15 minutes. This is the only possibility to find out if the sensor is still in operation. The sink mote is connected to a PC via USB. On the PC the message format is converted and the data stored in a data base.
The reason for this comprehensive description of the data flow lies in the fact that the way the data is generated, transported and stored influences the choice of the statistical model. The above-mentioned sensor uses a combination of synchronous and asynchronous messages. An example of the sensor data for one day in the mentioned office is given in figure 2.1.

Probability density estimation is commonly seen as the problem of modeling a probability density function (pdf) $p(\mathbf{x})$, given a finite (N) number of data points[1] $\mathbf{x}^n$, $n = 1, ..., N$. $\mathbf{x}$ being a d dimensional data point (e.g. brightness, humidity and temperature in case of a combined senor). The following subsections concerning pdfs follow the notation of (Bis95).
I will explain the notation here in symbiosis with the introduction of Bayes' theorem (which is needed in section 2.1.2). Bayes' theorem

$$P(A = a|B = b) = \frac{P(B = b|A = a)P(A = a)}{P(B = b)} \tag{2.1}$$

relates the conditional and marginal probabilities of two stochastic events. An event in this case happened when a random variable (e.g. A) takes a particular value (e.g. a), written

[1] In the field of statistics a data point is a single typed measurement. The term typed refers to the data type of the data point regardless of the type of the data source. In the above-stated example the type would be a three-dimensional vector of real values.

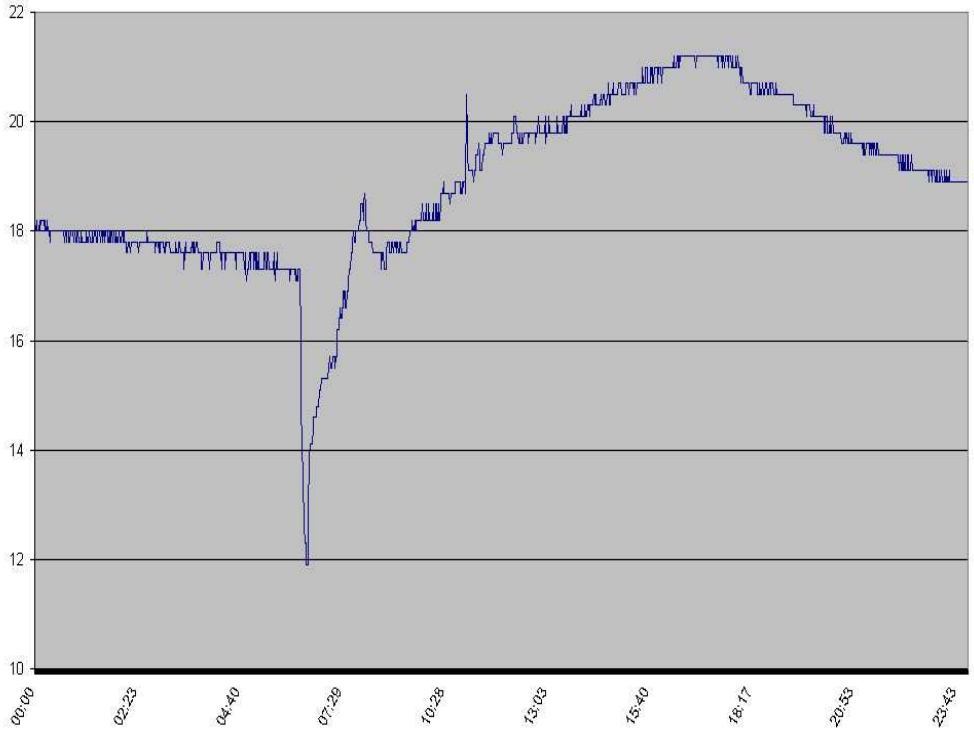


Figure 2.1: Data from a temperature sensor collected over one day. In periods where the temperature is changing quickly the sensor delivers more values than otherwise because of its hysteresis. In periods of constant temperature the sensor gives "keep alive's" every 15 minutes.

$A = a$ or $B = b$. $P(B = b|A = a)$ is the conditional probability of the random variable B being b given that the random variable A has taken a fixed value of a. The common used names for above terms are

- $P(A = a)$ is called marginal probability of A being a, or prior probability because it does not see about the influence of B.
- $P(A = a|B = b)$ is the conditional probability of A being a given that B definitely took b. In the Bayesian vocabulary it is also called posterior probability because it takes the impact of the random variable $B's$ value b on A taking its value a into account.
- $P(B = b|A = a)$ is the conditional probability of $B = b$ given $A = a$.
- $P(B = b)$ is the prior or marginal probability of $B = b$ and acts here as a normalizing constant.

If the random variable can take several events $\{A = a_i\}$, the posterior for a single event can be obtained by

$$P(A = a_i|B = b) = \frac{P(B = b|A = a_i)P(A = a_i)}{\sum_j P(B = b|A = a_j)P(A = a_j)}. \tag{2.2}$$

Uppercase letters (e.g. $P(A = a)$) are used for probabilities of discrete events, whereby lowercase letters (e.g. $p(x)$) are used for probability density functions. Multidimensional values are stated as vector (eg. $p(\mathbf{x})$).

In the theory of probability density estimation, there are three approaches to density estimation: parametric, non-parametric and semi-parametric. The former supposes a particular density function and estimates its parameters for the observed data. Unfortunately, there is no guarantee that the assumption regarding the form of the chosen function models the actual data well. By contrast, non-parametric density estimation makes no assumption at all and therefore lets *the data speak on its own*. The drawback is that the automatic determination of the form leads to large numbers of parameters in the result, typically growing with the size of the data set. Mixture models on the other hand are one particular form of semi-parametric models. Compared to parametric and non-parametric models, semi-parametric models are not restricted to specific functional forms and the size of the model only grows with the complexity of the problem to be solved, not with the size of the data set. The only disadvantage is that the training process of the model is computationally more intensive.

2.1 Parametric Methods

The parametric approach assumes that the probability density $p(\mathbf{x})$ can be expressed in terms of a specific functional form which contains a number of adjustable parameters. These parameters can then be optimized to find the best fit of the proposed pdf to the actual data. The most common parametric model is the Gaussian or normal distribution. The method of parametric probability density estimation can be described with any function. Fortunately, the normal distribution has several convenient properties. Therefore, I will discuss it here in detail.
The well known density function of the normal distribution for a single variable is given by

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}. \tag{2.3}$$

The parameters μ and σ^2 are called *mean* and *variance*, respectively. The factor in front of the exponent ensures the summation to 1 when integrating over $\mathbb{R}$. The mean and variance of the normal distribution are equivalent to its expected value and its 2^{nd} moment

$$\mu = E[x] = \int_{-\infty}^{\infty} xp(x)dx$$

$$\sigma^2 = E\left[(x-\mu)^2\right] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx$$

where $E[\cdot]$ denotes the expectation. In case of multidimensional data the d-dimensional density function is given by

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are a d-dimensional vector and a $d \times d$ *covariance matrix*, respectively. $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$ and the factor in front of the exponent again ensures summation to unity. Mean and variance also satisfy the expectations

$$\boldsymbol{\mu} = E[\mathbf{x}]$$

$$\boldsymbol{\Sigma} = E\left[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\right]. \tag{2.4}$$

This density function allows modeling each dimension of the data space with an arbitrarily

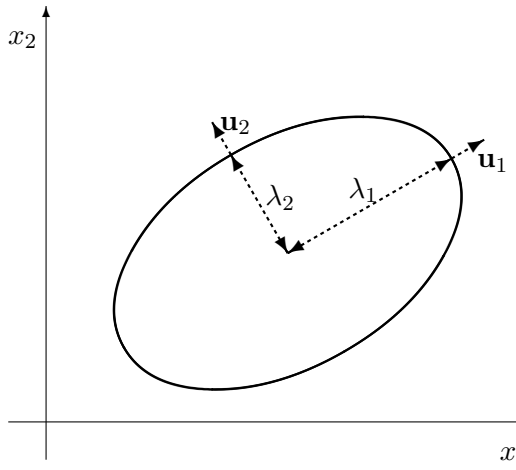


Figure 2.2: General curve with constant probability density of a $2D$ Gaussian pdf. The ellipse is aligned according to the eigenvectors $\mathbf{u}_i$ of the covariance matrix $\boldsymbol{\Sigma}$. The length of the axis is proportional to the corresponding eigenvalues λ_i.

aligned Gaussian pdf. An example of such a general Gaussian pdf for two-dimensional data can be seen in figure 2.2. Following I will introduce multidimensional Gaussian pdfs with reduced versions of the covariance matrix to have fewer parameters, but faster parameter estimation.
From equation 2.4 we see that $\boldsymbol{\Sigma}$ is a symmetric matrix and therefore has $d(d+1)/2$ independent parameters. $\boldsymbol{\mu}$ adds another d independent parameters, so that the full model is characterized by $d(d+3)/2$ parameters. The surfaces of constant probability density for the multivariate Gaussian pdf with full covariance matrix are hyper ellipsoids. Their principal axes are given by the eigenvectors $\mathbf{u}_i$ of $\boldsymbol{\Sigma}$ which satisfy

$$\boldsymbol{\Sigma}\mathbf{u}_i = \lambda_i \mathbf{u}_i.$$

The values of λ_i give the variances along the respective principal axes.
As mentioned above, it is sometimes advantageous for computational reasons to have models with fewer parameters. One possibility in the multivariate Gaussian case is to assume the random variables to be statistically independent and therefore allow only a diagonal covariance matrix

$$(\boldsymbol{\Sigma})_{ij} = \delta_{ij}\sigma_j^2$$

which reduces the total number of independent parameters to $2d$. In this case the surfaces of constant densities are hyper ellipsoids aligned with the coordinate axes (see figure 2.3). The eigenvalues are equal to the covariances in the diagonal of the covariance matrix. If we expect the computational time for each parameter to be roughly equal, then the time for fitting the parameters is reduced from $O(d^2)$ to[2] $O(d)$. The pdf can be simplified to be a product of

[2] In practice, the situation is even better than this, because for general Σ, we have to take the inverse, which takes at least something between $O(d^2)$ and $O(d^3)$.

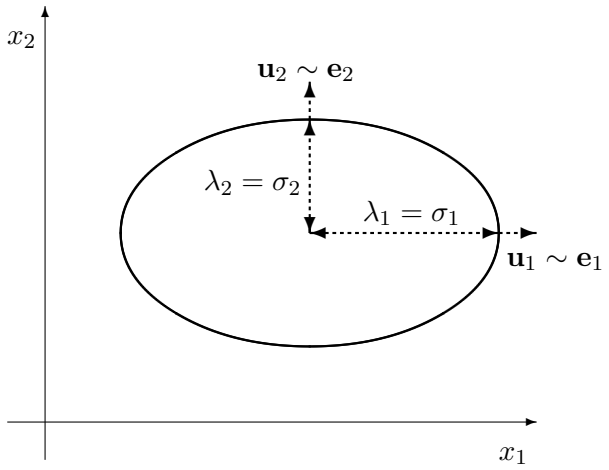


Figure 2.3: Curve with constant probability density of a $2D$ Gaussian with reduced covariance matrix. The ellipse is aligned according to the coordinate axes. The length of the ellipses' axis is proportional to the corresponding covariances σ_1 and σ_2. The properties from the general case still hold: the eigenvectors now lie in the direction of the unit vectors and the eigenvalues equal the covariances.

simple one-dimensional Gaussians

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

where each x_i has the form of equation 2.3.
Further reduction of parameters can be achieved by choosing $\sigma_j = \sigma$ for all j which leaves $d+1$ independent parameters to fully specify the model. The surfaces of constant densities are hyper spheres ($d \geq 3$, in the 2-dimensional case they are circles, an example is illustrated in figure 2.4). This type of model is especially interesting for mixture models (see section 2.2), where the model's likelihood consists of a combination of such simple models. This is used to best cover the data.

After having decided on a particular density function for modeling the data, the parameters of the function have to be adapted to the actual data, which is called statistical inference. Statistical inference in general is inference about a population from a random sample drawn from it or, more generally, about a random process from its observed behavior during a finite period of time. The two most important approaches are *maximum likelihood* and *Bayesian inference*, I will briefly discuss them in the next sections.

2.1.1 Maximum Likelihood

The maximum likelihood approach - as its name implies - tries to maximize the likelihood function of the parameters given the actual data. Formally, we consider our density function $p(\mathbf{x})$ being dependent on a set of parameters $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_M)^T$ which leads to the notation $p(\mathbf{x}|\boldsymbol{\theta})$. We also consider our data to be a set of N (statistically independent) vectors $\mathcal{X} \equiv \{\mathbf{x}^1, \cdots, \mathbf{x}^N\}$ whereby $\equiv$ stands for a definition. The joint probability density of the whole

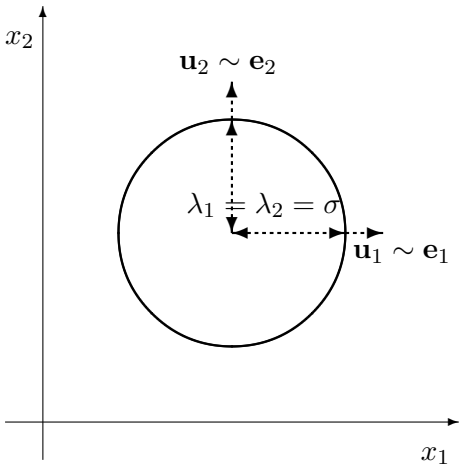


Figure 2.4: Circle with constant probability density of a $2D$ Gaussian with maximum reduced covariance matrix. The covariance matrix consists only of diagonal elements which are all equal.

data set is then given by

$$p(\mathcal{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{x}^n|\boldsymbol{\theta}) \equiv \mathcal{L}(\boldsymbol{\theta}). \tag{2.5}$$

$\mathcal{L}(\boldsymbol{\theta})$ is referred to as the likelihood of $\boldsymbol{\theta}$ given a fixed $\mathcal{X}$ and is subject of later maximization. The idea behind this approach is to have the likelihood as a fit-function to quantify how well the parameters of the chosen function model the actual data. The parameters are adjusted to maximize the likelihood and therefore give the best approximation of the data regarding the chosen density function.
Normally in practice, for numerical stability, log-likelihoods are used instead of likelihoods. Because of their strictly monotonic properties (maximizing likelihoods or log-likelihoods leads to the same result), this substitution can be used. The negative logarithm of the likelihood is given by

$$-\ln \mathcal{L}(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \ln p(\mathbf{x}^n|\boldsymbol{\theta}).$$

This term is normally used as an origin for the utilization of several numerical optimizers (see MB88a; EH81). In the case of the multivariate Gaussian in the form 2.1 it is possible to solve the problem analytically. The results for mean and covariance matrix are

$$\widehat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}^n$$

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{N-1}\sum_{n=1}^{N} (\mathbf{x}^n - \widehat{\boldsymbol{\mu}})(\mathbf{x}^n - \widehat{\boldsymbol{\mu}})^T.$$

2.1.2 Bayesian Inference

Bayesian inference interprets probabilities as degrees of belief[3]. The often use of *Bayes' theorem* (see section 2) gives the discipline its name. To each parameter a probability density function is assigned. At the beginning, where the model has not seen any data, a prior distribution has to be chosen. If prior knowledge about the parameter exists, it can be easily incorporated into a prior distribution. Ideally, the prior should encode the beliefs before seeing any data. Unfortunately, normally little about the distribution of the data is known, or although there exists knowledge, it is computationally more convenient to use a very broad prior. After having seen some data, Bayes' theorem can be used to obtain the posterior

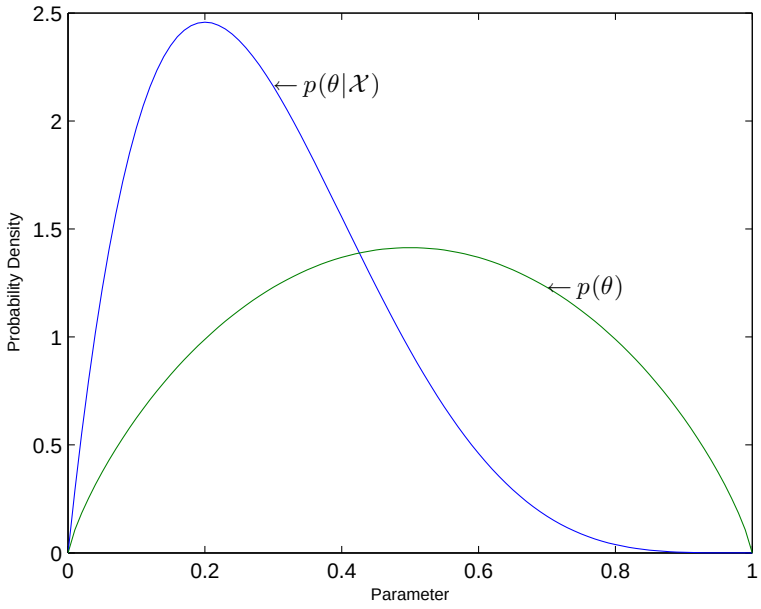


Figure 2.5: Example of prior and posterior distribution for a parameter θ. Our initial belief about the parameter without having seen any data is very broad. After the data set $\mathcal{X}$ has been seen, the posterior distribution is calculated incorporating $\mathcal{X}$ with use of Bayes' theorem. Corresponding to how good the data fits into our chosen parametric model, the posterior can be very narrow around a particular value.

probability. In parameter estimation the posterior gives a better estimate of the parameters than the prior, see section 2.5. The discrete version of Bayes' theorem was used to introduce the notation. The version for continuous functions is given below:
In the continuous case, and specifically when we want to estimate parameters of a model, we assign our parameters a prior probability density distribution. Therefore, to compute the likelihood of a single data vector we cannot simply approximate our pdf as with a partic-

[3]The second school of thought in statistical inference is the frequentist interpretation of probability. Frequentists see probabilities only in context of well-defined random experiments. The experiments' results are subsumed in events that can either occur or not. The frequentist probability is the ratio of the number of occurrences of the event to the number of repetitions of the experiment.
The drawback of that definition is the impossibility of assigning probabilities to anything else than an event in the above description. Therefore it is impossible to assign a probability to the belief of an unknown measure like e.g. the size of something. Frequentist parameter estimation is used in the maximum likelihood method, discussed in section 2.2.1.

ular value for a parameter in section 2.1.1, but we have to integrate over the range of the parameters. The likelihood for a new data vector $\mathbf{x}$ given the data set becomes

$$p(\mathbf{x}|\mathcal{X}) = \int p(\mathbf{x}, \boldsymbol{\theta}|\mathcal{X})d\boldsymbol{\theta} = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{X})d\boldsymbol{\theta}$$

the latter because we assume that $\mathbf{x}$ is independent of $\mathcal{X}$ given θ and therefore the dependency is removed in the conditional probability. The second factor in the last term is the posterior distribution over the parameters, which is computed as follows: With the likelihood of the data set (see equation 2.5) and Bayes' theorem (see equation 2.1) we can write

$$p(\boldsymbol{\theta}|\mathcal{X}) = \frac{p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{X})} = \frac{p(\boldsymbol{\theta})}{p(\mathcal{X})} \prod_{n=1}^{N} p(\mathbf{x}^n|\boldsymbol{\theta}) \tag{2.6}$$

where the marginal probability of the whole data set is obtained by integrating over the parameter space

$$p(\mathcal{X}) = \int p(\boldsymbol{\theta}') \prod_{n=1}^{N} p(\mathbf{x}^n|\boldsymbol{\theta}')d\boldsymbol{\theta}'.$$

These integrals are again often the subject of numerical investigations. An analytical evaluation is (normally) only possible if prior and posterior probabilities have the same functional form, e.g. both are Gaussians. Priors that ensure that the posterior has the same functional form are called *conjugate priors*.

2.2 Mixture Models

The model consists of a linear combination of basis functions, where the number M of basis functions is treated as a model parameter. The border between semi-parametric and non-parametric methods[4] is very narrow since some models that are considered as non-parametric like histograms can have parameters similar to M like the number of beans, also called M. Kernel functions have the kernel width h as their parameter and - as the name implies - K-nearest-neighbors depends on the parameter k.
We define our model as a linear combination of component densities $p(\mathbf{x}|j)$ in the form

$$p(\mathbf{x}) = \sum_{j=1}^{M} p(\mathbf{x}|j)P(j) \tag{2.7}$$

referred to as mixture distribution (see TSM85; MB88b). The coefficients $P(j)$ are called mixing parameters or prior probability. The priors and the component densities have to fulfill the stochastic constraints

$$\sum_{j=1}^{M} P(j) = 1$$

$$0 \leq P(j) \leq 1,$$

[4] I will not discuss non-parametric methods here in detail, because the goal of this work is to find a hierarchical structure of models with parameters that give an idea about the behavior of e.g. occupants in a building. Nevertheless, non-parametric methods are also of great importance in other areas where probability densities are estimated. (Sil86) is a very good introduction to this topic.

ensuring that the priors sum up to unity, so that the data falls into one of the components. The component densities on the other hand are normalized, so that

$$\int p(\mathbf{x}|j)d\mathbf{x} = 1$$

In opposition to true classification problems (see Sil86) we do not have the class label information - telling us to which component the data belongs - for each data point, so the data is *incomplete*. But, the probability distribution for a class $\mathcal{C}_k$ in a k-classes classification problem, $p(\mathbf{x}|\mathcal{C}_k)$ could be modeled with a mixture model.
Applying Bayes' theorem, we can introduce the posterior probabilities

$$P(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)P(j)}{p(\mathbf{x})}, \tag{2.8}$$

$p(\mathbf{x})$ being the sum of all prior-weighted component densities. The posterior for a particular component represents the probability that this component was responsible for generating the d dimensional data point $\mathbf{x}$. The posteriors fulfill

$$\sum_{j=1}^{M} P(j|\mathbf{x}) = 1$$

Priors and posteriors here are discrete values, therefore have upper case letters, while the densities of the various mixture components are continuous functions in general, therefore having lower case letters. For the discussion of the inference methods given in the following sections we assume the mixture components to have Gaussian distribution functions as presented above. An example of a 1D Gaussian mixture model with 3 components can be seen in figure 2.6. In these chapters we will discuss learning methods for mixture models with

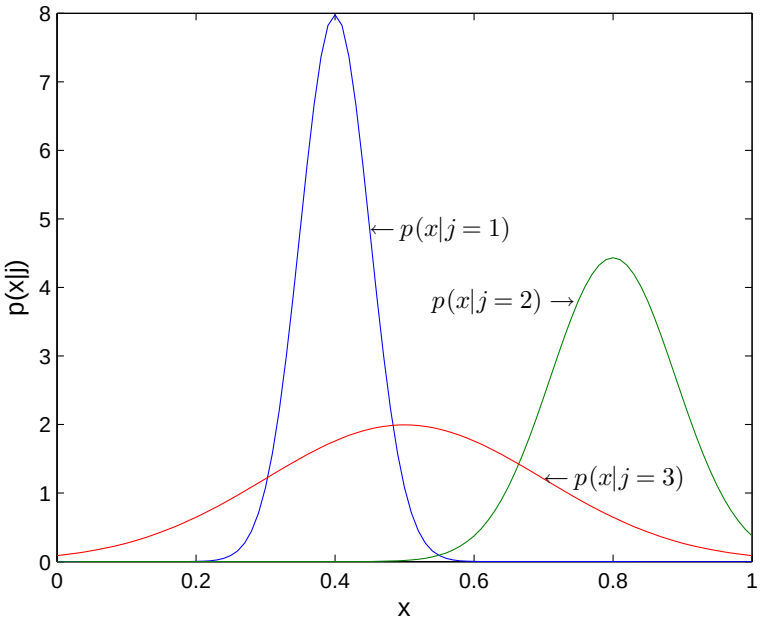


Figure 2.6: 1D Gaussian mixture model. Three component densities are drawn, the model probability distribution is the sum of the prior-weighted components.

statistically independent components, which means with reduced covariance matrices. The component density function is therefore given by

$$p(\mathbf{x}|j) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp\left\{-\frac{\| \mathbf{x} - \boldsymbol{\mu}_j \|^2}{2\sigma_j^2}\right\}. \tag{2.9}$$

$\mathbf{x}$ and $\boldsymbol{\mu}_j$ being d-dimensional values as opposed to the one-dimensional case in figure 2.6.

2.2.1 Maximum Likelihood

The negative log-likelihood for a Gaussian mixture model that is learned from N d-dimensional data points (the data set $\mathcal{X} = \{\mathbf{x}^n\}$) and the following adjustable parameters

M**:** the number of mixing components

$P(j)$**:** the discrete prior probability for each mixture component

$\boldsymbol{\mu}_j$**:** the d-dimensional vector of means for each component and

σ_j**:** the standard deviation for each component. A multivariate Gaussian with reduced covariance matrix is used here, so the covariance matrix is fully characterized by one single value.

is given by

$$E = -\ln \mathcal{L} = -\sum_{n=1}^{N} \ln p(\mathbf{x}^n) = -\sum_{n=1}^{N} \ln \left\{\sum_{j=1}^{M} p(\mathbf{x}^n|j)P(j)\right\} \tag{2.10}$$

whereby each $p(\mathbf{x}^n|j)$ is of the form of equation 2.9. The minimization of this function was subject to a number of publications starting with (see Day69), since this is no trivial task in a number of respects. In most other cases of pdf's, there is no analytical way to minimize the log-likelihood, but with Gaussians it can be done directly. Without going too much into detail, the minimization is done by taking partial derivatives and setting them to zero. As results we gain

$$\widehat{\boldsymbol{\mu}}_j = \frac{\sum_n P(j|\mathbf{x}^n)\mathbf{x}^n}{\sum_n P(j|\mathbf{x}^n)}$$

which can be interpreted as the mean of the j^{th} component is the mean of the data vectors weighted by the posterior probability that the particular component was responsible for generating that data point. The same interpretation can be applied to

$$\widehat{\sigma}_j^2 = \frac{1}{d}\frac{\sum_n P(j|\mathbf{x}^n) \| \mathbf{x}^n - \widehat{\boldsymbol{\mu}}_j \|^2}{\sum_n P(j|\mathbf{x}^n)}.$$

Finally, the prior probability estimate is given by the summarized and normalized posteriors:

$$\widehat{P}(j) = \frac{1}{N}\sum_{n=1}^{N} P(j|\mathbf{x}^n).$$

These formulas give the solution to the maximum likelihood estimate, but they do not provide a way of computing them in an efficient algorithmic way. This is given by the EM algorithm discussed in the next Session.

2.2.2 Expectation-Maximization

The Expectation-Maximization (EM) algorithm is an algorithm for finding maximum likelihood estimates typically of parameters in statistical models where the sample is incomplete (see DLR77). The incompleteness of the data sample comes from the assumptions made for the hidden parameters of the HMM. The basic idea of the EM algorithm is to provide an iterative procedure for finding minima of the error function. This is done in two steps, the expectation step and the maximization step. The expectation step takes an initial belief about the parameters and calculates the likelihood of the given data sample. In the maximization step the likelihood is assumed to be given and the parameters are re-estimated given the data.
When using EM for estimating the parameters of a Gaussian mixture model, the expectation and maximization steps are performed simultaneously. This is done by taking an initial belief about the parameters, calculating the right-hand sides of the above formulas and getting revised estimates. These "new" parameters are then seen as a better estimate and they are taken as the initial values for the next estimation round.
The full derivation of the EM results of a mixture of Gaussians model can be seen e.g. in (Bis95), the iterative estimation formulas are:

$$\begin{aligned} \boldsymbol{\mu}_j^{new} &= \frac{\sum_n P^{old}(j|\mathbf{x}^n)\mathbf{x}^n}{\sum_n P^{old}(j|\mathbf{x}^n)} \\ (\sigma_j^{new})^2 &= \frac{1}{d}\frac{\sum_n P^{old}(j|\mathbf{x}^n) \parallel \mathbf{x}^n - \boldsymbol{\mu}_j^{new} \parallel^2}{\sum_n P^{old}(j|\mathbf{x}^n)} \\ P(j)^{new} &= \frac{1}{N}\sum_{n=1}^{N} P^{old}(j|\mathbf{x}^n) \end{aligned} \tag{2.11}$$

2.3 Statistical Models for Error Detection in Automation Systems

In chapter 7 I compare a diagnostic system to a standard building automation system. The building automation system consists of a number of sensors and actuators connected by the LonWorks fieldbus (LON) (LDS01; KDP02; KNSN05). It offers a visual interface using a Management Information Base (MIB) for retrieving and manipulating system parameters and for the visualization of system malfunctions.

The diagnostic system is based on statistical "generative" models (SGMs). Statistical generative models attempt to reproduce the statistical distribution of observed data. The model can then be used to determine how likely a new data value is or to "generate" new data (i.e. draw samples from the model).[5]

Recent work in generative models had its focus on non-Gaussian models (see for example (HSG98; LGS99)). The diagnostic system uses a number of different SGMs to capture the

[5]For probability density models, which are functions of real-valued variables, the probability of the input value is not directly computed from the model. Rather, the probability of either exceeding the given value, or of generating a value within a small neighborhood of the given value is computed.

distribution of sensor data, including histograms, Gaussian mixture models (Bis95), and hidden Markov models (RJ86).

Sensor and control data poses several challenges (see also section 5.2 and section 6.1). The data can be anything from very low level temperature readings to higher level "fault" detectors or occupancy data from building entry systems. This very different data must be fused into a single system. The volume of data requires fast algorithms (Jaa97; JGJS99), and algorithms that can work with on-line data as it arrives (NH98). The data values are time-dependent, so a model must (explicitly or implicitly) take time into account (Kal60; RJ86; Sal00).

Previous work in applying statistical methods for fault detection includes the use of methods from statistical quality control (SQC) (see (FS94) and (SH03), for example), and statistical process monitoring and control (GLS+00). Statistical quality control methods compare sensor values to prepared statistical "charts". If the sensor values vary from their expected values over a prolonged period of time, the charts can detect this variation. These methods are appropriate when it is sufficient to reliably detect a small variation after collecting a large number of samples, but are inappropriate for detecting abnormalities in real time. Methods from statistical process monitoring have been applied to a number of systems including chemical processes monitoring (WG96), monitoring of engines (PL03), and monitoring of communications networks (HJ97). Typical statistical methods used principal components analysis and partial least squares models fit to historical batch (WG96) or online data (GLS+00). Deviations from normality are then detected using standard statistical tests.

There are several other approaches to fault detection. In classical model-based detection, detailed domain knowledge is used to build a model of the system. Deviations between model predictions and system behavior are flagged as faults (see (Ise84) for a survey). In pattern-matching detection, faults are induced in a system, and the resulting sensor values are recorded. A classifier, such as a neural network, is trained using this data set of normal and abnormal behavior to detect failures (see (HLS99) for an example). These methods require either a working system for experimentation, or in-depth knowledge of the system in question, both of which are lacking for large building automation systems.

Despite their success in other domains, SGMs have not been applied to error detection for building automation sensor and control data. There are two reasons for this. First, it has only recently become possible (and economical) to collect a wide range of cross-vendor sensor data at a central location. Recent developed description languages for automation systems also help to connect cross-vendor sensor networks (see for example (MHDP05)). Second, most algorithms to optimize the parameters of SGMs are quite compute-intensive. Many algorithms have only been of theoretical interest, or are restricted to small toy problems. Only recently have powerful approximation algorithms and powerful computers become available that can handle large quantities of data in real-time.

2.4 Statistical models for a diagnostic system

The goal of the diagnostic system is to automatically detect sensor errors in a running automation system. It does this by learning about the behavior of the automation system by observing data flowing through the system. The diagnostic system builds a model of the

sensor data in the underlying automation system, based on the data flow. From the optimized model, the diagnostic system can identify abnormal sensor and actuator values. The diagnostic system can either analyze historical data, or directly access live data.

I use a set of statistical generative models to represent knowledge about the automation system. A statistical generative model takes as input a sensor value, status indicator, time of day, etc., and returns a probability between zero and one.

Using SGMs has several advantages. First, because the model encodes the probability of an occurring sensor value, it provides a quantitative measure of "normality", which can be monitored to detect abnormal events. Second, the model can be queried as to what the "normal" state of the system would be, given an arbitrary subset of sensor readings. In other words, the model can "fill in" or predict sensor values, which can help to identify the source of abnormal system behavior. Third, the model can be continuously updated to adapt to sensor drift.

To illustrate the advantage of this approach we can think of a large building with a building automation system to collect temperature readings of all rooms. In a standard rule-based system the operator has to set up a rule base for each sensor. This rule base includes allowed - or considered to be normal - sensor values, e.g. 17 - 28 °C. If this particular range is a good choice will depend on a number of factors:

- Is the room on the window side of the house?
- If yes, what is the direction?
- Does the room have a special purpose (server room, meeting room, toilet, ...)?
- Who is using the room, and when?
- etc.

Here are just a few of the emerging considerations resulting from the list: If the room is a server room with climate control adjusted to e.g. 18°C, a very narrow range around this value is to be allowed only. Or: If the room has a window to the east or south and the temperature sensor is mounted in an unfavorable position, it may be that the sun will heat it up to above 28°C in summer.
SGMs on the other hand possess the property to learn about normal values[6], and are therefore able to adapt their monitoring or alerting behavior according to the real needs.
If somebody would like to initialize all of the normal ranges - which is an important parameter, that's true, but just one - according to the real conditions with a rule-based fashion, the initialization and maintenance costs would climb very high. Further advantages of this approach are the ability to adapt to the habits of particular users (temperature, working hours, light conditions, ...) and to have a second control loop for the function of the HVAC system. The latter is discussed in depth in chapter 7.

[6] in most of the cases there exists something like a mean value and something like a width of the distribution around that mean

2.4.1 Error Detection

Given an SGM, implementation of this functionality is straight-forward. The system assigns a probability to each newly-observed data value. When this probability is high, the system returns the information that the new data value is a "normal" value. When the probability falls below a specific threshold, the system rates the value as "abnormal". The SGM system generates alarm events when it observes abnormal sensor values. This leaves open the question of how to assign the threshold for normality. In practice, the user sets the threshold using a graphical interface. Initially, before the system has learned normal system behavior, many alarms are generated, and the user may decide to set the threshold to a value near zero. As the system acquires a better model of the sensor system, the threshold can be raised. In any case, the threshold parameter tells us how improbable an event should be to raise an alarm. The system can also use a log-probability scale, so that the threshold can easily be set to only register extremely unlikely events.

2.4.2 Statistical Generative Models

The system implements a number of SGMs (see Table 2.1).

Table 2.1: Statistical Generative Models

Model	Variable Type	Parameters
Gaussian	Real	μ,σ^2
Histogram	Discrete, Real	Bin counts
Mixture of Gaussians	Real	μ_i,σ_i^2,π_i
Hidden Markov model	Real	T_{ij},μ_i,σ_i^2
Hidden Markov model	Discrete	T_{ij},Bin counts

The more complex models add additional capabilities, or relax assumptions in comparison to a simple Gaussian model.

Histogram: This is a very general model that is appropriate for discrete sensor values, as well as real-valued sensors with an arbitrary number of modes. One drawback is that a histogram requires a rather large quantity of data before it becomes usable or accurate.

Mixture of Gaussians: This model relaxes the Gaussian assumption that the distribution has only one mode. A mixture of Gaussians is composed of a number of Gaussian models, and each data value is attributed to the Gaussian modes with a weighting given by a "posterior probability". See for example (Bis95).

Hidden Markov Model: This model is the equivalent of the Mixture of Gaussians model or the histogram model, but with the addition that the current sensor value can be dependent on previous values.

The diagnostic system uses SGMs of automation data points. For any given data value x, model M assigns a probability to x: $P_M(x) \rightarrow [0, 1]$.

Note that, for discrete distributions such as a histogram, the value assigned to x by the model $P_M(x)$ is a well-defined probability, since the set of possible assignments to x is finite. For a probability density, such as a Gaussian or mixture of Gaussians, the probability value assigned to x by the model is the probability *density* at that value. In order to convert this density to a probability, the probability of generating a value within a neighborhood $\pm\delta$ around x is computed as $\int_{-\delta}^{\delta} P_M(x+\phi)d\phi$, and approximated as $2\delta P_M(x)$ for small δ. Alternatively, the probability under the model of equaling or exceeding the observed value can be computed: $P_M(x' \geq x) = \int_{\phi}^{\phi+\infty} P_M(x+\phi)d\phi$.

The data x can be a sensor reading such as an air pressure sensor, contact sensor, temperature sensor, and so on. Given a new data value, the system assigns a probability to this value. When the probability is above a given threshold, the system concludes that this data value is "normal".

Given a sequence of sensor readings $\mathbf{x} = \{x_1, ..., x_T\}$ from times 1 to T, the system must create a model of "normal" sensor readings. The system uses an on-line version of the expectation maximization algorithm for maximum-likelihood parameter estimation. Given a model M with parameters θ, the log-likelihood of the model parameters given the data $\mathbf{x}$ is given by:

$$\mathcal{L}(\theta) = \log P_M(\mathbf{x}|\theta) \tag{2.12}$$

where the notation $P(\mathbf{x}|\theta)$ denotes the conditional probability of $\mathbf{x}$ given the current values of the parameters θ.
The maximum-likelihood parameters are defined as the parameter values that maximize the log-likelihood over the observed data:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \{\log P_M(\mathbf{x}|\theta)\}$$

2.4.3 On-line Parameter Updates

In order for the system to continually adapt the model parameters, the parameter update algorithm must incrementally change the parameters based on newly observed sensor values. Such "on-line" updates have the advantage that there is no time during which the system is in an "optimization" phase, and unavailable for diagnostics.
For the tests described in chapter 7, mixture of Gaussians models were used. For the mixture of Gaussians model, the system uses a simple stochastic estimation method, based on an expectation-maximization algorithm. As each new data value x_i is observed, the parameters are adjusted in a two-step process. First, the posterior probability of each element of the mixture given the data value is computed. Second, the parameters are adjusted so as to increase the expected joint log-probability of the data and the Gaussian mixture component. See section 2.2.2 for details.

Chapter 3

Hidden Markov Models

The goal of this work is to provide a model structure for detecting, storing and recognizing recurring patterns in the behavior of persons and/or objects in buildings. From the model's perspective this distinction is irrelevant. This goal is a clear improvement compared to the learning of normal ranges of values - as discussed in the previous chapter - because it adds a level of abstraction to the monitoring task. Nevertheless, we will see that the SGMs are also a crucial factor to fulfill this ambitious goal.
The system perceives its environment through sensors and the goal is to create a possibility to distinguish between different kinds of behavior. (RG99) have shown that several basic methods for the task of unsupervised learning can be unified under a single basic generative model: factor analysis, principal component analysis, mixture of Gaussian clusters, vector quantization, Kalman filter models, and hidden Markov models. This unification means that all of these different models are in fact variants of a basic model and therefore somehow interchangeable. The main difference when choosing a model as the language of data description is whether if the data under observation is discrete or continuous in time and data space. For the aforementioned task we are searching for a model that is used for discrete time and continuous values, as is the case for the HMM. It has a vector of states that is said to model the underlying behavior of the data source. Second, each state has an emission probability distribution which allows the modelling of similar output for different states as well as various different output symbols for each single state. This definition matches well with the definition of scenarios in chapter 5, when we see the images as a multidimensional vector of sensor readings - the emissions - and the scenario as the underlying behavior - the state chain - which can also emit varying emissions at each state.
Therefore, I give an overview of Markov models of different complexity in this chapter, starting with the basic definition of the Markov property and going into quite complex segmental models with the hidden semi-Markov models (HSMM) as their easiest version. HSMMs have emission, transition and duration probability distributions for each of their states.

3.1 Markov Property

Consider a system that observes its environment and shall e.g. protect humans from potentially dangerous situations. This system bases its actions and warnings on information retrieved from a number of sensors and memories of previous inputs. This information tells

the system something about the *state* of the world. A system being in some state s at time t executes some action and receives a reward from the environment. This is called a *decision process*. The task of determining to perform an action based on the reward is called *reinforcement learning*. If the next state is only dependent on the current state and action, the decision process is said to obey the *Markov property*[1]. In other words: Predictions about future behavior of a Markovian system when knowing only the current[2] state of the system cannot be improved by gaining more information about previous states. The decision process then is called a *Markov decision process* (MDP). If the sets of states and actions are finite, we talk about *finite MDP*. If the output samples of the process are finite, the problem is called a finite horizon problem in contrast to infinite horizon problems, where a data source continuously emits symbols. In this work, only finite horizon problems are considered and only models with finite state space are used to fit the data.

3.2 Markov Process

The discrete time Markov Process (MP) produces a discrete time Markov chain (MC) of values. A MC of n^{th} order is defined as follows:
$P(Q_{t+1} = q_{t+1}|Q_t = q_t, Q_{t-1} = q_{t-1}, ..., Q_0 = q_0) = P(Q_{t+1} = q_{t+1}|Q_t = q_t, ..., Q_{t-n+1} = q_{t-n+1})$ whereby uppercase letters denote random variables and lowercase letters actual values thereof. This means that the probability for being in some particular state at some particular time depends only on the n preceding states in contrast to knowing about all previous states. The simplest, but most important case is *first order Markov chains* (see equation equation 3.1 and figure 3.1)

$$P(Q_{t+1} = q_{t+1}|Q_t = q_t, Q_{t-1} = q_{t-1}, ..., Q_0 = q_0) = P(Q_{t+1} = q_{t+1}|Q_t = q_t) \tag{3.1}$$

where the probability of being in some state at some time is only dependent on the previous state. In other words: additional knowledge about the past does not enhance the predictions about the future.

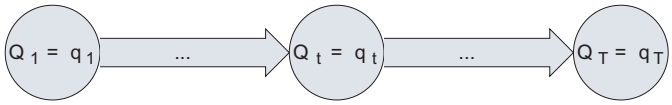


Figure 3.1: The Markov chain. The time index goes from 1 to T in case of completed data samples (with horizon T).

3.3 Markov Model

If the number of possible states in a Markov process (its state space) is assumed to be finite, we can model it with a *Markov model* (see figure 3.2). The Markov model explained here, and all subsequently introduced models in statistics, try to model processes which are assumed to possess the Markov property. Sometimes this assumption does not hold,

[1] of first order in this case
[2] n^{th} order Markovian systems know about n preceding states

but the approximation is still sufficient and convenient (see for example (Eie99)). Every state of the Markov model is said to produce some *output symbol s* which is an element of the output or symbol alphabet[3] Σ. The probabilities of transitions between states - the *transition probabilities* -

$$A = \{P_{ij} = P(Q_{t+1} = j | Q_t = i)\},$$

Q_t denoting the current state, together with the Prior distribution

$$\pi = \{\pi_i = P(Q_0 = i)\},$$

also called initial state distribution vector, formally define a Markov model. With these definitions we can use the compact notation $\lambda = (A, \pi)$ and say the pair (A, π) characterizes the model λ. Markov models provide a framework which the user can adapt by supplying transition pdfs of an arbitrary shape. If one uses the basic Markov model and extends it by an emission probability distribution to allow the association of a distribution over some output symbols with each state, it is called hidden Markov model (HMM, see section 3.4). Additionally adding a duration probability distribution - this allows the modeling of the time, the model stays in a state before leaving - gives a hidden semi-Markov model (see section 3.5). It is also possible to define a model that consists of a Markov model on the top level having HMMs as the states. In chapter 6 I will give a more detailed introduction to this idea.

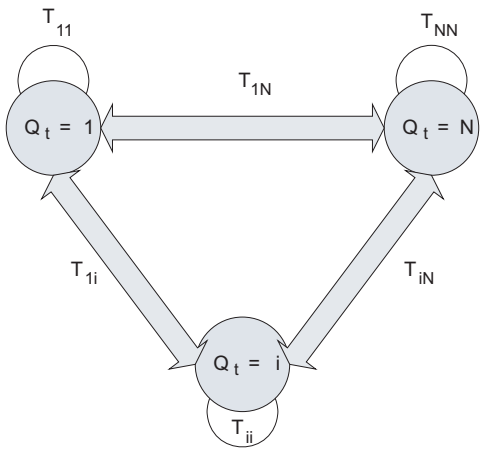


Figure 3.2: The Markov model. It consists of N states with a $N \times N$ transition probability matrix T. Depending on the non-zero transitions, the actual state at time t, Q_t can take every value from 1 to N.

3.4 Hidden Markov Model

Under some circumstances, the process we want to model is not described sufficiently by a Markov model. Consider a situation where you can measure - or observe - some value, but you would like to infer the driving force behind the values from that observations. In this case,

[3] In the case of the Markov model, there is no distinction made between a state and its corresponding output symbol, because each implies the other. In the case of more complex models, emission probability distributions are introduced which allow to associate different output symbols with each state.

hidden Markov models(see figure 3.3) are used. They are defined as a triple $\lambda = (A, B, \pi)$, B being the confusion matrix[4]

$$B = \{b_{ik} = P(O_t = k|Q_t = i)\}$$

which gives the probabilities of emitting symbol k while in state i for all times t. In other words, the hidden Markov model extends the Markov model by emission probability distributions. The complete definition of a hidden Markov model is given below:
A hidden Markov model is a variant of a finite state machine having a set of states $\mathbf{Q}$, a transition probability matrix A, an output alphabet $\mathbf{\Sigma}$, a confusion or emission probability matrix B and initial state probabilities π. The states are not observable and therefore called hidden. Instead, each state produces some output symbol according to the emission probability distribution B. HMMs with time-independent transition and emission probability distributions are called stationary or homogeneous. I will only consider these and just speak of HMMs. HMMs are characterized by:

- The number of states N;
- The number of output symbols in the output alphabet, M;
- The transition probability matrix $A = \{P_{ij}\}$
  $P_{ij} = P(Q_{t+1} = j|Q_t = i) \qquad 1 \leq i, j \leq N;$
- An emission probability distribution in each of the states $B = \{b_{ik}\}$
  $b_{ik} = P(O_t = k|Q_t = i) \qquad 1 \leq i \leq N, 1 \leq k \leq M;$
- Finally, the initial state distribution vector $\pi = \{\pi_i\}$
  $\pi_i = P\{Q_0 = i\} \qquad 1 \leq i \leq N.$

Q_t is the current state at time t. All of these variables are probabilities and therefore have to fulfill the normal stochastic constraints:

- $0 \leq P_{ij} \leq 1 \qquad 1 \leq i, j \leq N;$
- $0 \leq b_{ik} \leq 1 \qquad 1 \leq i \leq N, 1 \leq k \leq M;$
- $0 \leq \pi_i \leq 1 \qquad 1 \leq i \leq N;$
- $\sum_{j=1}^{N} P_{ij} = \sum_{k=1}^{M} b_{ik} = 1 \qquad 1 \leq i \leq N;$
- $\sum_{i=1}^{N} \pi_i = 1$

The notation in this chapter follows the standard notation given by (RJ86). After having selected the HMM to model a specific process, there are three possible tasks to accomplish with the model.

[4]The term confusion matrix is used in mathematics. We can also use the terms emission matrix - in case of a discrete number of output symbols - or emission probability distribution.

1. Inferring the probability of an observation sequence given the fully characterized model (evaluation).
2. Finding the path of hidden states that most probably generated the observed output (decoding).
3. Generating a HMM given sets of observations (learning).

In case of learning, HMM *structure learning* (finding the appropriate number of states and possible connections) and HMM *parameter estimation* (fitting the HMM parameters, such as transition and emission probability distributions) must be distinguished.

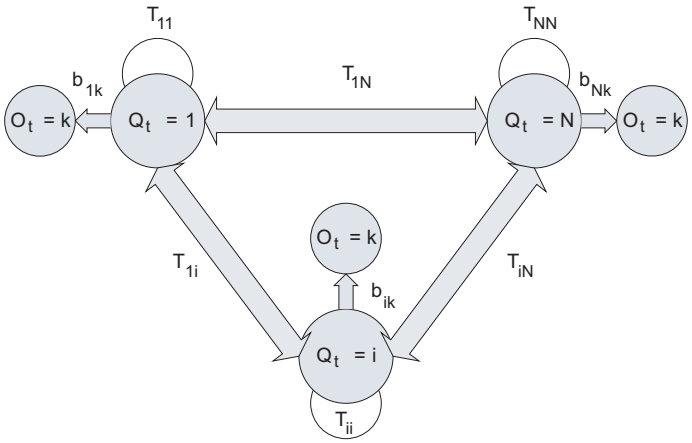


Figure 3.3: The hidden Markov model. It consists of N states with transition probabilities between these states, denoted T_{ij}. Each state i also has an emission probability distribution over output symbols b_i. Here, the random variable for the output is depicted as O_t and the random variables for the states as Q_t.

3.4.1 Forward Algorithm

Consider a problem where we have different models for the same process and a sample observation sequence, and we want to know which model has the best probability of generating that sequence. This task is accomplished by the *Forward Algorithm*.
E.g. in the SEAL application (see chapter 4), where we will have a set of possible semantic symbols like "person walking" or more abstract "normal day in an office", possibly modeling a sequence of values we want to know which one could be the most probable cause for the sequence we see, or in other words: how probably is each of these scenarios the cause for the seen sequence of sensor values.
One method of calculating the probability of the observed sequence is the exhaustive search method where we try to find all possible sequences of hidden states and sum up those probabilities. It can be easily seen that this method is computationally expensive and an unnecessary overhead because we can use the time invariance of the model to recursively compute the sequence step by step.
Consider already having the probability for the first m values of our sequence. We can now compute the probability for the next single value by incorporating all possible next states, accessible from the current one (transition probability > 0), having an emission probability

for observation sequence symbol $m+1$. This has to be done for all possible current states that could be reached in this way.
The starting conditions at time $t = 1$ are given by the initial state probabilities multiplied with the corresponding emission probabilities. We can set $\alpha_1(j) = \pi(j)b_{jO_1}$, α being a joint probability and $\alpha_1(j)$ the forward probability for state j after time $t = 1$ emitting the output symbol observed at time $t = 1$, O_1. With this initial conditions and the description above, we can give the schema for the rest of the observation sequence as:

$$\begin{aligned} \alpha_t(j) &= P(O_1\ O_2\ \cdots\ O_t, Q_t = j|\lambda) \\ &= b_{jO_t} \sum_{i=1}^{N} \alpha_{t-1}(i)p_{ij} \end{aligned} \tag{3.2}$$

Finally, the probability of the HMM generating the given observation sequence is the sum of all probabilities at time $t = T$, T being the length of the sequence $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$.

3.4.2 Viterbi Algorithm

The Viterbi algorithm addresses the decoding problem. Thereby we have a particular HMM and an observation sequence and want to determine the most probable sequence of hidden states that produced that sequence.
In section 6.3 we will see another application where the Viterbi algorithm is used to approximate the parameter updates of a HMM with reduced computational complexity.
As with the forward algorithm, the exhaustive search method is possible but not viable for large HMMs (i.e. large values of T, $T > 30$) considering the time invariance of the problem. The solution is to define the Viterbi path probability δ, which is the probability of reaching a particular intermediate state, having followed the most likely path:

$$\delta_t(i) = \max_{q_1,q_2,\cdots,q_{t-1}} P(Q_1 = q_1,\ Q_2 = q_2,\ \cdots\ Q_t = i, O_1\ O_2\ \cdots\ O_t|\lambda)$$

The difference between the forward probability α and δ is that the latter is the probability of the most probable path to a state on a way through the model while the first sums up all possible path probabilities up to the corresponding state. $\delta_t(i)$ denotes the Viterbi path probability of being in state i at time t. This is the highest probability of all possible paths to that state. The corresponding path is called partial best path.
While moving through the model, the current partial best path can change its past states from step to step. This depends on the progression of all the possible paths $\delta_{t'}(i)$ to $\delta_{t'+1}(j)$. The Viterbi path probabilities for the first step of the algorithm are simply $\delta_1(j) = \pi(j)b_{jO_1}$ - as in the forward algorithm. As mentioned above, the expansion from step $t-1$ to step t is the path with best probability to the next state, formally

$$\delta_t(j) = \max_{i,j}(\delta_{t-1}(i)p_{ij}b_{jO_t}) \tag{3.3}$$

The Viterbi path probabilities give us the probability for the best path through the model, but the aim is to find the best path and not only its probability. The solution is to remember the predecessor of each state that optimally provoked the current state, or in other words to store a back pointer ϕ for each intermediate state.

$$\phi_t(j) = \arg\max_{i}(\delta_{t-1}(i)p_{ij}) \tag{3.4}$$

Here, the argmax operator selects the index i that maximizes the expression. Please note that the $\phi's$ do not require the emission probabilities to be computed.

3.4.3 Forward Backward and Baum-Welsch Algorithm

These algorithms address the third - and most difficult - problem of HMMs: finding a method to adjust the model's parameters to maximize the probability of the observation sequence, given the specific model. Unfortunately, there is no analytical way to accomplish this task. All we can do is to locally optimize $P(O|\lambda)$, O being the observation sequence. To describe the procedure for re-estimation of parameters, we need to define some more variables. We will need $\xi_t(i,j)$, the probability of being in state i at time t and j at time $t+1$, $\gamma_t(i)$, the probability of being in state i at time t, both given the observation sequence O and the model λ. The computing of these variables is done in the forward backward algorithm (FBA), and re-estimating the parameters is the scope of the Baum-Welsch algorithm (BWA).
With the variables from the FBA we can calculate an intuitive estimate for transition probabilities, which is defined as (number of transitions from state i to j)/(number of transitions from state i) for p_{ij}. For their computation we need $\beta_t(i)$, the probability of the partial observation sequence from $t+1$ to T, which is defined in a way similar to the α's.

$$\beta_T(i) = 1$$

and

$$\begin{aligned} \beta_t(i) &= P(O_{t+1}\ O_{t+2}\ \cdots\ O_T|Q_t = i, \lambda) \\ &= b_{ik} \sum_{j=1}^{N} \beta_{t+1}(j) p_{ij} \end{aligned}$$

With these backward and the forward probabilities we can introduce the ξ's as

$$\begin{aligned} \xi_t(i,j) &= P(Q_t = i, Q_{t+1} = j|O, \lambda) \\ &= \frac{P(Q_t = i, Q_{t+1} = j, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) p_{ij} b_{jk} \beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) p_{ij} b_{jk} \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) p_{ij} b_{jk} \beta_{t+1}(j)} \end{aligned}$$

and the γ's as

$$\begin{aligned}\gamma_t(i) &= P(Q_t = i|O, \lambda) \\ &= \frac{P(Q_t = i, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}\end{aligned}$$

These factors are related by summing up over j

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

If we now sum up over time, the result gives us the expected number of transitions from state i ($\gamma_t(i)$), and the expected number of transitions from state i to j ($\xi_t(i, j)$).

$$\begin{aligned}\sum_{i=1}^{N-1} \gamma_t(i) &= \text{expected number of transitions from state } i \\ \sum_{i=1}^{N-1} \xi_t(i, j) &= \text{expected number of transitions from state } i \text{ to state } j\end{aligned}$$

With these values we can give a method for re-estimation of model parameters that either do not change the parameters in case of a (local) optimum or change the parameters in a way that $P(O|\lambda)$ rises, which are the Baum-Welsh re-estimation formulas. The set of re-estimation formulas for π, A and B is

$$\begin{aligned}\overline{\pi}_i &= \text{expected frequency in state } i \text{ at time } t = 1 = \gamma_1(i) \\ \overline{a}_{ij} &= \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}\end{aligned} \tag{3.5}$$

$$\begin{aligned} \overline{b}_{jk} &= \frac{\text{expected number of times in state } j \text{ observing output symbol } k}{\text{expected number of times in state } j} \\ &= \frac{\sum_{t \in \{t' : O_{t'} = k\}} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned} \tag{3.6}$$

3.5 Hidden Semi Markov Model

In many fields, like speech recognition and DNA sequencing, HMMs still play a role in most recognition systems (see (BJM83; Rab89) for example), but the proponents are searching for models of higher order to have more flexibility and to better describe complex situations. Such models tend to require higher computational effort, but overcome the limitations of the HMMs, which are:

- weak duration modeling
- the assumption of conditional independence of observations given the state sequence
- restrictions on feature extraction, imposed by time-window-based observations.

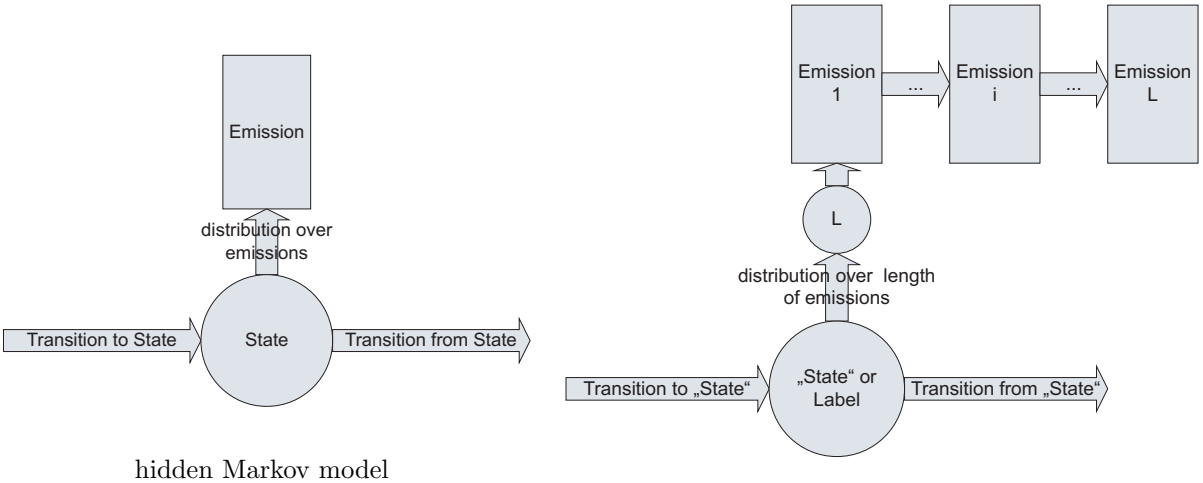
In (ODK96) the authors give an overview of various higher-order models that are intended to solve at least one of these issues.
In speech recognition, the goal is to find a sequence of labels $\mathbf{a} = \{a_1, \ldots, a_N\}$ that is most likely given the sequence of T d-dimensional feature vectors $\mathbf{y}_1^T = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$:

$$\hat{\mathbf{a}} = \arg\max_{N,\mathbf{a}} p(\mathbf{a}|\mathbf{y}_1^T) = \arg\max_{N,\mathbf{a}} p(\mathbf{a}) p(\mathbf{y}_1^T|\mathbf{a}).$$

figure 3.4 shows the difference in the understanding of the model structure to the HMM. In speech recognition, the labels are used for building blocks of words like “phones”, “triphones”, etc., their expressions in building automation could be footsteps, moving objects, or other direct sensor-derivable situation-building blocks. The conditional probabilities in the acoustic model $p(\mathbf{y}_1^T|\mathbf{a})$ can be interpreted in the same way as the scenarios presented here in chapter 5. The most straight-forward extension of the HMM can be found in the hidden semi-Markov model HSMM (see figure 3.5), where the HMM is just extended by a distribution over state durations ((RM85)). Formally, the likelihood of the random-length observation sequence of the general sequential model

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_T, l|a) = p(\mathbf{y}_1, \ldots, \mathbf{y}_T|l, a) p(l|a) = b_{a,l}(\mathbf{y}_1^T) p(l|a)$$

is a joint probability distribution consisting of a family of emission probability densities that describe observation sequences with different lengths and the duration distribution. $b_{a,l}$ is called likelihood of the segment given label a and length l. If we assume our model has one emission probability distribution per state and our observations are independent and



hidden Markov model

segmental model

Figure 3.4: An HMM generates an output emission y per state s, while a segmental model generates a variable-length sequence of output emissions $\mathbf{y}_1^T$ per label a.

identically distributed (iid), then the likelihood of the segment becomes the product of the probabilities of the observations

$$b_{a,l}(\mathbf{y}_1^T) = \prod_{i=1}^{l} p(y_i|a)$$

and the segment model is reduced to a HMM with explicit duration model $p(l|a)$ as opposed to the geometric duration distribution of a HMM.

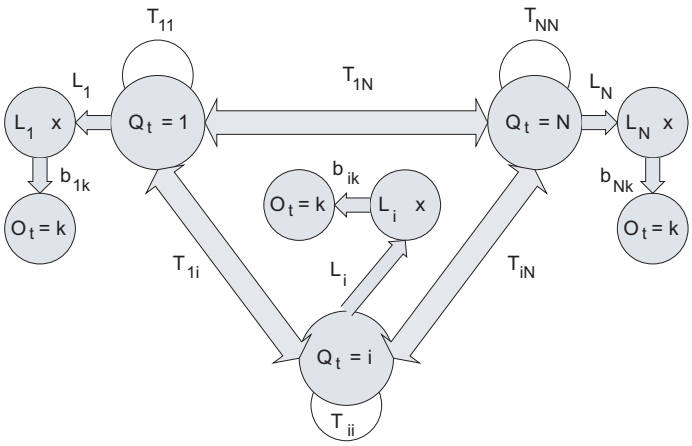


Figure 3.5: The hidden semi-Markov model. It consists of N states with transition probabilities, emission probability distributions over output symbols and duration probability distributions that give a number of repetitions every time each state is visited.

The previosuly mentioned HMMs and their algorithms are implemented in later chapters. For future continuations of this research I have provided the HSMMs as a starting point.

Chapter 4

Ubiquitous Computing Case Studies

In this work, I want to introduce machine learning techniques to building automation. I am convinced that the methods presented in the first chapters of this thesis are the necessary ones out of the plentitude of possibilities and have therefore given an overview of the mathematical background thereof. To locate my work I have also stated the developments in ubiquitous computing.
In later chapters, I will introduce different ideas on how to use these methods, and how to combine them with standard building automation systems and methods. Therefore, I will frequently refer to examples to illustrate how the techniques can be used and what is the expected outcome.
In the following two sections I will present two case studies on ubiquitous computing systems. Both have their origin in the ARS project and can be seen as first steps towards disseminating and commercializing the ideas of the intelligent environment.
These two projects clearly describe current developments, aims, state-of-the-art and visions on ubiquitous computing. It is essential in the later chapters to have the same understanding of the terms and ideas of ubiquitous computing when the approaches and solutions will be discussed.
As mentioned in section 1.2, AAL (Ambient Assisted Living) is an upcoming topic in funding programs both on national and European level. This can be seen e.g. in the 7 challenges of the IST priority of the European Commission's 7^{th} research framework programme (FP7):

- Pervasive & trusted network & service infrastructures
- Cognitive systems, interactive environments & robots
- Higher performance & reliable components & subsystems & embedded systems
- European digital content and knowledge factory
- Sustainable and personalized health care
- ICT[1] for mobility, sustainable growth & energy efficiency

[1] In this case ICT stands for Information and Communication Technologies

- ICT for independent living and inclusion

Nearly all of these challenges are more or less related to the prerequisites of intelligent environments. Because of the appreciation of this topic on the European level on the one hand and the search for suitable consortia with the necessary background on the other, the two mentioned projects - SENSE[2] and SEAL[3] - were prepared for the 6^{th} European Framework Programme.

4.1 Case study 1: Security in public spaces

The SENSE project (Smart Embedded Network of Sensing Entities) will develop methods, tools and a test platform for the design, implementation and operation of smart adaptive wireless networks of embedded sensing components. The network is an ambient intelligent system, which adapts to its environment, creates ad-hoc networks of heterogeneous components, and delivers reliable information to its component sensors and the user. The sensors cooperate to build and maintain a coherent global view from local information. Newly added nodes automatically calibrate themselves to the environment, and share knowledge with neighbors. The network is scalable due to local information processing and sharing, and is self-organizing based on the physical placement of nodes.
A test platform for a civil security monitoring system will be developed as a test application, composed of video cameras and microphones. The test platform will be installed in an airport, to yield real data and performance goals from a realistic test environment. Each sensor is a stand-alone system consisting of multiple embedded components: video system, audio system, central processor, power source and wireless networking. The security application will implement object/scenario recognition (e.g. baggage left unattended, people "lurking" in an area). Nodes will recognize local objects, using a combination of video and audio information, and neighboring nodes will exchange information about objects in a self-organizing network. The result is a global overview of current objects and events observed by the network.
The key innovative aspects are the methods by which the network perceives its environment, fuses these perceptions using local message passing to achieve local and global object recognition, and calibrates itself based on its environment. Challenges include perception, adaptation, and learning, as well as tools to diagnose and maintain a self-adapting distributed network of embedded components.

4.1.1 SENSE high level objective

The objectives of SENSE can be characterized in one sentence:

> To create a system in which distributed and embedded devices cooperate to form and maintain a self-consistent global world view from local sensor information, and which is robust to the addition and removal of devices from the network.

[2]The SENSE project's proposal was written mainly by my colleagues at ARC Seibersdorf research GmbH - Dr. Brian Sallans and Dr. Gerhard Russ - and by me. Their research institute (`www.smart-systems.at`) is now coordinating the project.

[3]The proposal for SEAL was also initiated by that group, but in the early stages we approached Prof. Zagler from the Institute "Integrated Study" at the Vienna University of Technology, who then assumed the role of the coordinator.

The concept of "ambient intelligence" and "ubiquitous computing" follows directly from current trends in computation: miniaturization, cost reduction, increasing computing power and wireless networking. Traditional system design and planning is appropriate for monolithic, centrally-planned projects. However, ubiquitous computing implies a system that develops organically over time, with the constant introduction, modification and removal of nodes during the lifetime of the system. For such systems, planning and maintenance responsibility must be shifted as far as possible away from the system designer, and to the systems themselves.
In order to achieve the vision of ubiquitous computing, ad-hoc networks of embedded devices must be able to:

- Self-network without centralized planning;
- Incorporate new devices into the network without modification to older devices;
- Adapt devices to changes in the working environment;
- Communicate at a device-independent level;
- Understand their operating context through perception of the environment

The embedded devices themselves must:

- Be unobtrusive and provide service without disturbing the public
- Operate with as little installation and maintenance effort as possible

The SENSE project will realize such a system, and test it in a busy and real environment (the International Airport Krakow). The system is based on intelligent nodes, which perceive their environment using audio and video sensors. The sensors form a network to exchange information about the environment at a semantic level, independently of individual sensor types. The intelligent nodes adapt themselves to their environment, including the presence of other intelligent nodes.
At the application level, the SENSE project will implement a network for civil security applications. The network will automatically detect unusual situations without requiring these situations to be pre-defined. In addition, the system designer will be able to provide alarm conditions that should also be detected.
The SENSE system will be grounded in the audio and visual surveillance of a data-rich and complex environment: an airport terminal. Each node in the network will perceive its local environment through its sensors, and convert this sensor information to local features. These features will be fused into semantic concepts at the local level, and made available to neighboring nodes, informing them of the current local state around the sensor. The key to the SENSE network's ability to operate at the semantic level is that these semantic concepts will be formed by the nodes themselves, based on a statistical analysis of their surroundings. Nodes will then incorporate their neighbor's views into their own local view. The result will be a global, self-consistent world view of the environment perceived by the entire network.
The system will also be flexible and scalable to respond to

a) Changes in the construction of the local environment (differencies in the flow of people, for example).

b) Changes in the local knowledge and topology of the sensor network (removal, addition or relocation of sensors).

4.1.2 Specific scientific and technological objectives and state of the art

SENSE will address the following scientific and technological objectives:

- To understand how to build networked systems of embedded components that can dynamically and automatically re-configure themselves
- To understand how to convert low-level local information to semantic knowledge
- To understand how to use semantic-level knowledge for network-centric computation
- To understand how a shared semantic vocabulary influences dynamic node discovery and configuration
- To understand how perception and information processing can be combined using low-level and high-level feature fusion
- To understand how to facilitate networks of heterogeneous devices using a high-level semantic layer

Embedded computer systems have seen their computing power increase dramatically, while at the same time miniaturization and wireless networking allow embedded systems to be installed and powered with a minimum of support infrastructure. The result has been a vision of "ubiquitous computing", where computing capabilities are always available, extremely flexible and support people in their daily lives.
Although hardware and networking capabilities have increased dramatically, one necessary piece of technology for ubiquitous computing has generally lagged behind. The intelligent middleware embedded in smart devices for resource discovery, adaptive configuration, flexible cooperation and high-level perception has not kept pace with hardware advances. Some advances in this direction have been made. Resource discovery and dynamic networking are active topics of research in embedded systems, and perception and adaptation are current topics in robotics and artificial intelligence.
The goal of SENSE is to combine these two aspects in a common framework of semantic knowledge discovery and sharing. The SENSE system will encompass aspects including:

- construction of a modality-neutral embedded test platform;
- raw sensory processing;
- transformation of sensory data into semantic knowledge;
- sharing of knowledge between intelligent nodes;
- automatic discovery and configuration of new intelligent nodes;
- automatic recognition of unusual and alarm situations;
- communication between nodes to produce a consistent world view; and

- communication between the intelligent network and an operator.

There is a large body of literature on low-level processing of sensor data for embedded systems. This is often done in a robotics context, where the sensor data is then used to directly support decision-making. Transformation and fusion of sensor data into semantic concepts is also common in robotics, as well as in research on multi-modal interfaces. However, unlike the SENSE system, the domain of operation of the system, as well as the semantic concepts used by the nodes, are fixed in advance. We are unaware of embedded systems which, like SENSE, develop their own semantic symbols based on an analysis of their environment. SENSE will incorporate research from machine learning to discover statistical regularities in its environment, and compress these regularities into informative semantic symbols. At the local level, SENSE will use algorithms such as "Expectation-Maximization" to optimize each node's set of semantic symbols.
Sharing of knowledge between nodes is also a topic of research, both in distributed systems and artificial intelligence. The SENSE system will use a mature algorithm called "belief propagation". This algorithm specifies how to share probability distributions over semantic concepts between nodes, such that a self-consistent world view results.
The unique feature of SENSE is that it combines these various technologies from embedded systems, robotics, networking and machine learning research in a new way. The result is a framework for the development of smart networks of embedded components, which are flexible, adaptive and device-independent.
"The falling costs of bandwidth and computing have enabled the growth of large-scale computing networks. For example, we are now moving towards decentralized systems in large computing networks, in telecommunications, and defense. A challenge for networks engineers is to develop networks that can configure themselves and adapt to changing demands and environments (ACGHW) . . . "
Networks that cover those challenges are called ad hoc networks or self-organizing networks. Their development is driven by the wireless community, but some of their principles are also of interest for wired networks. Advantages can be seen in a reduction on effort required for installation, initialization and maintenance as well as their inherent fault tolerance and their possibility to save energy within the network. This is true for both wired and wireless types but typically only relevant for the latter. In contrast to self-organizing networks, traditional networks have a very time-consuming commissioning phase, which also involves expert knowledge. Further disadvantages arise when nodes are added or removed. In ad hoc networks, connections are constantly created and destroyed. This is called "plug and participate".
SENSE will progress the state of the art through innovation at all levels varying from raw sensor data to high-level abstraction, reasoning and interpretation:

Development of a distributed processing solution to large-scale systems design. SENSE tackles the problems of scalability and complexity by completly decentralizing of both processing and knowledge, relying upon the fusion of information at a high semantic level to allow computation.

Automatic learning of semantic symbols. Unsupervised learning is a key feature of SENSE. The sensory input used by the system will be decided by the system itself. Although some a priori knowledge must be given to the system, SENSE attempts to make its own decisions about what it senses, how it should describe what it senses, what are normal activities and how best to share this information with other sensing elements of the system

Transferability between application domains. By providing no "hardwired" knowledge about either architecture or the specific goals of the system, SENSE presents a generic approach to large-scale system development. In addition, traditional hardwired design requires a thorough understanding of the application and objectives of the system. In the SENSE system, this is replaced by flexible adaptation to changes in the environment.

Semantic abstraction. SENSE decides what is important and what is normal, and automatically generalizes to provide its own view of the world.

Generic solutions to systems design. SENSE addresses two specific forms of sensor data (audio and visual); however, the low-level extraction of features (on which the semantic levels operate) ensures that developments in the higher layer are applicable to any form of sensorial data.

Reliability in complex systems through distributed processing. Reliability in traditional applications suffers because of their own complexity. The failure of a single component may cause the breakdown of the entire system. In SENSE, each sensor is an autonomous entity, and the overall system adapts to changes in the structure and to the loss of components.

4.2 Case study 2: Security, care and comfort for the elderly

The SEAL project (Smart Environment for Assisted Living) will develop a system to prolong independent living for the elderly. The system is an intelligent, adaptive network of wireless sensors and actuators, which is installed in the home and monitors the daily activities of the elderly user. The system adapts to the behavior patterns of the user, allowing it to detect abnormal or dangerous situations; to assist with common tasks; to increase comfort and social inclusion; and to help with early detection of emerging medical conditions. The system can provide information to the primary user, and to secondary users (such as informal caregivers and family) via local or remote interfaces.
The user will play a central role in all aspects of the project. Four European experimental sites will be used to test the system with its intended audience. Regional differences in long-term care and the needs of primary and secondary users will be integrated via user input during design, test, and implementation. The SEAL system will be based on wireless networks, making installation in the user's home unproblematic. The system will be designed by extending proven building automation technology, making development as fast, inexpensive and reliable as possible, while also supplying ready-made channels for eventual product distribution. By using existing building automation systems as a basis, and by providing resources for third-party developers, SEAL will be open to both hardware and software extensions. Traditional user interfaces (such as touch screen or mobile phone) will be combined with voice control to provide user-friendly system interaction.
The key innovative aspects of SEAL are the project-wide focus on ethics and user needs; advances in human activity recognition; and improvements in voice control and home automation. The result will be a working, marketable system which can be installed in the home to assist elderly people in their everyday lives and prolong their independent lifestyle.

4.2.1 SEAL high-level objective

The continuing increase in average longevity creates great and new opportunities for individuals to fulfill their potential and their role in society. At the same time, the demographic change of an aging population poses new challenges to the sustainability of public and private long-term care. The general goal of the Project SEAL is to develop a system that by monitoring of ADL patterns (ADL = Activities of Daily Living), by enhancing building automation and by fostering communication, will support challenged persons to stay longer within their familiar and independent living environment before a transition to institutional care becomes necessary.

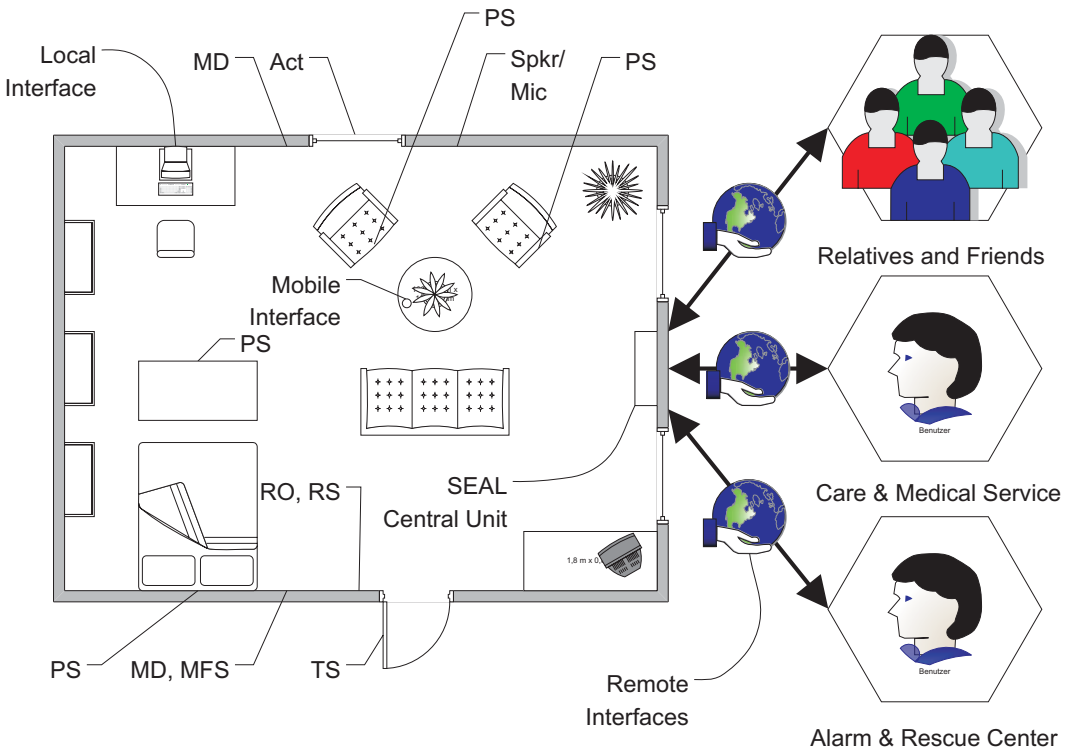


Figure 4.1: The SEAL System and its components

User Interfaces: Local Interface, Mobile Interface and Remote Interfaces
Sensors (examples): MD: Motion detector, PS: Pressure sensitive switch,
SW: Simple on/off switch, TS: Touch sensitive switch,
SO: Smart object (e.g. with accelerometer)
MFS: Multifunctional Sensor: e.g. Temperature, light, humidity, sound-level, smoke detector, gas (CO, CO2) detector
Actuators: Act. e.g. for moving blinds, door opener, window opener, DL: Door lock
RO: Remote controlled outlet (sensor for voltage and current (= power consumption) and actor for remote switching)
RS: Remote controlled switch (e.g. for lights)
Communication and Control: Spkr/Mic: Loudspeaker Microphone combination

The implementation of the system in the existing home of the user will on the one hand contribute to the persons' safety and security and on the other hand increase their comfort and the quality of life. The system will be based on a network of wireless and distributed sensors and actuators and include features such as intelligent learning of normal and exceptional patterns of behavior (dangerous situations or indicators for emerging health or social problems), raising of alarms and control of elements which are typical for a smart-home environment. The project is devised to be guided by a set of straightforward basic principles and objectives: We know and we positively take into account that ...

- investments into the living environment of older persons not only prolong their time of independent living but will pay off often within weeks due to the otherwise high costs for institutionalization B.8.2.
- medical experts agree that emerging medical conditions (physically as well as mentally) can be detected in changes of the ADL patterns before they become critical (TIL04).
- a vast majority of aging persons definitely prefers to stay in the familiar living environment as long as ever possible shunning to be transferred into a form of institutionalized living (Neu99).
- high rates of transitions into institutions are caused by few and often simple reasons, and that a felt deficiency in safety, security and peace-of-mind by the person but even more often by relatives triggers the decision for institutionalization (Sch98).
- acceptance of technology is not mainly a question of age but of clear noticeable benefits which a new product or service can introduce to daily life (DRA04).
- plenty of good groundbreaking work for technologically based support for older persons has been done and there is a well-founded state-of-the-art in home automation and monitoring which is ripe and awaits exploitation (FMM+03; Ber99; FH99; HY02; TMIL04; BIT04).

For the above reasons, the proponents of the SEAL Project want to ...

- find out and tackle the 20% or 30% of reasons which are responsible for 70% to 80% of the transitions into institutionalized life, bearing in mind that not all problems can be solved in a three year's project and thus a pragmatic concentration on the essentials is necessary.
- take and use as many "off-the-shelf" experiences and products as possible and mold them into a practical solution to give the project a head start in the development. The goal shall not be to re-invent the wheel but to finally place the vehicle on track and get it running.
- carefully and responsibly assess and respect the users' needs and preferences in order to come up with a solution that is readily accepted for its noticeable benefits and its user-oriented design.
- achieve a cost-benefit ratio which will ensure that an investment into the system will in average pay already off if only a six months' delay of institutionalization can be achieved.

- obtain sustainability in such a way that the system can gracefully and cost-efficiently be implemented and integrated into existing housings with no need for costly modifications.
- obtain additional sustainability in such a way that the system can be scaled and modified at any time of its life-cycle, i.e. to be adapted to changing needs or even be transferred to a new location.

Thus our expectations for the project are ...

- to come up with the appropriate design and specifications after nine months of careful study of user needs, expectations and preferences and a straightforward translation into the technical specifications by scenario analysis.
- that a permanent guidance of the project by user involvement and a mindful observation of the ethical factors will finally lead to acceptance and success.
- that an intensive testing with real users in the field for the duration of almost half a year will not only lead to technological refinement but also pave the way to acceptance and market-take up.
- that after the three years of intensive project work (2007-2009) the time-to-market can be contained (12 to 18 months) due to the significant commitment of our industrial partners.
- that by deliberately concentrating on the essential and the effective, the system will have the potential to positively influence the social situation of an aging Europe from 2011 onwards.
- that by disseminating the benefits of the SEAL system to stakeholders in Europe during the field trials, successful national scenarios for financing or reimbursement of the system will be developed.

4.2.2 Specific scientific and technological objectives and state of the art

The goal of the project is the development of an environment supporting older persons in their everyday lives at home and, as a consequence, to achieve social as well as economic benefits. This major goal can be divided into three groups of benefits:

- Social benefits
  - Increase independence
  - Increase communication
  - Reflect national factors
- Technological benefits Establish
  - Viable context-aware systems
  - Speech control of environment
  - New sensor and actuator technology and

  - Reliable communications systems

- Economic benefits
  - Reduce cost of long-term care
  - Establish Europe as leader in AAL
  - Promote European economy through new AAL products and services

These benefits will be achieved through implementation in the project Workpackages.

4.2.2.1 Technological Project Objectives

SEAL will develop a context-aware environment targeted at older persons and people with disabilities. It will develop techniques for processing and analyzing information collected through a network of distributed sensors and actuators as well as wireless communication devices/services. Therefore, the data can be interpreted and the environment becomes aware of how the users interact in their homes. As the system is learning, possible exceptional user behaviors become identifiable. This context-aware environment will be evaluated in real-life settings (existing homes), while target users perform their daily activities. Evaluation includes system usability (e.g. system functionality, transparency, reliability, efficiency, scalability, and extensibility), user behavior, and organizational aspects of using such a system at home. The end-goal is a system which can prolong the period of living at home, reduce the workload for caregivers, reduce costs to the public health system, and be affordable for the end-user.
The specific technological objectives are:

- Improve usability by incorporating local and remote user interfaces, as well as improving noise suppression by adaptive blind source separation, allowing robust voice control.
- Improve interoperability by providing standardized interfaces for third-party vendors of services and devices.
- Enable the user to control objects in their home in an easy and context-aware manner by developing new sensors and actuators, as well as intelligent context-aware controllers.
- Develop fast and dependable machine learning algorithms for the modeling of daily routines and detection of unusual or dangerous events.
- Enhance detection of dangerous situations by reorganizing redundant sensor data into robust semantic concepts.
- Ensure privacy and high reliability of the system through network and protocol design.

The progress towards these technological objectives can be measured within the project with reference to the project deliverables.

4.2.2.2 State of the Art

State-of-the-art telematic health monitoring systems already provide health-care professionals with remote monitoring, diagnosing and even health care provision to patients in their home environment. Such systems are presently relying on proprietary combinations of advanced remote monitoring devices, telecommunication technologies for distance monitoring of vital signs at the patient's home, thereby delivering increasing savings of health care's ever-increasing costs. Yet, these systems are typically limited to the communication of alphanumeric data, such as temperature and blood pressure.
Current state-of-the-art context awareness and inference about activities of people can be separated into research and commercial products. Research includes behavior-based adaptive home control (Moz05), security applications (ORP00), smart meeting rooms and offices (OH05), as well as the topic of assisted living (Wil03). This stream of research is based on using machine learning techniques to either adaptively recognize specific behaviors, or detect unusual situations. Most are video-based, although the neural network house (Moz05), and the assisted living system from Honeywell are, like SEAL, based on common building automation sensors (motion, temperature, luminosity). Although the general machine learning modeling methods are well known (Rab89; RHW88; CV95; Pea86), their use is very application-specific, depending on behaviors to be recognized, type and number of sensors, required reliability, and so on. The various methods in the literature must be investigated in the AAL context to produce a reliable and useful system.
Commercial products include some similar categories, specifically home automation (BP01), security applications (Weg98), and safety in swimming pools (CGMR05). Other mentioned examples have remained only topics of research, with no current commercialization. Ambient assisted living is one such topic that has not migrated from the laboratory to a commercial product.
The state-of-the-art in multi-modal interfaces has moved from dialogue systems to multimodal interaction frameworks, where the processes for conversational systems based on natural language are augmented with semantic-level content, such as that provided from tactile and/or visual input modalities. The project SmartKom[4], for example, has set a holistic framework for symmetric multi-modality in mixed-initiative dialogue systems along with other approaches. SmartKom employs the same software architecture and components in three fully operational application scenarios: a home/office working environment, public access to the Internet and to information services, and a mobile device (basically a well-advanced cell phone). Other interesting projects that will be studied within SEAL are SHARE[5], an ongoing project aiming at providing a new interaction mechanism for mobile multimedia content sharing, and INSPIRE[6], which integrates a multilingual, interactive, natural, speech dialogue-based assistant for wireless command and control of home appliances (e.g. consumer electronics) from several points inside a house, and from remote locations through the telephone network or a packet-based network.
The infrastructure to support the SEAL environment inside the house will rely strongly on the so-called industrial communications (Zur05). Its use in building automation is already well settled (MM02; KNSN05) when relying on cabled solutions. However, the emergence of building automation for assistive technologies (SZB04) makes a strong push towards the use of wireless technologies and requires special attention on dependability and security issues.

[4] `www.smartkom.org`
[5] `www.ist-share.org`
[6] `www.inspire-project.org`

The utilization of wireless communications as a support for fault-tolerant communications is still being delayed by many problems (Dec02; WMW05), such as the ones derived from interferences. Nevertheless, standards well-settled in the IT domain such as Bluetooth or WiFi (IEEE802.11) or emerging standards targeted for building automation such as ZigBee are promising and the former are already being explored for real-time applications (BFD05; Ni05).
The state-of-the-art approaches to robust automatic speech recognition are based on several techniques including: speech pre-processing algorithms to remove the noise from the incoming speech and to provide a clean version to the ASR engine, acoustic modeling of noisy speech based on the training of the ASR engine on large samples of noisy speech and using large speech databases, integrated approaches such as multi-channel speech processing and recognition based on the exploitation of robust multi-channel mechanisms known in human hearing, robust speech capture systems, based on microphone technologies (microphone technologies range from microphone arrays to throat microphones that might be used in cases the environment is noisy (e.g. TV is on) or the user cannot speak loudly and clearly). Speech collection has been enhanced using beamforming-approaches.
The state of the art in converged (voice and data) communication systems (both wired and wireless) is now rapidly progressing towards all-IP systems. However, streaming data communication in the context of interactive applications like voice and video interaction presents significant difficulties with respect to the quality of voice and video communication sessions since the IP family of protocols was not designed for streaming data transmission. The problem is particularly severe in transmission-critical applications such as security alarm and health monitoring data transmission where delays and even loss of connection might be life threatening. The problem predominates in the widely popular WLANs with presently only proprietary solutions. However, the problem of quality of service of multi-media communications has recently received increasing attention both from the industry as well as standardization bodies (IETF and others).

4.2.2.3 Expected Innovations

The most important innovation of the SEAL project will be a commercially viable Ambient Assisted Living technology which can be installed in the private home or institutional care facility. This will result from a mix of experienced research and technology partners, commercial partners committed to further development and exploitation of the technology, and technological innovation.
In addition to the overall goal of pushing AAL technology out of the lab and into the home, the SEAL project will result in a number of expected social and technological innovations as well as economic benefits for both the public and the user.

Project-wide Focus on Ethics and User Needs

A large part of the project effort will be expended on research into ethical questions of data protection, privacy, and interaction between the system and the direct and indirect users (formal or informal caregivers). This focus on ethics and user needs will feed into all aspects of the project, from user interface design to system specifications, network design, and user modeling. The goal of SEAL is not just to develop another AAL technology, but to develop a system which is usable and acceptable to the user and society.

Human Activity Recognition

SEAL will further the state-of-the-art by applying machine learning methods to model the daily routine of users based on standard building automation sensors. Our work will be distinct from previous efforts in several ways. We will use standard motion, temperature, luminosity and door sensors rather than expensive cameras. The system will be dependable while still being fast and running on small embedded or industrial platforms. The system will be open to expansion both to new software functionality and new sensors. The result will be a system that is usable, affordable, and easy to install. This will allow us to move the system out of the lab, and offer an ambient assistance solution which will increase the welfare of the European citizens while allowing time and cost savings for national health plans, insurers and caregivers.

Voice Control

Usability is a key factor to increase system acceptance. SEAL will be innovating the areas of dependable voice control by combining commercial-grade voice recognition with multi-microphone methods for blind adaptive noise reduction, i.e. blind source separation. Ordinary methods for multi-channel signal processing presume prior knowledge about the specific positions of the speakers, e.g. in terms of from which direction the source signals are coming. In contrast, blind methods are able to adapt to different source positions without knowing the positions of the speakers or noise signals, respectively. A powerful criterion for the discrimination between coherent signals is given by the degree of statistical independence (ICA Independent Component Analysis) which is exploited also in other areas of signal processing for blind separation of mixed signals. Via maximizing statistical independence between output signals, the source signals can be reconstructed. In order to track moving speakers, different scenarios or misadjusted microphones, it is necessary to develop robust methods for adaptive blind source separation of convolutive mixtures. For this purpose, spatial information will be exploited not only for dealing with frequency permutation problems, inherent in frequency domain ICA-methods, but also for identifying speaker positions. Additionally, the development of efficient adaptive joint diagonalization procedures leads to online methods not showing the usual channel permutations over time.

Home Gateway Platforms

The SEAL project will be innovating the areas of distributed and embedded systems, industrial communications and system integration. The partners will build on already existing telecommunications knowledge, and also define and develop new interfaces and a new complete central management and intelligent collecting device, which also acts as a mediation device for other external and internal downstream processing systems, such as governmental statistical department, health & care centers, or emergency organizations. Innovative contribution to intelligent home gateway platforms (like (Ski02), who describes a control infrastructure for HAVi devices via Jini, or (Wei06), who allows multimedia streaming over various networks) will be in integration of advanced standards and methods for an open all-IP-based telecommunications and home networking platform (either from industry or from project development) taking full advantage of the progress of ongoing standardization in QoS (specifically IEEE802.11e for WLANs) for multi-media alarm signaling, based on the

HW/SW adaptation of an existing sbcPC platform. Another innovation will be the design and prototype execution of a HW-optimized integrated home gateway platform tailored to the functional and technical needs of the project based on selected microprocessor technology supporting multimedia event recognition and communication.

Semantic Information Processing

The project contains great potential for scientific innovation, supported by previous work in the field. The key innovation is the combination of two different approaches: symbolic information processing by means of rule-based mechanisms is used for high-level semantic processing, while the generation of this symbolic information is done by using real-world sensory input. The expected outcome of this combination is a system that is able to use symbolic algorithms (e.g. first-order predicate calculus) with symbols that are created based on sensory data (e.g. by statistical methods or neural networks). The creation of such a symbolic world representation is supported by an ontology that contains the relevant facts, which are used for associated symbolic information.

Chapter 5

Model Structure

In the previous chapters the reader has been given an overview of the fields of ubiquitous computing and several methodologies from statistical pattern recognition and hidden Markov models. In this one and the following chapter of the thesis, several approaches to combine this knowledge for the extraction of semantic meaning from sensor data are introduced. As a reference for comparison, I will use primitive rule-based approaches. More intelligent systems that try to extract semantic meaning via a pre-defined symbolic representation are also an appropriate measure of performance for comparison.
The presented approach is not meant to be applicable to particular applications in this early stage. But it is my strong conviction that many applications require a situation or scenario recognition system with much higher capabilities compared to state-of-the-art systems. Such systems will have (or need) the ability to detect what is going on, and what are the intentions of persons. In a football stadium, for example, the general mood of the crowd, recent discussions of controversial political decisions, the current condition of the teams, their financial situation, their position in the national or international leagues, and whether or not the fans are intoxicated, etc., will affect the behavior of the crowd. A system that has (or is intended to have) the ability to enhance security in such an environment ideally would anticipate the impacts of all of these and many more factors.
In the following section, Statistical Generative Models (SGMs) are investigated regarding their possible use in the building automation domain. Descriptive examples depict the field of possible applications for particular types of models and sensors.
section 5.2 bridges from standard rule-based systems to the introduction of statistical models in this domain. These models can be used as the lowest level of a hierarchical system presented later.
The second section describes an approach of modeling events and scenes for which the system designer has much information. The type of events as well as a rough idea of the time of occurrence is assumed.
The third section explains how to model events and scenes that happen at arbitrary times with arbitrary frequency. A more flexible transition framework for the basic models is illustrated. The events themselves have to be known.
In the last section, a model structure that allows the modeling of fully unspecified events is introduced. A particular implementation of this model is described in the next chapter.

5.1 Terminology

Before I present the description of the models, a few terms[1] have to be specified to generate a common understanding of the architecture of a perceptive system, of which this thesis covers a large part:

A bionic approach similar to the mental processes in the human brain is used[2]. The whole system is called **perceptive system**. While permanently abstracting and combining symbols, the perceptive system tries to represent the sensed environment in a hierarchical way. This approach is shown in figure 5.1

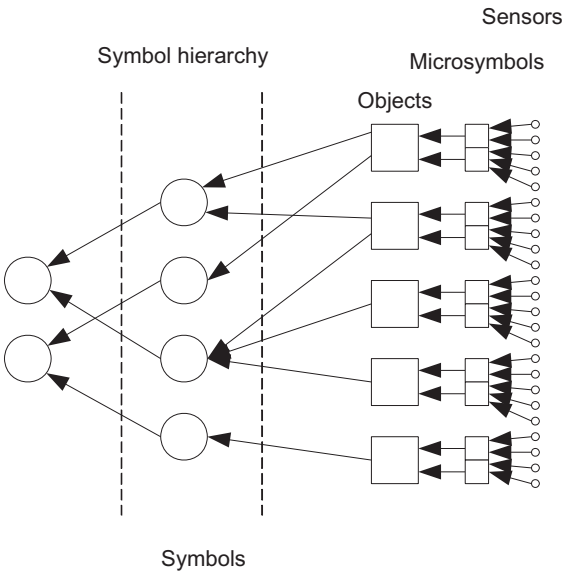


Figure 5.1: The perceptive system uses a bionic approach to abstract from sensor readings via objects into a hierarchy of symbols.

The perceptive system senses physical properties from the environment via **sensors** and in a later phase influences the environment via **actuators**. The sensor readings are combined to **microsymbols** - like footsteps or temperature ranges, etc. Several microsymbols form the representation of **objects**. The difference is that an object contains the full representation of a physical object like a person, a table, etc., while the microsymbol contains only the information of a single sensor which can even belong to several objects.

With permanently abstracting and combining **symbols** and objects, the perceptive system tries to represent the sensed environment in a hierarchical way. Objects and symbols

[1] These terms are already partly used in the ARS project on defined in (Pra06), but the approach and viewpoint are slightly different there. The main differencies are: (Pra06) does not distinguish between objects and symbols, and his concept of snapshot symbols implies the concepts of images and situations presented here. Also, no predictions about the future are intended.

[2] The intention is not to build a mind with human-like capabilities, but to use the same principles that appear in the human brain - found by psycho-analsysts - to build technical systems. This idea is explained in depth in (DLP+06; PPDB05; RLFV06)

have **properties** and **associations**[3]. These two concepts are also applicable for higher level representations introduced soon. I see the associations as pointers to similar properties. An association-finding process is needed to run in the background. It permanently searches for such connections. An example of an environment, its sensors and its representation in the perceptive system is given in figure 5.2.

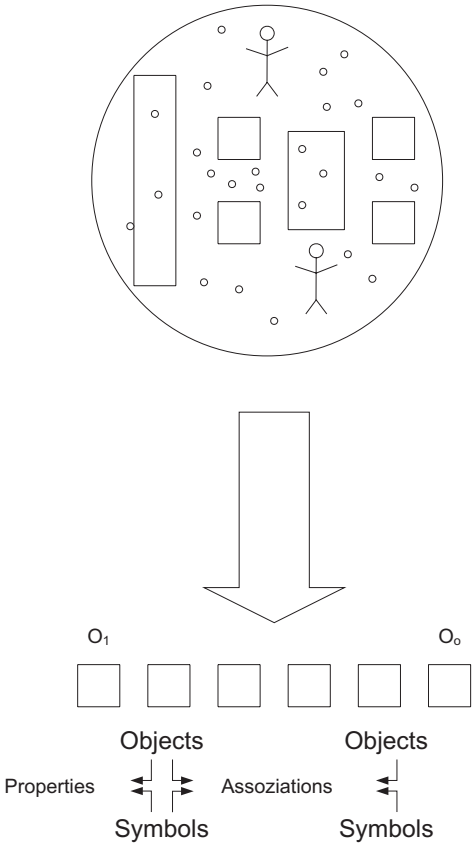


Figure 5.2: An example of how the perceptive system works. The top area shows a kitchen with a table, chairs, people and a sideboard. The small circles indicate sensors. The sensor readings are used to create microsymbols and combined to create objects that - in the ideal case - represent the whole environment as a human would. During the creation of the objects, properties are assigned and associations are sought. A property belongs to either an object or symbol, while each association combines at least two of them over similar properties.

All previously introduced concepts are time-independent. To bring a temporal representation into the perceptive system, **images** are presented. An image contains all symbols, objects, or even microsymbols or sensor readings at a particular point in time. For convenience, images have a mental viewpoint and do not necessarily contain all of those symbols.

Past images up to a particular point in time form a **situation** together. The correlation between images and situations is depicted in figure 5.3. The situation concept allows

[3]The associations part is still under investigation while this work is being written. Technicians and psychoanalysts discuss how to make a functional model that can be realized in a technical system.

the combination of several events to happen at different times. An often used example is a child being alone and close to a hot stove. If a parent is cooking and leaves the room and then the child comes in, it should lead to the same situation as: the child and a parent are cooking and then the parent leaves the room, etc.

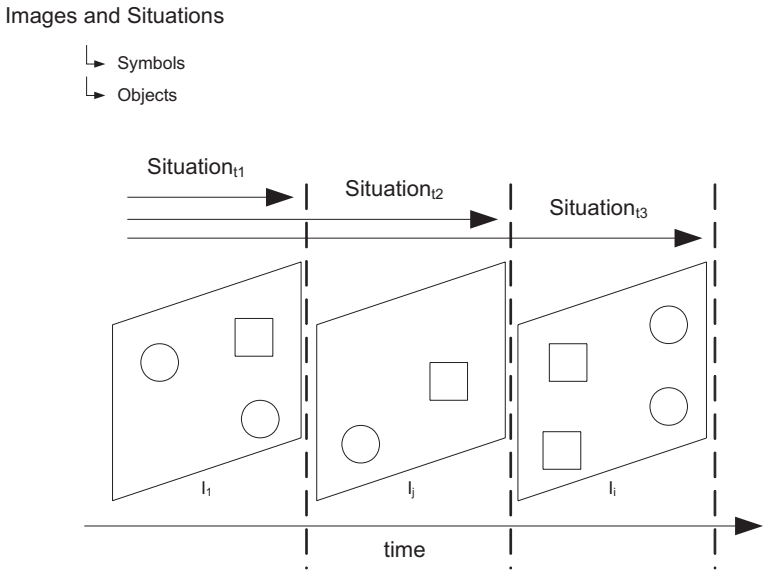


Figure 5.3: Introducing time to the perceptive system. Symbols, objects and subsequents at a particular point in time form an image. Several past images form a situation together.

The - currently - highest level of abstraction used in the perception system is the **scenario**. It consists of a relatively ordered sequence of images (figure 5.4). The goal of this work and many others is to detect scenarios and situations or maybe even to prevent a - dangerous - scenario[4]. This is shown in figure 5.5[5].

With that description in mind, the three most necessary terms can be specified in the following way:

Events happen at particular points in time and have a short duration. Events can be for example the opening or closing of a door, somebody entering a room, falling down or the like. Events are used in situations and scenarios - they change the images. In the following, I only use the terms situation and scenario - which implies the underlying images - and say that events change the situation and therefore affect the scenario.

Situations describe also a particular point in time, but have a past and encompass several sensor values - or require a more global view than events - while events can be typically detected by a single sensor.

[4]With this definition it cannot be the goal of such a system to detect scenarios and react to them because the scenario includes the incident. The correctly formulated task of such a system must be to detect situations that could lead to some scenario and react accordingly. The detection of scenarios is useful for monitoring

[5]Later, I will introduce sub-scenarios, which are parts of a scenario and allow the better modeling of a different behavior into one structure.

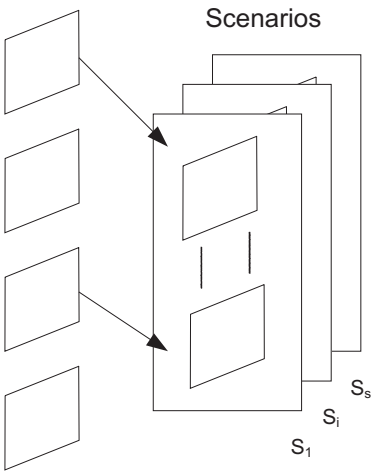


Figure 5.4: A sequence of images forms a scenario.

States are used to group sensor values in time. They are the basic elements to model scenarios.

Scenarios have an arbitrary duration in time and can - in the model representation - consist of an arbitrary number of sensor values. The scenario is the highest level in my model and it consists of several sub-scenarios, which are represented as different paths through the model. While a scenario is happening, each path up to the current state forms a situation and, in general, sub-scenarios or even states within sub-scenarios are subdivided by events (see also figure 5.13).

5.2 Time slice models

This section deals with the modeling of time slices and the introduction of several statistical models in the automation domain. Standard (building) automation systems use their sensor information to search in a set of pre-defined rules whether some of them are violated or not. If a rule violation has occurred, advanced systems classify their alarm e.g. into three categories and launch them via a GUI to the system administrator.
This procedure has clear limitations. The major one in my point of view is that it can only be used for parameters that can be classified easily. Temperature or rotational frequency of HVAC devices for example can be observed well with this kind of systems. Motion detectors or door contacts on the other hand give information that cannot be dealt with in general. Only in security sensitive areas or at times where no presence of humans is allowed these sensors can be utilized for observation systems using rules. However, the motion and presence of humans or the movement of a crowd over time is very specific to particular areas of buildings. The same is the case with information from door or window contacts.
These considerations lead to the idea of modeling time slices. The time slice idea assumes that the behavior of humans in a building is quite similar at a given point in time every day. Therefore, a model is constructed that learns about sensor values within a small time frame on consecutive days. Let me illustrate the idea with an example taken from the SEAL project (section 4.2): A daily morning routine in a flat. It could consist of

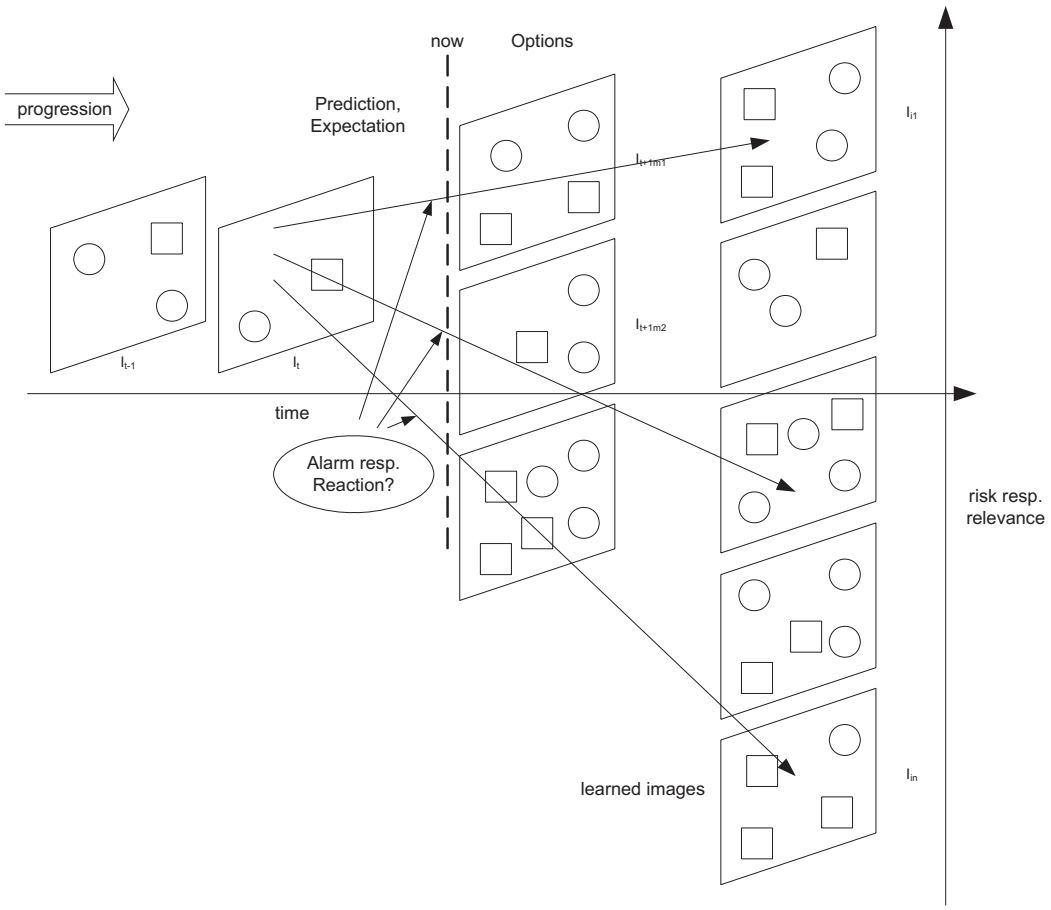


Figure 5.5: Overview of the task of a perception system. Images from the past and up to the present form the current situation. Because of learned or pre-defined scenarios, the system can make predictions about what will happen in the near future and therefore create a list of expected images (the options). If necessary, this procedure can be recursively repeated: assuming the current situation + the first expected image are happening, what could come next? If the system also possesses either a definition of dangerous scenarios or a procedure to set the focus of attention, it can arange the expected images (or the events that will lead to these images) in order of risk or relevance, for example.

- sleeping
- 06:15 : alarm
- 06:22 : 2^{nd} alarm
- 06:23 - 06:35 : taking a shower
- 06:36 - 06:48 : having breakfast
- 06:49 : leaving flat
- no events

- 18:31 : entering flat
- etc.

Let's assume we have all sensory information necessary to detect these events and occurrences. How could a rule-based system diagnose "everything fine"? First of all: the alarm bell. We have to think of rules that fit most people's habits in order to minimize false alarms and maximize correct detections. A quite rough definition would be: The alarm can ring between zero and five times between 4 o'clock in the morning and 10 o'clock in the morning. This definition is so broad that it essentially gives no information. Neither the compliance nor the violation of this rule is useful to any kind of scenario detection procedure or security system. We could go through the list or give other examples but the key issue is that a primitive rule-base is not sufficient for advanced scenario recognition systems.

The time slice procedure divides a day into time slices, e.g. each 30 min long[6]. Within

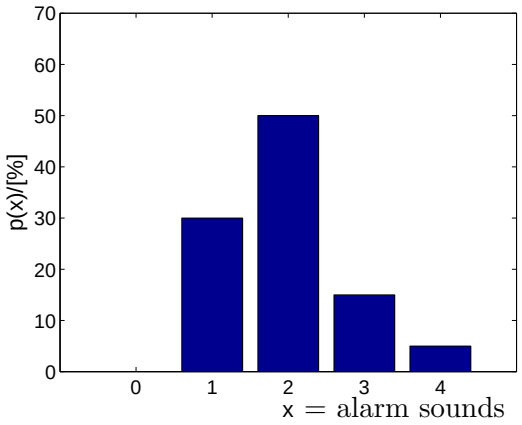


Figure 5.6: Hypothetical histogram of alarm bell counts for one time slice. In this case an alarm bell sound is recognized between one and four times, therefore a missing alarm bell sound would cause an alarm.

this time frame a simple probabilistic model for the sensor is constructed. In case of the alarm bell an event count histogram could give valuable information (see figure 5.6). In this example, a missing alarm bell within this time frame (e.g. the person is on vacation or has set her alarm to another time) would cause an alarm in the observation system.

We can also state that for discrete events with sharp distinctions between event probabilities, a histogram is the model of choice. If we model the same distribution of values with a Gaussian probability distribution, we would get $\mu = 1.95$ and $\sigma^2 = 0.65$ (see figure 5.7). As can be verified easily in opposition to the histogram, there would be a probability of the alarm bell count 0 of greater than 0% and even greater than the probability of the alarm bell count of 4!

The following list gives an overview of simple statistical generative models that can be used for modeling sensors within small time slices. A list of sensors and a description of their time frames - how and how often they generate values - is given in section 6.1.

Histogram

Histograms are just counts of occurrences in sub-ranges of the parameter space. In the discrete example above the parameter itself forms the separation into subclasses. In general,

[6] 30 min seems to be an interval, in which people can subdivide their daily routine in a natural way.

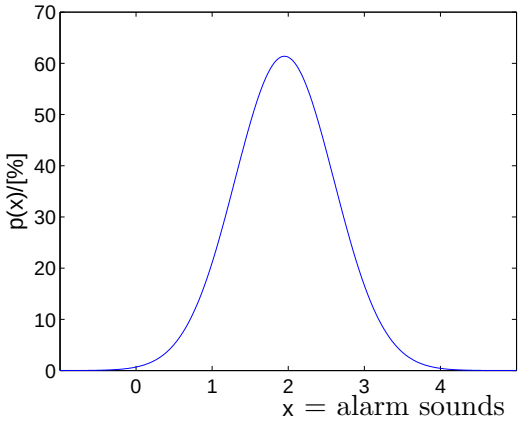
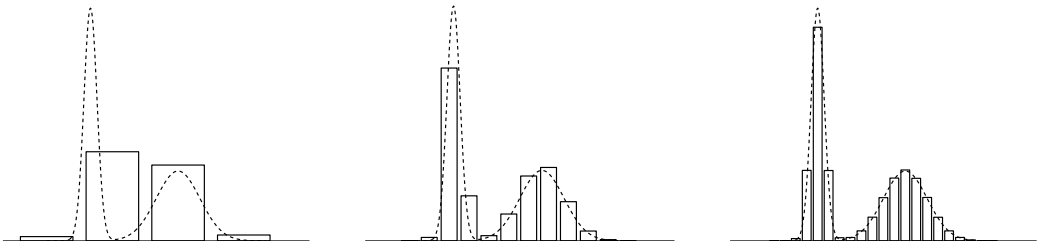


Figure 5.7: Gaussian pdf with the same hypothetical alarm bell counts. The Gaussian model gives a probability for count 0 greater than 0%. The Gaussian model's probability for count 0 is even greater than that for count 4 because of the asymmetric parameter distribution around the value 2 in figure 5.6.



(a) $M = 4$ (b) $M = 9$ (c) $M = 21$

Figure 5.8: An illustration of the impact of a histogram's number of bins. The original pdf consists of the sum of two normal distributions. That curve is plotted as a dashed line. A large sample set was generated by sampling the original pdf. This set was then estimated with use of histograms with 4, 9 and 21 bins, whereby the outer bins were centered at $\pm 4\sigma$ of the respective outer Gaussian component. The number of bins - M - acts as a smoothing parameter.

the operator needs some strategy to find the number thereof and the borders between the particular subclasses. An intuitive approach would be to start with only one class and two border classes for greater and smaller values. If the number of values in one of the borders is of a significant amount, a new class is introduced. On the other hand, if more than a threshold number of the values (compared to other classes) falls into one particular class, this class or all classes should be split. The impact of the number of bins - M - is illustrated in figure 5.8. The estimate on top of the figure could be re-estimated as a simple Gaussian model in contrast to the two quite different ones that formed the original model.

Gaussian model

This model is well applicable for probability distributions where the parameter values are quite symmetrically distributed around a mean value with decreasing occurrence away from the mean. It is represented by only two parameters, μ and σ, mean value or location parameter and standard deviation or dispersion parameter, respectively. The parameters can be computed in closed form for the entire data set as batch or for additional values as on-line update. The on-line update can be accomplished gradient-ascent, moving window average or

decaying average, the update formulas are given in equation 5.1.

$$\mu_{new_{ga}} = \mu_{old} - \epsilon\frac{1}{\sigma^2}(x - \mu_{old})$$

$$\mu_{new_{mwa}} = \frac{1}{T}\sum_{i=t_1}^{t_2} x_i \tag{5.1}$$

$$\mu_{new_{da}} = \frac{1}{N}x_i + \frac{N-1}{N}\mu_{old}$$

When choosing the model to be adaptive, meaning that after an initial phase the model still adapts to changing data conditions, the learning rate - represented in the formula by either ϵ, T or N - becomes a third regular model parameter[7]. The impact of slow or fast learning can be seen by comparing figure 5.9 and figure 5.10. The difference for the operator in this case is the moment of launching the alarm . Neither faster nor slower learning can be selected as the better approach beforehand. The speed of learning depends on the application.

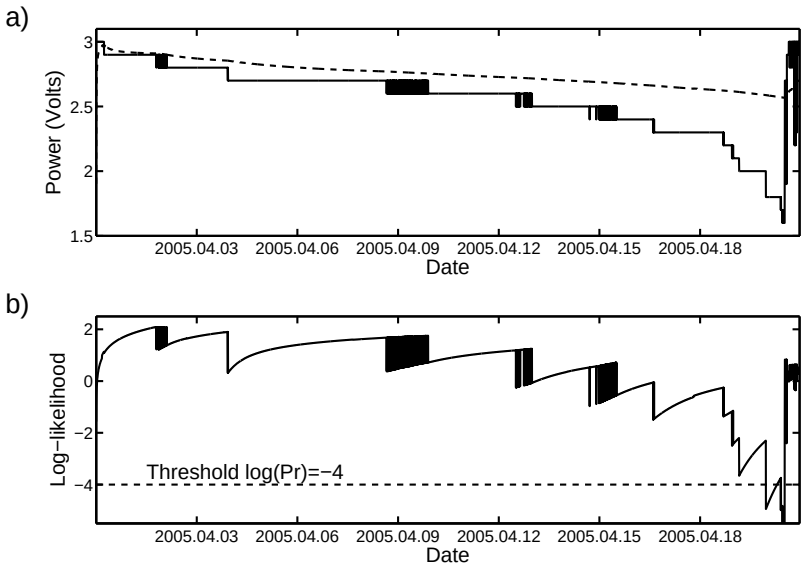


Figure 5.9: A model of the operating voltage of a wireless sensor mote. On top we see the actual voltage and the dashed mean value of the model. On bottom of the figure the logarithm of the likelihood is shown. The rate of decrease of the likelihood is dependent on the speed of adaptation of the model.

Hidden Markov models

In the time slice model approach, HMMs can be used to model the value of a sensor within a small time frame. A predestinated sensor would be a sensor that outputs values that depend somehow on the latest ones according to the Markov property. A very simple example is a door contact sensor that alternates between “0” and “1” as can be seen in figure 5.11.

Rate models

The aforementioned models are basic models for modeling the output value of a sensor. However, in many cases the value of a sensor is just a small part of the whole information of the

[7]The learning rate becomes a parameter and implies the update rule and vice-versa.

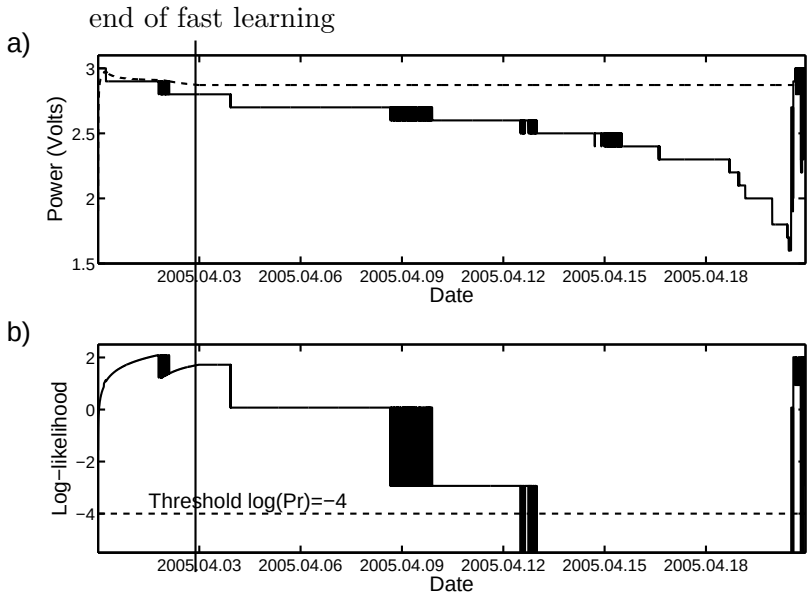


Figure 5.10: Same input data as before. The learning rate is much slower after an initial fast phase, and therefore the likelihood leaves the normal range earlier.

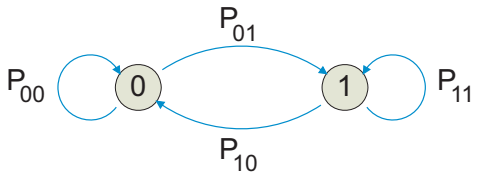


Figure 5.11: HMM of a door contact. Only the two important states are depicted, the transition probabilities P_{01} and P_{10} (open → close and close → open) will be close to 1, while their counterparts will be close to zero. Status or error messages of such a sensor can be modeled either via more states responsible for those messages or via emission probability distributions (not shown here) that allow the output of status or error messages within the two main states.

sensor. The time of occurrence or the period of time between new sensor values is often as important as the value itself. Rate models are basic models as discussed above, and used to model the lapse of time between sensor values. The rate model combined with the basic model give a much better overview of a sensor's information than a single model could. The models used for value and rate do not have to be the same, they can be any statistical model independent of each other.
The characterization of the sensor values can be further enhanced with models for the value's rate of change (its derivative, for trend analysis) and/or its second rate of change (for analyzing the trend's trend).

5.3 Scenarios

Here and in the following sections I will introduce several approaches to model scenarios in enhanced care or security systems for buildings. Such systems will have to deal with problems

where a number of subsequent scenes form the routine of the day. In the following I will refer to two examples taken from the case studies in section 4.2 and section 4.1. But first I'd like to start with an introduction on how I understand scenarios. Therefore, in figure 5.12 an example of how some operations can be subdivided into scenarios is shown[8]. In this example, a person wants to fly somewhere. I divided this operation into

- decide where to go
- book flight
- book taxi
- go to airport
- queue at check-in
- queue at security check
- proceed to gate, and finally
- queue at gate

before entering the plane. Somebody else or some automatic procedure would maybe find other scenarios within this operation. In case of the automatic system it probably would not even find a connection between these scenarios because of the temporal distance. What is important is that the result of a learning method cannot be completely predicted beforehand. The scenarios in this example are separated either by particular events, temporal distance or

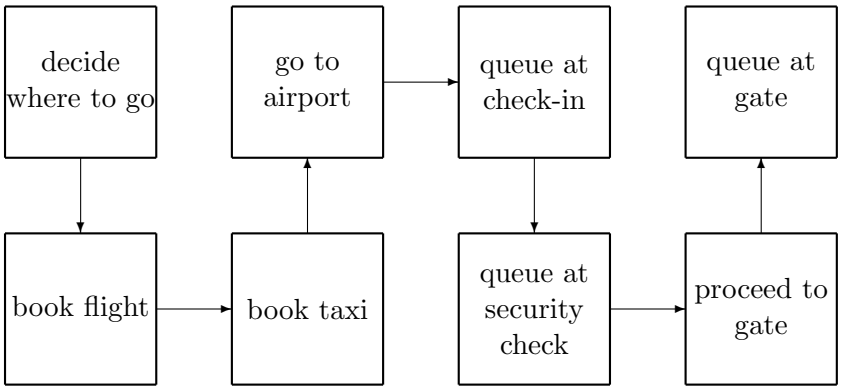


Figure 5.12: Example of scenarios: the tasks to be taken when somebody wants to fly somewhere. I did the first, intuitive subdivision of these tasks, and someone else could find different divisions. This is only one of the difficulties when dealing with scenarios.

spatial distance. This point of view gives hints for algorithms to learn about the distinctions

[8]The intention of this thesis is to propose a structure for fully learned models, therefore I use somewhat controversial examples - controversial in the sense that the constructed scenarios/states/transitions/emissions - the model elements - cannot be predicted beforehand.

between scenarios introduced in later chapters.
Another important aspect of the mentioned scenarios is the possible time span between scenarios which belong to the same operation. Related scenarios can be discontinued by scenarios, which are not concerned with the mentioned operation. Therefore, a system must be capable of dealing with a multitude of operations like those a human can perform.
But it is not within the scope of this work to deal with human operations like in the above example. On the one hand the computational effort would be far too large because of the huge number of possibilities, on the other hand the presented system is not intended to observe single persons in all aspects of their lives. Quite the opposite is the case, the system shall be installed in a building and therefore sees only small time frames out of a particular person's life. The detected scenarios and operations refer more to the life cycle - or the raison d'être - of the building than of persons.
In the following figures I want to discuss an example of a scenario in detail. The scenario is the "queue at check-in" scenario already briefly introduced in figure 5.12. It is also a relevant scenario in the SENSE project described in section 4.1.
In this scenario I assume to have position data of persons from pressure sensors in the ground, video cameras, RFID tags or some other data source. For a common security system or even for a security guard this information is worthless. In normal check-in halls there are hundreds of persons standing or moving around. A table with hundreds of data sets of position data cannot be overseen by security personnel. A useful interpretation of the data must be a layered representation of what is going on in the hall with respect to e.g.

- the behavior of the whole crowd
- the behavior of groups of persons like
  - a queue in front of a desk
  - a group of travelers
  - a chain moving from one place to another
- the behavior of single persons or
- luggage items.

All of these "regions of interest" for the security personnel can be seen as framework for scenarios. They are naturally identifiable for humans, although all have different forms of appearance. The representation of the model in the learning system therefore has to take into account that scenarios can have different forms, durations, etc. figure 5.13 shows the most abstract illustration of a scenario. The representation consists mainly of states and transitions, which are drawn in the form of circles and vectors, respectively. Each state is based on sensor values. Here, states could be footsteps of a person or pauses between footsteps. Transitions are possibilities to move from one state into another in the model. Only transitions with a probability greater than 0 are shown here. The states to the left and right are initial and final state. They are used only for convenience. The model stays in the initial state until sensor values force it to proceed into a first state. It then progresses through the model until it finishes by moving into the final state. Various scenarios can be connected using initial and final states. In the "queue at check-in" example, the initial and final states can be interpreted as entering, respectively leaving the scenario as shown in figure 5.14. *Entering* can be the movement towards the area of the desk and *leaving* the movement away from the desk after

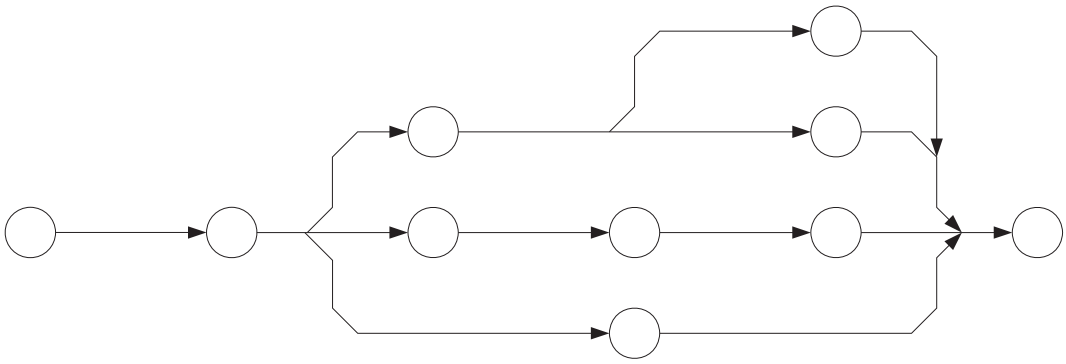


Figure 5.13: Internal representation of a scenario. The circles stand for states while the vectors stand for possible transitions between states. Left and right outer states are initial and final states, respectively. Their only use is for connecting scenarios. Each path through the diagram from initial to final state represents a particular form of the scenario.

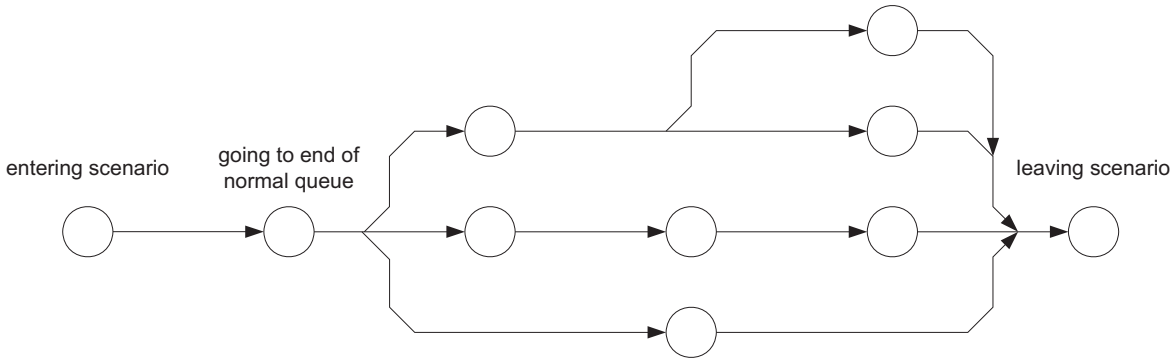


Figure 5.14: Interpretation of states. Initial and final states stand for entering and leaving the scenario. The next state to the left, which is part of all possible scenarios, could be the person's movement to the end of the queue.

being very close. Different forms of the scenario can be modeled via different paths through the model. Each path consists of a number of states with transitions between them, and can in itself again be split into subpaths. Here, in figure 5.15 I have shown a path with only one state, i.e. the path at the bottom of the diagram. In the context of the example this could be a form of the scenario where a person can move very quickly through the queuing area. This could happen when the person is very early or the flight has not enough passengers for an appreciable queue, for example. The path at the top of figure 5.16 consists of three states and 2 subpaths. It could represent scenarios where the queue in front of the desk is large enough for passengers to spend some time there waiting but the crowd does not fill the whole area. Therefore, the first state represents the fast movement to the end of the queue. The two different forms can then be seen as faster respective slower movement of the whole crowd in which the particular person is situated. Last but not least, the middle path with the largest number of states has to be scrutinized. In this example I have shown one important aspect of learning systems: humans cannot interpret everything that a machine has learned. figure 5.17 shows the hypothetical interpretation of this subform of the scenario which could

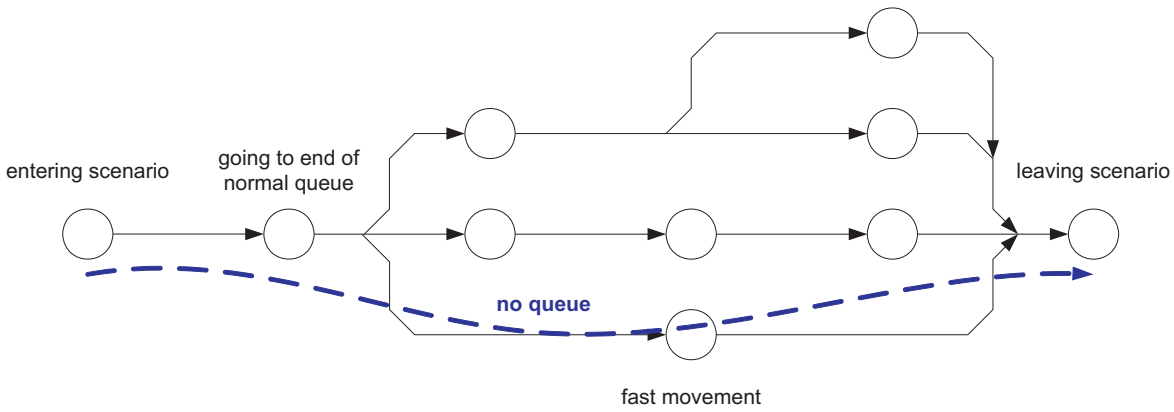


Figure 5.15: Interpretation of states. The path on bottom of the diagram could represent a situation where the person is early or there are only a few passengers for the flight and so nobody or just a few people are queuing. The state therefore could represent fast footsteps of the person. The whole scenario with entering the area, proceeding to the end of the normal queue, the fast proceeding through the area, where persons normally have to queue, and finally the leaving of the area is one particular form of the "queue at check-in" scenario.

be the normal process of queuing in the long queue with some steps followed by pauses, etc. Why the model distinguished between states in exactly this way is often dependent on the first few observations. When the model has seen more data, the number of states stays fixed, but the parameters continue to be adapted to the new information, so from the operator's point of view there may be no difference between those states. This is a major aspect that influences algorithms and learning rates and will be discussed in more detail in the next chapter.

5.4 Model structure

After having discussed the internal representation of scenarios and their interpretation I would like to introduce the first - and maybe most natural - way of letting scenario recognition become part of the building automation system: with the use of pre-defined scenarios. The idea can be briefly explained: The system operator has to define which scenarios should be detected. The scenarios must be defined regarding their sensor information and their possible sequences during a day. I will discuss the idea with the help of another example, this one taken from the SEAL project introduced in section 4.2.

5.4.1 Pre-defined model structure

Consider an example where we have a small flat for one person in a home for the elderly. In this particular house the occupants are treated like guests. They have their flats - most of them live alone in a one-room flat - where they live independently. The flats are equipped with kitchen, bathroom, toilet, a vestibule and the main room. The kitchen is located either in the vestibule or the main room. The stove has a switch that turns itself off every ten minutes. The house has several general purpose rooms, a restaurant, lobby and many others.

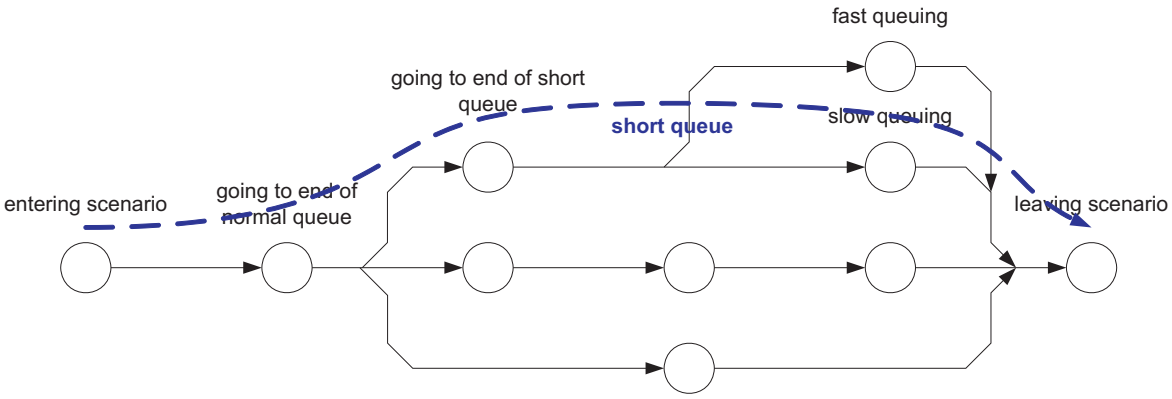


Figure 5.16: Interpretation of states. Paths within a scenario can split into different forms of one scenario as is shown here on top of the diagram. These paths could represent a scenario where some people are located at the queue, but there is still space. The first state then represents the fast proceeding to the end of the short queue. The two following possibilities could then mean fast respective slow movement of the queue in the "queue at check-in" scenario.

The occupants can get breakfast and lunch in the restaurant and dinner is served in the rooms. The occupants' attendance is noted on the one hand in the restaurant in the morning - where the people have to take a ticket with their name from a panel and put it in a box. The second check is in the afternoon when dinner is served.
Using the example of such a flat we can identify various scenes and events that a human would distinguish during a day. Some of them could be:

- sleeping
- getting up
- taking a shower
- having breakfast
- using toilette
- leaving flat
- no one in flat
- entering flat
- etc.

In a normal flat environment the user will not act in the 30-min intervals in which the aforementioned time slice models "think". Therefore, those models would experience decreasing probability when scenes happen around the temporal conjunction of the time slice models. An answer to this problem could be the overlapping of time slice models, a method that I do not want to discuss in more detail here. Instead, I will start directly with scenario recognition algorithms.

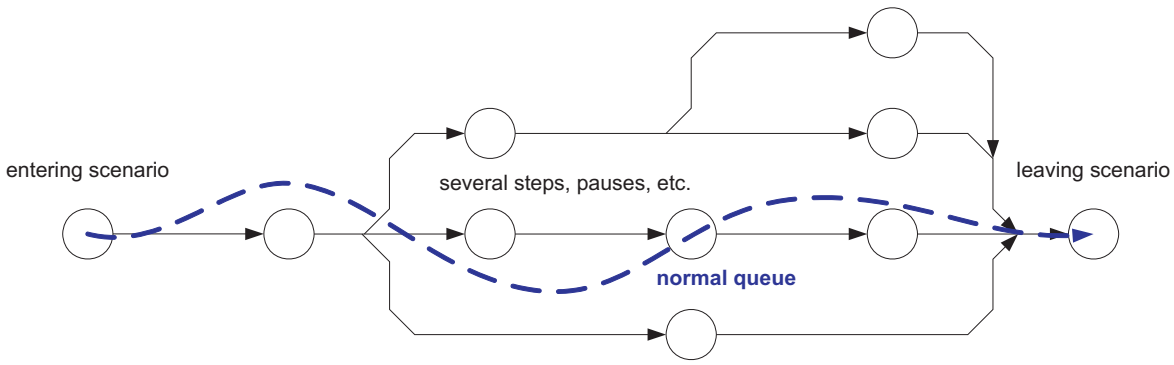


Figure 5.17: Interpretation of states. This example shows what a learned representation of a scenario could look like. The learning system always produces two kinds of states: ones that are easily interpretable, and ones that cannot be interpreted but have statistical significance. Here, the states in the middle path of the diagram could represent normal queuing where the persons make some steps, wait, go, wait, etc. Why the model distinguished that into several states is algorithm dependent, but remains unknown from the interpretation point of view.

In a system with pre-defined scenarios some scenes are mandatory, some optional, some of them happen at specific times, others have a time frame for start and duration or end. For modeling all these options, there exists a variety of possibilities. Two of them will be proposed and discussed in detail here:

1) prior probability distributions over time for scenarios

2) prior probability distributions for transitions between scenarios

An illustration of the first idea is shown in figure 5.18, where each scenario has a time dependent prior probability of occurrence. The system then has to find out, e.g. with the Viterbi algorithm (see section 3.4.2), which of the currently possible scenarios is the most probable to happen next based on the sensor values. This basic idea is independent of transition probabilities.
Some prior assumptions for the scenario recognition system in the flat are[9]:

- sleeping: from 11:00 pm till 5:00 am 80% with rising and falling edges of one hour
- toilette: from 11:00 pm till 5:00 am 10%, 5:00 am - 6:00 am 20%, 6:00 am - 11:00 am 5%, 11:00 am - 11:45 am 1%, 11:45 am - 12:00 am 75%, ...
- bath: from 5:00 am - 6:00 am 50%, 12:00 am - 12:30 am 50%, ...
- breakfast: from 6:00 am - 6:45 am 80%
- unknown: 10% all day long

[9]In this application the scenario recognition system knows beforehand when persons go to sleep and wake up and when they have lunch.

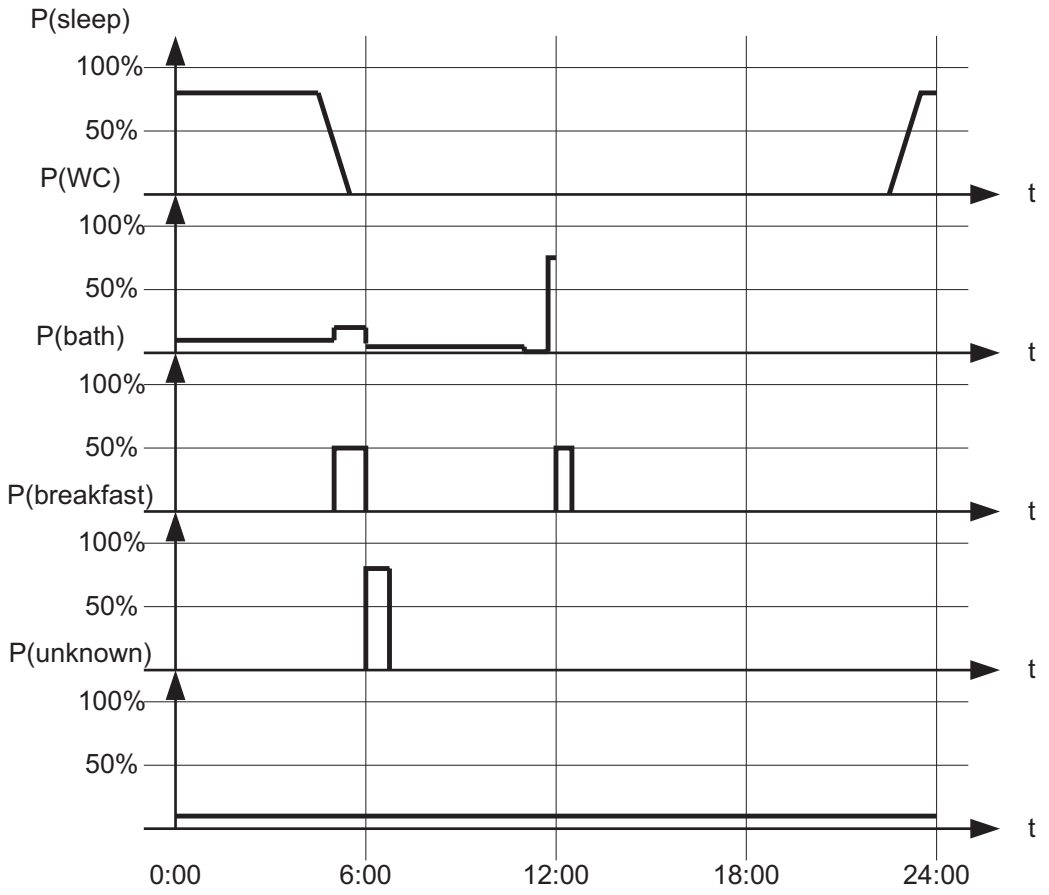


Figure 5.18: Pre-defined prior probabilities for pre-defined scenarios. Different possibilities to model the prior for the occurrence of a particular scenario within a particular time frame. The image shows the aforementioned values. The sum of all priors at any time must be 1. The unknown scenario is used to model unexpected occurences of other scenarios (at unusual times for example), and to model behavior that does not fit to one of the other scenario definitions.

In this example, the system has very good knowledge of the inhabitant's behavior. In the case of unknown behavior - which will be the normal case - the initial priors have to be much flatter. Only after a learning phase with Bayesian methods where the posterior of one day is used as a prior for the next, a useful prior distribution can be derived. The gained priors may not be as handsome as those stated above, depending on the mathematical function family. For convenience one could use conjugate priors like the Gaussian model as basis functions. In this case (see definition in section 2.1.2) the posterior distribution resulting from the utilization of Bayesian inference (see equation 2.6) is of the same mathematical form as the prior and can be used for iterative procedures as mentioned here, where the posterior of one experiment acts as the prior for the next.
The second idea for pre-defining scenarios uses a graph representation for the top level of the model, where all the scenarios and possible transitions between them are defined. In figure 5.19 a simplified illustration of this approach is given. It defines scenarios for the flat example where the occupant comes back from a meal and the three scenarios "Toilet", "Visitor" and "In bed" can be detected. This means that every combination of sensor values (except leaving)

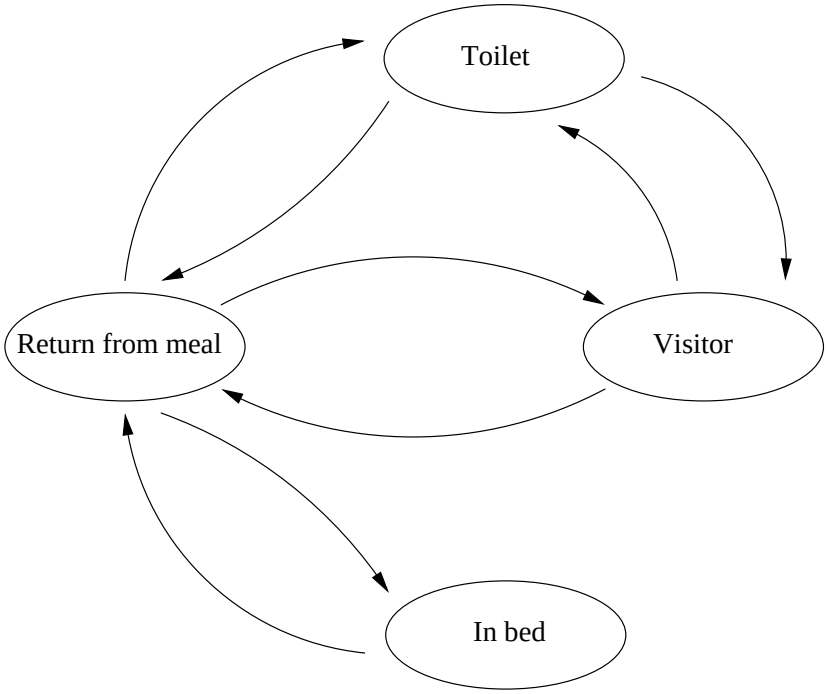


Figure 5.19: Simplified graphical representation of pre-defined scenarios. The possible scenarios in an elderly home's flat are defined. Only four scenarios can be distinguished: being in the room (which is called: "return from meal"), having a "visitor", being in "bed" or in the "toilet".

will be mapped to one - the most probable - of those states[10]. The occasion for changing the state can be defined either via directly defined sensor events like door contact, via transition probabilities in the same manner as the pre-defined prior probability distributions mentioned above or via a combination of both. So could the transition from "Return from meal" to "Visitor" be modeled simply as a door contact event in combination with motion events around the door afterwards. A door contact event without motion would be a transition to the missing "Out" state, meaning that the occupant has left the flat. A fully specified state diagram with the event-triggered transition policy is given in figure 5.20. As can be seen here, some state/transition combinations are not distinct. If the system is in "Toilet"-state and the door contact at the toilet indicates that a person is leaving the room, to which state should the system change? Solutions to this case could be:

- to remember the previous state
- to create a second "Toilet"-state, one for transitions from "Visitor", one for "Return from meal"
- to have a counter (or some other representation) for the door contact that implies a visitor

[10]If we use Viterbi (see section 3.4.2), we get the most likely state. If we use some other algorithm like the forward algorithm (see section 3.4.1) or maximum a posteriori, we get a distribution over states.

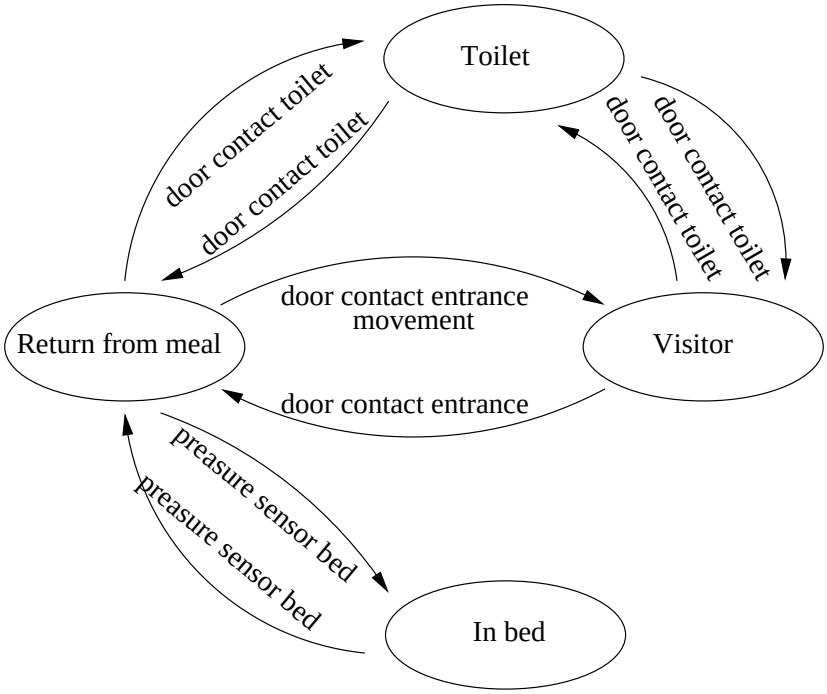


Figure 5.20: Graphical representation of pre-defined scenarios. In this case the transitions between the scenarios are pre-defined and associated with a particular sensor event.

Furthermore, the "Visitor"-state itself is not distinct. A second door contact event could mean a second guest visiting the occupant or it could mean the leaving of the single visitor. These overlapping considerations lead to the introduction of a second layer of transitions in the background, which allows the system to solve the conflict. I do not want to get into more detail since these are only two of a number of aspects of practical implementation.
The whole model of the flat example with event-driven transitions can be seen in figure 5.21. I did not include the time information - which is crucial e.g. when the system is in state "Stay in flat" and a door contact event occurs - because this would complicate the figure.
The mentioned model is a quite basic representation of daily routine in a flat in a home for the elderly, and it could only be further simplified by merging the states to the left into a single "Out"-state. On the other hand, the goal of such systems is to give as much reliable information as possible and therefore one should extend this basic model. The extensions are application-dependent, but some suggestions are:

- Duplicating the states to the right ("Stay in flat" + rest) into two or even more copies for "Morning in flat", "Afternoon", etc. This would also allow the introduction of states for eating in the flat, when the door contact indicates the delivery of the meal.
- Duplicating the "Bed"-state to differentiate between sleeping at night and during the day or after a meal. This would also open the possibility to introduce a new "Toilet"-state for nightly use.
- Introducing more distinct states for getting ready in the morning.

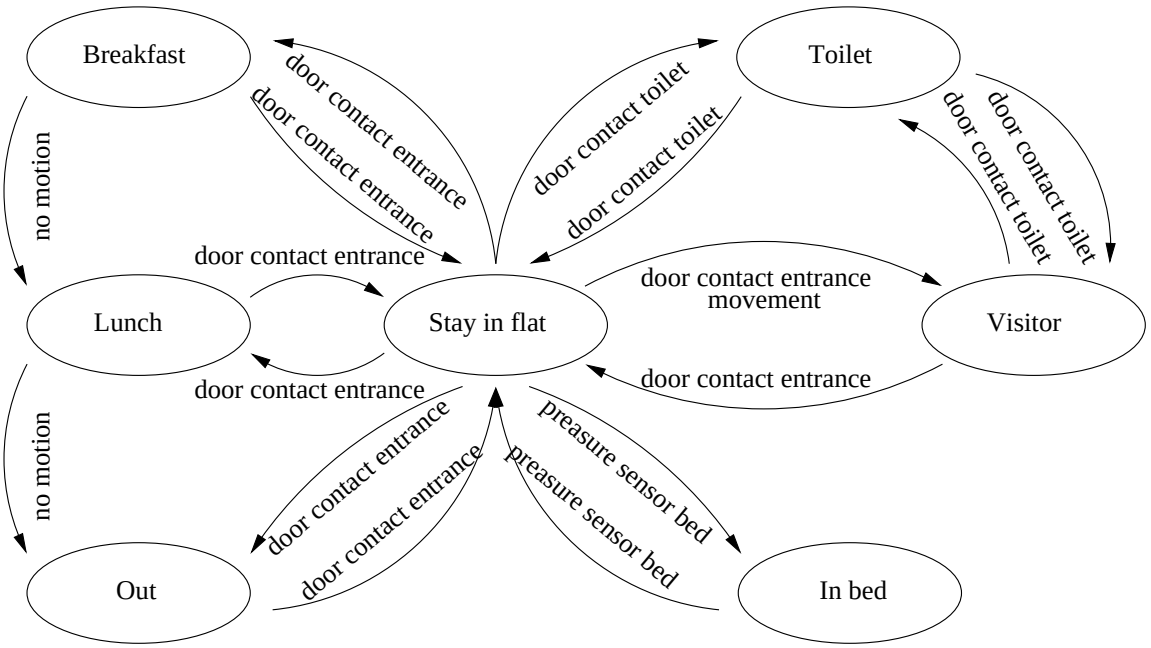


Figure 5.21: Enhanced graphical representation of pre-defined scenarios. Additional, time depended states are introduced. Following this recipe - enough knowlegde of the inhabitants presumed - an accurate model of the usage of a flat can be built.

- Introducing a second layer with states to which transitions are always possible like the "Toilet"-state or an accident or the like.
- Providing additional information like counters for going to the toilet at night, for example.
- etc.

5.4.2 Partially pre-defined and learned

Another approach for modeling scenarios with slightly more freedom is discussed in this section. The idea is to define the scenarios, but learn the corresponding sensor values in a supervised fashion. In (ORP00) a system for an office environment is proposed that is able to distinguish between five different types of scenarios plus one for unidentified behavior:

- Phone conversation
- Face-to-face conversation
- Ongoing presentation
- Distant conversation (the user is in the office and makesg conversation but is outside the field of vision of the camera)
- Nobody in the office and

- User is present and engaged in some other activity

The system uses a layered representation with a 4-state HMM in each layer. It runs on the user's PC and uses two microphones, a camera and the history of keyboard and mouse inputs as sensor information.
The authors have shown that a layered architecture with different time bases for each layer performs very well for this purpose. They have also shown that their model needs re-learning for a new user only in its lowest - closest to the sensor values - layer.

5.4.3 Fully learned and interpreted

The third approach in mining sensor data to extract meaning is fully unsupervised. "Fully unsuperviced" indicates that the model - and so the programmer of the model - has no idea about what could happen. Neither data types, data ranges nor correlations are known beforhand. If we think of a later product, this will be the normal case.
The goal is to have a system that has a set of statistical models to model the sensor values, but to have these models situated in a structure that allows the interpretation in a more abstract way. These type of scenarios - where my main focus lies - are explained in depth in the next chapter.

Chapter 6

System Structure Learning

"Aware" implies knowledge gained through one's own perception or by means of information[1].

After having presented the ideas on how to represent scenarios in a computer environment, the implementation aspects are to be discussed now. I give a detailed description of how states and scenarios of a HMM are learned. The introduction to HMMs including the algorithmic principles are in chapter 3, and the description of scenario structures can be found in chapter 5. As already stated above, the utilization of the learned scenarios can be either in a system that fully relies on unsupervised learned models or in a system that pre-defines the overall structure and fills up the graphical representation by means of learning the sensor values. The learning method stays the same and is independent of whether the structure is pre-defined or inferred. The system becomes "aware" of what is happening.

6.1 Time frame

Each sensor has its own time frame for generating values. The same is true for each physical entity that is observed by a sensor. A learning system has to take that into account and support a number of policies on how it manages its beliefs of actual physical conditions through its observations. To better describe this crucial problem I want to state data generating policies of several sensor types:

Wireless motion detector: That sensor - a wireless off-the-shelf X10[2] branded motion detector sensor - sends a data packet with value 1 and its sensor-ID - its unique identifier within the network - in case of detected motion. When the sensor permanently detects moving objects, it sends packets at a maximum speed of five seconds - which is a programmable value up to one minute. After detecting no moving object for more than 1 minute, the sensor sends a packet with value 0 and ID. The system has no possibility to detect dead links or sensors, because there is no messaging procedure for this. In a typical office environment, every motion detector sensor produces about 1000 data packets per day.

[1] definition found in several dictionaries, e.g. `www.thefreedictionary.com/aware`

[2] `www.x10.com`

Wireless ultra-low power multi sensor: It has the capability of sensing temperature, humidity and brightness and was developed at the Institute of Computer Technology (Mah04; MB04). The scope was to build an ultra-low power sensor for large multi-hop peer-to-peer wireless sensor networks. For power saving purposes the sensor switches itself off most of the time and restarts to query its sensors. It has a sensitivity for each modality and if one of them is exceeded, the sensor sends a packet with all of its sensor values including its operating voltage and connection quality to the next node that is closer to the sink node. This data transfer can only happen during small synchronous time frames. That enables the sensor to switch off its transceiver nearly all the time to save energy. The next sensors forward the packet until it reaches the sink node, which is connected to a PC. If a sensor does not sense differences exceeding its threshold for longer time periods, it sends a "keep alive" packet - with its current, unchanged values. The system can identify broken links or dead sensors by a lack of messages.

Light barrier: Depending on the manufacturer, the light barrier behaves like a switch and gives just binary values when it is activated, or it periodically gives values as long as it is activated. When equipped with a 4-20 mA interface it can detect broken links.

Microswitch: Used for door contacts, window contacts, drawers, etc. Has to be placed and mounted carefully to ensure reliable detections because of its typically small dimensions. Another important issue is to avoid bouncing either with designated hard- or software, if the manufacturer did not solve this issue.

Pressure sensor: In its simplest version it is a micro switch that is activated through reasonable weight. More intelligent pressure sensor grids that give an actual measurement of the weight with a resolution of several kilos in weight and several centimeters in space are already on their way into the labs.

Mouse and keyboard input: These modalities are useful in office applications, but both examples in chapter 4 where conducted without a PC, so I do not focus on the analysis of these data types.

Energy counters: Energy counters or water counters with digital pulse output normally give a digital pulse when a certain amount of energy is consumed. The rate of pulses gives a measurement of the total energy consumption in a flat, if there is (only) a single energy counter.

Video Camera: The use of cameras in observation systems - not just for storing or displaying to a security guard - is getting more and more popular, although their performance in an unrestricted environment is questionable[3]. They are used for detection of patterns like persons or other desired objects. In such automated systems the output of the camera system is a stream of object symbols with properties like size, position and color.

Microphone: Another modality to characterize human activities with advantages over the others is that the range of detection can be greater. Today's speech recognition for

[3]It is my deep conviction that at some time camera systems will be spread in urban environments for several intended purposes, but for now users have to be aware of the strengths and weaknesses of this technology and should not fall for buzz-words.

particular applications[4] is quite highly developed and in combination with future intelligent systems that have a comprehensive representation of some human concepts, the recognition rate of words, phrases and sentences will rise to a level that allows conversation.

Digital Retina Sensor: A sensor developed by scientists at Seibersdorf research (LPD; LBD$^+$), which detects motion in a gray-scaled picture in hardware and delivers on- and off-pixels for changes in the picture's contrast. Moving objects can be detected as the area inside an on- and an off-pixel line. In contrast to common motion detection systems, the retina outputs pixel-information as it notices a difference. Therefore, the frame-rate of a camera is changed into a pixel-rate for the retina.

All these different policies in data generation force a learning system to either receive input about the nature and policy of the data and its generation or to make assumptions. The statistical models presented in section 5.2 have different expectations about the data, so a mapping has to be provided. I will discuss an example with data generated by a wireless motion detector and I am using a rounded 30-minutes average, which means that either "more" or "less" movement has occurred during that time[5].

6.2 System Structure Learning Principles

Finding patterns in sensor data that can be used to create states heavily depends on the type of sensory data and the characteristics of the data's generation. I will use a very simple data base consisting of binary motion detector data to illustrate its principles later on.
The way from sensor data to a scenario representation starts with selecting a data base consisting of a sensor's value chains. Ideally, those value chains should have their first value immediately after the start of the scenario and their end should coincide with the end of the scenario. The particular number of values within the chain is less important - but, certainly, also part of the information that later forms the scenario. Much more important than the length of the chains is their number, which form the dataset for learning the scenario and will be discussed later in detail.
After having obtained a dataset, the value chains are compared. The idea is that similar parts of the chain at similar places - which should correlate with the time - should be combined with a state. Especially at the beginning and the end of the chain I expect a scenario to have quite similar sensor values. Therefore, I compare the values of one chain at the beginning of the scenario with the first values of another chain to find a common progress within the two chains. When the two chains possess differences after the first few values[6], I assume a change in behavior of the observed object, and therefore the scenario representation splits into subscenarios. If the sensor values are binary, a binary tree after the "initial state" is built with this procedure (see figure 6.1). The same assumption as for the start of a scenario

[4] With the term "speech recognition for particular applications" I am trying to highlight that today's speech recognition systems need a very narrow focus on a particular application - a vocabulary. A system that is capable of "talking" to people in a way a stranger - someone who does not share our vernacular or trade jargon - would, requires far more knowledge about the human mind. This is because language is just a projection of sophisticated mental processes into a highly compressed data channel, where even the type of compression and encryption has to be negotiated.

[5] This may sound to be very inaccurate, but for the purpose of subdividing daily routines in reasonable states - or finding discrepancies during normal operation - this is sufficient.

[6] In fact, even their first value can differ.

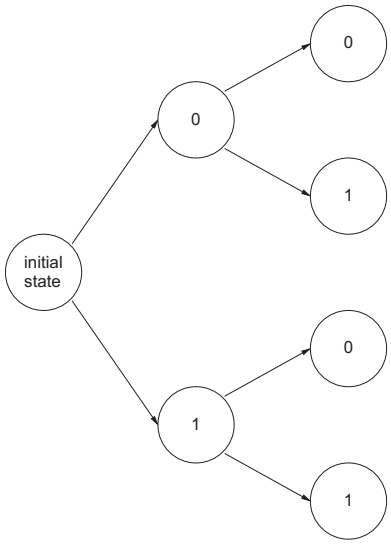
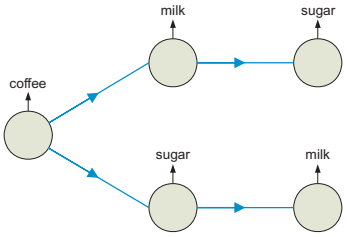


Figure 6.1: Binary tree built by the "comparing of the state's beginning and end" procedure. Sensor values are assumed to be from a binary sensor like a motion detector. The numbers "1" and "0" in the circles represent the sensor value of that intermediate state.

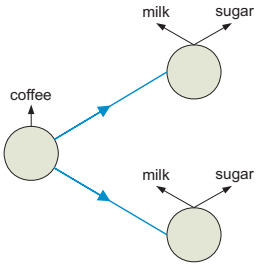
- that a scenario has several forms, but all have similar starts - is made for its end. The chains are compared from the "final state" backwards as long as they have similarities.
After this first step of merging, the scenario consists of a *two-sided tree* which starts on the one side from the "initial" and on the other side from the "final state" and is connected via chains of values of arbitrary length, whereby every value creates a state.
The second and third part of the algorithm work with these states. They assume consecutive states of the same sensor value belong together. Therefore, they are merged into a single state. A motion detector for example gives similar sensor values as long as there is motion. In an office this could be a meeting, in a lobby a group of colleagues going for lunch, etc. So, the second part of the algorithm just merges states that are consecutive with the same sensor values and with a transition probability of 100%.
Finally, the third part of the algorithm allows scenarios to vary in their order of events. If somebody makes a cup of coffee, it is of no relevance if he puts first coffee, then milk and then sugar in the cup or if this is done in a different order (see Figure 6.2 for illustration). The algorithm takes this into account by merging consecutive states with transitions of 100%. This policy decreases the probability for - the supposed two - events each by 50%, but it allows the order to be changed to prevent the system from learning two different sub-scenarios for similar things. For a better illustration, the three steps are characterized in detail in the following section.

6.3 System Structure Learning Example

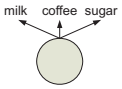
The aforementioned principles are now discussed with a particular example: I present a system that learns semantic symbols from a binary motion detector sensor. That sensor - as described before - sends a data packet with value 1 in case of detected motion. When



two sub-scenarios after step 2 of the algorithm



the same two sub-scenarios after step 3



possible collective state after second execution of (a slightly modified) step 1 and then step 3

Figure 6.2: The third step of state-merging merges consecutive states with transitions of 100% in between. This allows scenarios to vary their events in order without being modeled as different sub-scenarios. If these states are part of a (larger) scenario, the situation after them is always the same: a person puts milk, coffee and sugar in a cup. The third figure shows a state that could "emerge" after step 1 and 3 of the algorithm would be repeated. But during the repetition of step 1, state chains - not value chains - need to be merged. Afterwards, step 3 would find a 100% transition between the coffee and the remaining (milk, sugar) state, and would merge them.

the sensor permanently detects moving objects, it sends packets at a maximum speed of five seconds. After detecting no moving object for more than 1 minute, the sensor sends a packet with value 0. The system is not directly supported with the motion detector's sensor values, but with averaged sensor values. I divided the 24 hours of a day into 48 time slots, each 30 minutes long[7]. In those time slots the mean of the sensor values is computed and rounded. If no value is available during 30 minutes, I set the mean to 0 which is synonymic to "no motion". The chains of 48 values are then fed into the (empty) model and during a procedure of the following three steps the structure of the model is learned (see also (BSR06)):

1) Comparison of the chain's beginning/end

2) Merging of identical states

3) Merging of consecutive states

These steps in combination with the averaging of the sensor values shall produce HMMs with a manageable number of states. The number of states of HMMs is a compromise between generalization (low number of states, the model is applicable for a wide range of different scenarios, but not able to distinguish between particular ones) and specialization (rather high number of states, not every possible scenario is depicted in the model and quite similar scenarios can have different paths). [8]

[7] The interval length of 30 min was chosen, because it seems to be natural to distinguish in a daily routine roughly 30 min intervals. A second consideration is the length of the value chains: in case of shorter intervals the chains become longer and a higher amount of states is created. The interpretation thereof may be harder.

[8] (SO93) have shown that model merging of very specific models always ends up with better or equal results than specializing very general models. They also introduced a method for merging models and maximizing the posterior probability of the model. The starting point for model merging is to find an application-specific way to reduce the number of states dramatically, because the computational effort for their proposed best-first model merging is relatively high. In our application we waive the best-first model merging because the below stated application-specific ways work fine with motion detection sensors.

6.3.1 Comparison of Chain's Borders

As mentioned above, the algorithm starts with a procedure, during which the sensor values at the start and end of the chain (see figure 6.3) are compared. From here on onwards, I will use

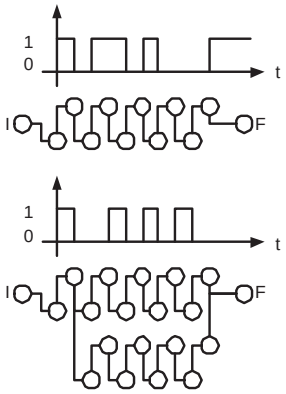


Figure 6.3: Comparison of chain borders. The top sequence of sensor values is mapped into a chain of states with appropriate emissions and transitions with 100% from state to state. The next sequence of sensor values is compared to the earlier emissions and in case of different values the model splits and introduces a new path. This is done in forward and backward direction.

the term emissions - in style of the outputs of HMMs - for the sensor values. Each sensor value creates a state. Each state has a weight that is used for computing transition probabilities in case of merging, an emission probability distribution (in case of the binary sensor this is just a simple binomial distribution), and a transition probability distribution(which is a multinomial one). In case of sensors with a continuous rang of values, Gaussian or even mixture of Gaussians pdfs could be used.
As long as the new values match the emissions of the states at the beginning of the already seen chains[9], we just update the weight of that state in the model. This procedure is accomplished for both the first states in the model and the first new values (see pseudocode 6.1) and the last states of the model and the new last values (see pseudocode 6.2). The remaining new values in between form a new state chain, which will be located between the two tree ends of the model (see pseudocode 6.3). A possible result of this procedure is shown in figure 6.4.

6.3.2 Merging of Identical States

In case of binary sensors or sensors with discrete sensor values, it can happen that a sensor emits a sequence of identical values, so that chains of identical states appear. For modeling the events behind system behavior we can assume that either nothing (status messages) or the same event happens when a sensor continuously emits the same value. The motion detector in our application sends every 5 seconds a new value "1" when it sees objects moving. For scenario detection purposes it makes no difference if that motion takes 15 times 5 seconds or only 11 times 5 seconds, but in the model these two situations would appear as different ones.

[9]This algorithm is greedy, which means it follows the first path it finds with matching emissions, although there may be another path where more emissions could be merged. In the case of empty model learning, this behavior is of no relevance, but if new chains are incorporated after executing the other steps of the algorithm, it may be of advantage to use the Viterbi algorithm (as presented in section 3.4.2) here.

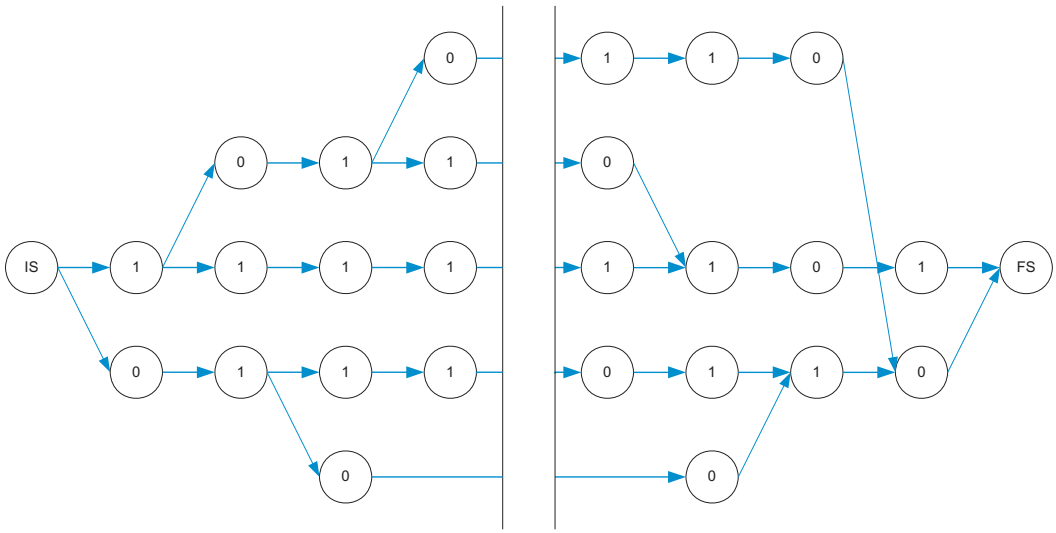


Figure 6.4: A possible result of the first part of the algorithm. The outer states (IS and FS) depict initial and final state. Each of them spans a (binary) tree as described in figure 6.1. Each sensor value created its own state with weigth=1, emission=sensor value (indicated with the numbers inside the circles) and a transition to the consecutive state. Repeatedly used states have appropriate higher weigths and non-unity transitions. The dashed lines imply an intermittence, because typical state chains for thirty minutes contain more than fifty states.

This consideration leads to the idea of merging identical states. Thereby, chains of states with 100% transitions from one to the consecutive next state and identical emissions are sought. These states are then merged into a single one as shown in figure 6.5 and described in the pseudocode 6.4. The "self-transition" is computed as $T_{ii} = N/(N+1)$, N being the number of states merged. It is important to note that this way of creating a new state produces a geometric duration probability distribution. If a different duration probability distribution is required, HSMMs (see section 3.5) must be used - or another policy of computing the duration pdf in that state has to be applied.

6.3.3 Merging of Consecutive States

In the model's learning phase a model with initial state, final state and a number of paths in between is created. Each of the paths has the potential to be a scenario and each state has the potential to represent a semantic concept. Each splitting in a path is interpreted as a change in system behavior. During model-merging, each of these paths has to prove its value for the model. We assume that scenarios can vary their order. This means that values in a chain can change their place with other values that happen often at the same time, as illustrated in the coffee example in Figure 6.2.
These considerations lead to the third procedure: states with transitions of 100% are merged into a single state. So, if - after the merging of identical states - chains of states with unity transitions remain, then those chains are merged (see figure 6.6 and pseudocode 6.5 and 6.6 - which shows the "Otherwise" part of the "Switch" that did not fit into the same page).

Algorithm 6.1: MergeBegin()

```
Input: a non-empty new value chain
k ... current state
k_o ... last state
i ... emission index
len =length(new chain)
i = 0
k = 0
k_o = 1, em = false
while i < len ∧ (k ≠ k_o ∨ (Transition_{k,k} ≠ 0 ∧ em == true)) do
    k_o = k
    forall other states j do
        if state j has suitable emissions then
            em = true
        else em = false
        if Transitions_{k,j} > 0 ∧ em == true then
            update Transition pdf's
            update Emission pdf's
            update weights
            k = j
    i = i + 1
if i == len then
    // sample already covered by existing states
    update Transition pdf's // to ''final state''
    update Emission pdf's
    update weights
    exit // sample completely covered, no new states created
k_fr = k // save index of last state and matching data point
i_fr = i
```

Pseudocode 6.1 shows how new value chains are incorporated into the model. Therefore, variable k (index of current state), k_o (index of last state) and i (index of emission symbol in the observation sequence) are introduced. Additionally, *em* is necessary, which stores the result of the check, whether or not there is a possibility to emit the next output symbol from the state under investigation.

The variables are initialized and a loop is started. As long as there are further output symbols in the chain, and either the current state differs from the last or there is a self-transition and the emission is still valid, the loop is executed. This loop is executed for each output symbol.

In the output symbol loop all states are visited. Each state is checked if its emissions are suitable for the output symbol and the result is stored in *em*. Finally, if there exists a transition from the old state to the envisaged state and *em* is true: the symbol can be represented within the model!

The transition is updated according to the new weight of the state. Also, all other transitions from the current state have to be updated (to ensure summing up to unity). The emissions need updates, too, depending on their model. In the case of multinomial distributions all emissions need sequentially an update similar to the transitions. Finally, the weight is increased, the index updated and the next symbol is processed.

When the loop terminates, some of the symbols of the new chain are incorporated into the model. It has to be checked whether the chain is fully incorporated. If so, a transition to the final

state is introduced and the rest of the current state's transitions are updated.

Algorithm 6.2: MergeEnd()

```
i =length(new chain)
len = 0
k_o = 0
k = last state
while i > len ∧ (k ≠ k_o ∧ i > i_fr ∨ (Transition_{k,k} ≠ 0 ∧ em == true))
do
    k_o = k
    forall other states j do
        // in reverse order
        if state j has suitable emissions then
            em = true
        else
            em = false
        if Transitions_{j,k} > 0 ∧ em == true then
            update Transition pdf's
            update Emission pdf's
            update weights
            k = j
    i = i − 1
if i == i_fr then
    // sample completely found in model
    update Transition pdf's // to intermediate state k
    update Emission pdf's
    update weights
    exit // sample completely included, no new states created
k_ba = k // save index of last state and matching data point
i_ba = i + 1
```

Pseudocode 6.2 shows how new value chains are incorporated into the model from their back. The procedure is similar to incorporating chains from the beginning, except the variables have to be initialized differently. Also, when checking whether the sample is fully incorporated, the index of the last output symbol incorporated from the other side has to be considered.

Algorithm 6.3: ConnectChains()

```
length(states) =length(states) + (i_ba − i_fr)
set Transitions between consecutive new states to 100%
set Emission pdf's
set weights
∀k update Transitions_{k_fr,k}// ''connect'' existing front part with
    new chain
k = end of new chain
update Transitions_{k,k_ba}
```

Pseudocode 6.3 depicts the creation of the intermediate state chain between the two tree ends of the model.

Algorithm 6.4: MergeIdenticalStates()

```
k ... current state
k_o ... old state
lt ... length(states)
repeat
    k_o = k
    while Emissions_k == Emissions_{k+1} ∧
          Transitions_{k,k+1} == 1 ∧
          weight_k == weight_{k+1} do
        inc(k)
    nr = k − k_o
    if nr < 1 then exit
    // here we have a chain with at least 2 identical consecutive
       states
    forall l do Transitions_{k_o,l} = Transitions_{k,l}/(nr + 1)
    for l = k_o + 1 to k do weight_{k_o} = weight_{k_o} + weight_l
    for l = k_o + 1 to lt − nr − 1 do
        // update indices of all remaining states with
           Transitions, Emission and weights
        weight_l = weight_{l+nr}
        Emissions_l = Emissions_{l+nr}
        forall m do Transitions_{m,l} = Transitions_{m,l+nr}
    for l = k_o + 1 to lt − nr − 1 do
        forall m do Transitions_{l,m} = Transitions_{l+nr,m}
    Transitions_{k_o,k_o} = nr/(nr + 1)
until end of states
```

Pseudocode 6.4 shows how identical consecutive states with unity transitions are merged. Therefore, variable k (index of current state) and k_o (index of last state) are introduced and the global variable lt (the number of states) is also considered.

A loop is executed until all possible state chains are seen. After this, another loop over all subsequent states is started. In that loop all states k from k_o on are visited. Each state is checked if its emissions equal the emissions of the consecutive state[a] and if there exists a unity transition from k to $k+1$. Finally, if the states possess the same weight, they belong to a chain of - at least two - identical (see footnote a) states found for merging.

The following commands are used to delete the states in the chain ($k_o + 1$ to k) and to compute the transitions of the new "super state" as well as its weight. The transitions of state k_o are computed to be the transitions from state k scaled by the new weight. The transitions to state k_o need no update. Finally, the weight is increased. I added the single weights of the states in the chain instead of multiplying the weight with the number of states because later generations of the algorithm may also merge state chains with slightly varying weights.

In case the algorithm is implemented having an array of states, all subsequent states have to be moved nr positions to the front.

When this algorithm terminates, identical states are merged into "super states" as I call them because of their large weight.

[a] equality in a general sense. In case of non-multivariate pdf's a measure of equality and a threshold have to be used.

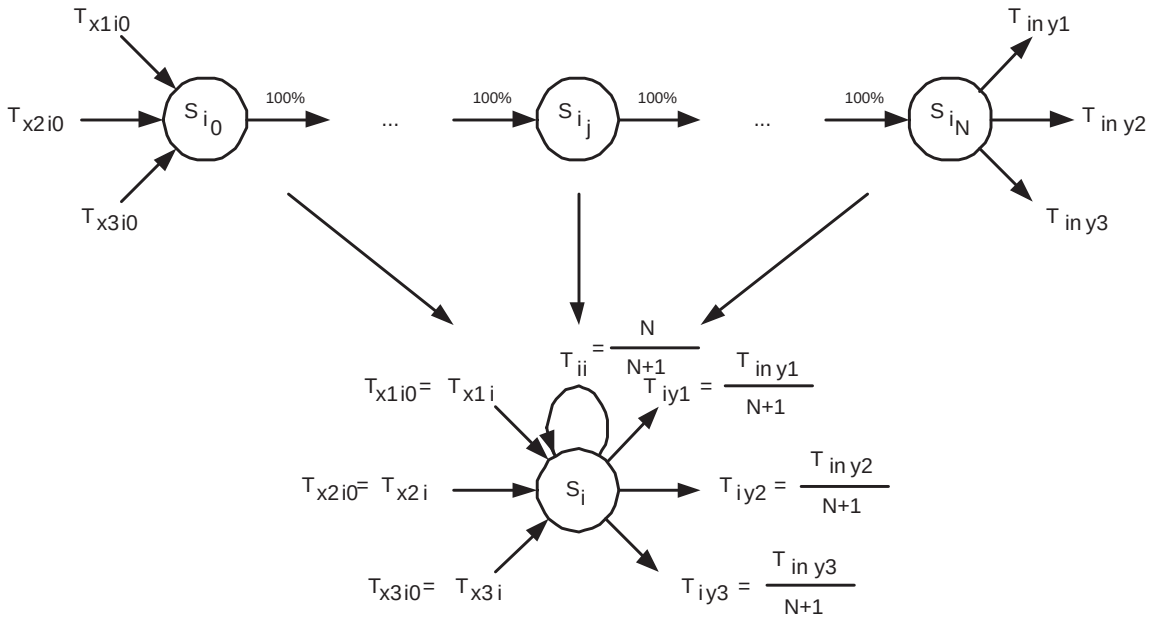


Figure 6.5: Merging of a state chain part with N identical states into one single state i. The original chain has unity transitions, x's and y's being other states in the HMM.

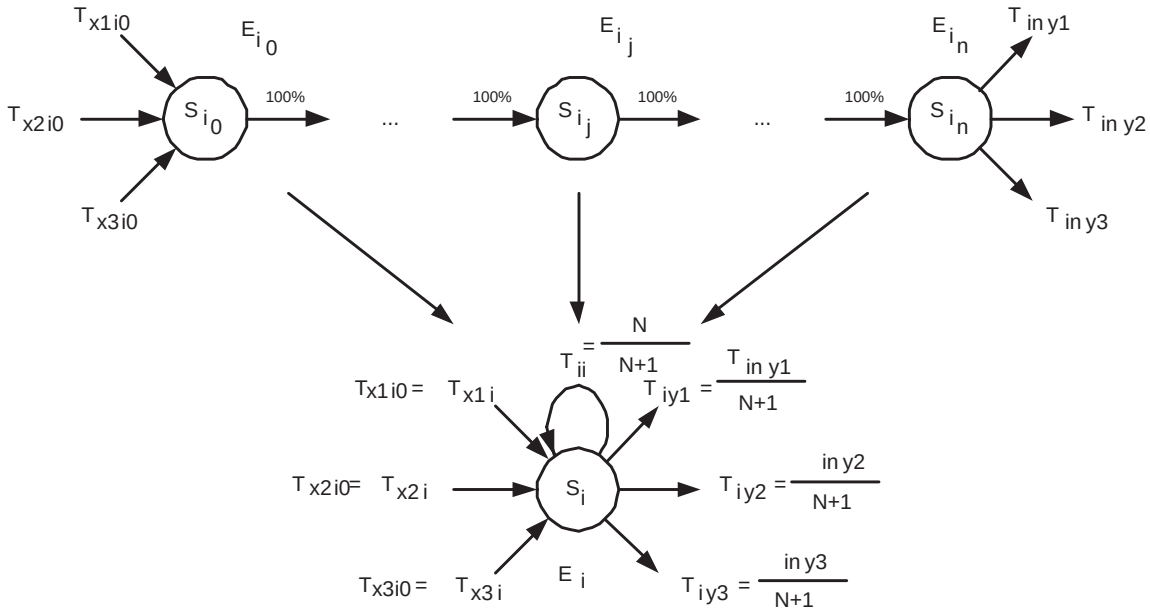


Figure 6.6: Merging of a state chain part with states with unity transitions in between into one single state i. The new Emissions E_i are computed as Emissions of original states times state's weight and normalized to sum to unity.

Algorithm 6.5: MergeConsecutiveStates_1()

```
k ... current state, k_o = 0 ... old state, lt ... length(states),
st ... array of states
repeat
    k = k_o,  n = −1,  b = true
    while b == true do
        inc(n),  b = false
        forall l do
            if Transitions_{k,l} == 1 then
                forall m do if Transitions_{m,l} ≠ 0 then break
                b = true,  st_n = k,  k = l
                break
    switch n do
        case 0 : break
        case 1 :
            if k ≠ lt then
                // merge k_o with its successor k
                forall l do
                    if l ≠ k then
                        Transitions_{k_o,l} =
                        Transitions_{k_o,l} + Transitions_{k,l} weight_k / (weight_k + weight_{k_o})
                Transitions_{k_o,k_o} = (Transitions_{k,k} weight_k + weight_{k_o}) / (weight_k + weight_{k_o})
                Transitions_{k_o,k} = 0
                forall l do
                    Transitions_{l,k_o} = Transitions_{l,k_o} + Transitions_{l,k}
                update Emissions according to weights
                weight_{k_o} = weight_{k_o} + weight_k
                for l = k to lt − 1 do
                    // update indices of all remaining states with
                       Transitions, Emission and weights
                    weight_l = weight_{l+1}
                    Emissions_l = Emissions_{l+1}
                    forall m do Transitions_{m,l} = Transitions_{m,l+1}
                    for l = k to lt − 1 do
                        forall l do Transitions_{l,m} = Transitions_{l+1,m}
                    dec(k_o), dec(lt)
        case otherwise : see 6.6
    inc(k_o)
until end of states
```

Pseudocode 6.5 shows how non-identical, consecutive states with unity transitions are merged. Therefore, variable k (index of current state) and k_o (index of last state) are introduced and the global variable lt (the number of states) is also considered. Furthermore, st (an array for storing indices of states in a chain) is also introduced.
A loop is executed until all possible state chains are sought. Therefore, in a twice-nested subloop all possible successors l from k on are visited. Each state is checked if there exists a unity transition from k to l.

If so, and if there exists no other transition to this state, its index is stored in *st* and further successor are sought. This complicated procedure is necessary because it is no longer guaranteed in this phase of the algorithm that consecutive states with unity transitions have subsequent indices. Now the algorithm splits into one part for a chain of just two states - where the utilization of *st* is not necessary - and one part for merging longer chains, which is depicted in algorithm 6.6.
The merging procedure is similar to that in algorithm 6.4, except that the emissions have to be merged, too. In the case of multivariate emission pdfs, the procedure is the same as for the transitions. In other cases, appropriate methods have to be utilized.

6.3.4 Software Implementation

All of the algorithms described in this and the following two chapters are implemented in Borland[10] Delphi 7. Here I want to give a very brief overview of the program:
The Main Class `ModellerMainCode` consists of a front end (a GUI that displays sensor values and likelihoods and allows the user to monitor model parameters), database and socket connections (to the various sources of sensor data), and a `ModelList`. The idea of the `ModelList` is as follows: each `Model` in the list is passed the sensor values and each `Model` knows for itself, if it is interested in the data. Therefore each `Model` has to register `DataInterests`, that are mainly capable of limiting the number of sensors and narrowing the time frame a `Model` is interested in. Each `Model` can register an arbitrary number of `DataInterests`, the result is the union of all those interests.
Each new sensor value is encapsulated into the structure `DataValue` and passed to the `ModelList`. `DataValue` contains information about the data type, the actual value and the timestamp. The `ModelList` has the information about the `DataInterests` of the `Models`. Each appropriately registered `Model` will get the new `DataValue` and can incorporate it.
`Models` are for example the `GaussianModel`, `HistogramModel`, `AverageRateModel`, and `MixtureModel` (all of them used for the BASE system described in chapter 7), `HMM_Binary_Emissions` (used for the example described in this Chapter), `HMM_Multinomial_Emissions` and `HMM_Mixture_Emissions` (used in chapter 8). The architecture of the program is illustrated in figure 6.7.

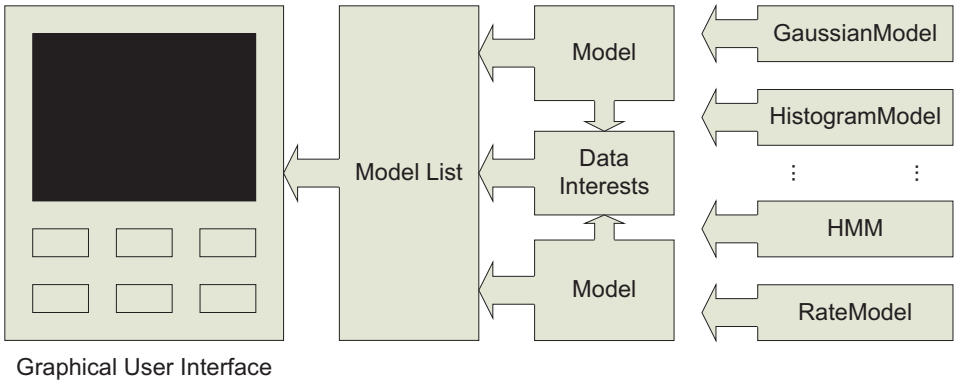


Figure 6.7: Architectre of the Delphi program.

[10] `www.borland.com`

Algorithm 6.6: MergeConsecutiveStates_2()

```
⋮
see 6.5
⋮
switch n do
    case 0 : break
    case 1 : see 6.5
    otherwise
        for l = 1 to n − 1 do
            // merging
            forall k do
                if k ≠ st_l then
                    Transitions_{st_0,k} =
                    Transitions_{st_0,k} + Transitions_{st_l,k} weight_{st_l} / (weight_{st_0} + weight_{st_l})
            Transitions_{st_0,st_0} = weight_{st_0} / (weight_{st_0} + weight_{st_l})
            Transitions_{st_0,st_l} = 0
            forall k do
                Transitions_{k,st_0} = Transitions_{k,st_0} + Transitions_{k,st_l}
            update Emissions according to weights
            weight_{st_0} = weight_{st_0} + weight_{st_l}
            for k = st_l to lt − 2 do
                // update indices of all remaining states with
                   Transitions, Emission and weights
                weight_k = weight_{k+1}
                Emissions_k = Emissions_{k+1}
                forall m do Transitions_{m,k} = Transitions_{m,k+1}
                for k = st_l to lt − 2 do
                    forall m do Transitions_{k,m} = Transitions_{k+1,m}
                dec(lt)
                for m = l + 1 to n − 1 do st_m = st_{m−1}
⋮
see 6.5
⋮
```

Pseudocode 6.6 shows the second part of merging non-identical consecutive states with unity transitions. The difference is that the index of the next state to be merged has to be recovered from the array st and - because the chain's states do not possess consecutive indices - the deleting of the state and moving of the other states has to be repeated $length(st)$ times.

6.4 System Structure Interpretation

Finally, the model is constructed. In the running example motion detector sensor values from a building automation system in an office environment were merged to set up a model of the

behavior inside the office. Now it is time to develop tools for analyzing the structure of the model.

6.4.1 Visualization

The model consists of a HMM as a framework, transition probability distributions being multinomial probability distributions and emissions being binomial distributions. To be able to interpret the structure of the model, it is necessary to visualize it. During my work it turned out to be a difficult task to automatically draw a meaningful picture of the model. Therefore, I give the algorithms for drawing the graph here.
I introduced a data structure consisting of an object *state*, which possesses an ID and two arrays of pointers of states, one for *Children*, the second for *Parents*. The states together with these links form the tree structure of the model. This structure is of advantage to recursively process the tree. The creation of the structure is described in the pseudocodes 6.7 - 6.9.

Algorithm 6.7: CreateTree()

```
Input: Transitions
if empty Transitions ∨ final state then exit
forall indices i do
    if Transitions_{this.ID,i} ≠ 0 then
        if this.ID ≠ i then
            p = getRoot().FindChild(i)
            if empty p then
                Create new state i
                Add state i to this.Children
                Add this to i.Parents
            else
                if p already in this.Children then exit
                else
                    Add p to this.Children
                    Add this to i.Parents
        // selftransition will be used for later drawing
        else
            this.selftransition = Transitions_{i,i}
forall Children i do
    i.CreateTree(t)
```

Pseudocode 6.7 describes the recursive creation of the tree structure. Therefore, the list of non-zero transitions is needed. Each state attempts to create for its successors a new state and include them into its "Children"-list while being added to its successors' "Parents"-list. This structure is a kind of double-linked list with the extension of more links per entity. To avoid creating the same state twice or more often by different predecessors, *getRoot* followed by $findChild(i)$ is executed. *getRoot* steps back to the initial state and $findChild(i)$ searches the tree for an already created instance of state i.

Algorithm 6.8: FindChild()

```
Input: ID ... ID of state to find
if ID == this.ID then return this
forall Children i do
    p = i.FindChild(ID)
    if p not empty then return p
return empty pointer
```

Pseudocode 6.8 describes the recursive search procedure for a hypothetical and already created instance of state i. Therefore, each state is asked whether it is the sought state. If not, all - if any - successors are asked. If yes, the exit condition is fulfilled and a pointer to self is returned.

Algorithm 6.9: GetRoot()

```
if this.ID = 0 then return this
forall Parents i do
    p = i.getRoot()
    if p not empty then return p
return empty pointer
```

Pseudocode 6.9 describes the recursive search procedure for finding the root node. Therefore, each state is asked whether it is the sought state. If not, all - if any - predecessors are asked. If yes, the exit condition is fulfilled and a pointer to self is returned.

After the tree is created, the next step is to find out its size in terms of maximum length and width in states. The idea is to have a mental grid below the drawing space. Each state can be drawn on one of the crossings. The procedures given in pseudocode 6.10 and 6.11 are responsible for this part.

Algorithm 6.10: GetWidth()

```
if length(this.Children)== 0 then return 1
forall Children i do
    this.WayWidths_i = i.getWidth()
w = 0
forall Children i do
    w = w + this.WayWidths_i
return w
```

Pseudocode 6.10 describes the recursive search procedure for computing the width of the tree for later drawing. It also calculates the *WayWidths* of each state to know the width of the subtrees. Therefore, each state is asked its width - 1 for the "final state" - and it answers by adding the widths of its successors.

Algorithm 6.11: SetWayLength()

```
Input: Transitions
if empty Transitions then exit
forall Children i do
    i.setWayLength(Transitions)
    k = 0
    forall other states j do
        if i.WayLengths_j > k then
            k = i.WayLengths_j
    this.WayLengths_{i.ID} = k + 1
```

Pseudocode 6.11 describes the recursive search procedure for finding the length of paths through the tree. It also calculates the *WayLengths* of each state to know the length of the subtrees. Therefore, each state is asked its waylengths to the "final state". Each state answers with an array of the lengths of the longest ways of its "Children" + 1.

These two procedures also fill the arrays for WayLengths and WayWidths in each state of the model. With the information of the size of the tree a 2D-array is created to store the position of the states. Finally, all the information for drawing the tree is available.
The procedure for drawing the tree is based on following consideration:

- draw the longest path in the middle
- the shorter the path - the less states it has - the further away from the middle it shall be drawn
- before a path is placed ensure the width is available

For finding out the placement of the transitions it is necessary to always proceed in two steps. First, start from the first state in a path back to the last and set all the positions during this step. Then start drawing the state there, step backwards, and, finally, draw the transition. During the first process, the positions of the children are unknown. The complete description of the drawing of the model is given in pseudocodes 6.12 - 6.14 and subsequent verbal descriptions.

Algorithm 6.12: ComputeOrder()

```
k = 1,  l = 0
forall ordered Children j do
    forall State indices i do
        m = this.Children_{order_{j-1}}.ID
        if j == 0 ∧ this.Waylengths_i ≥ k ∨
           j > 0 ∧ this.Waylengths_i ≥ k ∧ this.Waylengths_i ≤
        this.Waylengths_m then
            if j > 0 ∧ i == m then continue
            k = this.Waylengths_k   l = i
        forall Children i do
            if i.ID == l then order_j = i
    k = 1
```

Pseudocode 6.12 sorts the successors. The criterion is their Waylength.

Algorithm 6.13: ComputeWheretoDraw()

```
a = findClosestChild(this)
for j = this.ix to a do
    k = 0
    forall l in width(grid) do
        if b_{j,l} unused then
            if k == 0 then inc(space_{0,j})
            else inc(space_{1,j})
        else
            space_{1,j} = 0
            k = 1
k = min(space_0)
l = min(space_1)
// decide where to draw:  overhead or underneath?
if k + abs(this.iy − height(grid) − l) < l + abs(this.iy − (k − 1)) then
    j = height(grid) − l
else  j = k − 1
```

Pseudocode 6.13 computes the logical y-location for the successor.

Algorithm 6.14: FindClosestChild(Parent)

```
Input: Parent...one of the predecessors
if Parent.drawn − this.drawn == −1 ∨ final state then return ix
ret = MAXINT
forall Children i do
    k = i.FindClosestChild(self)
    if k < ret then ret = k
return ret
```

Pseudocode 6.14: recursive procedure for finding the nearest already drawn successor. The result is the logical x-position of that state on the grid.

Algorithm 6.15: DrawTree()

Input: $Parent \ldots$ pointer to predecessor
Input: $x, y \ldots$ position to draw the state
Input: $dx, dy \ldots$ space between neighbors
Input: $ix, iy \ldots$ logical position in the grid
Input: $b \ldots$ the grid
$inc(drawn)$
if *Parent.drawn - this.drawn == -1* **then** $dec(this.drawn)$ **exit**
if *final state* **then**
  draw a circle at (x, y)
  write the ID of the state in the circle
  $b_{ix,iy} = this.ID$
  exit
$Compute\_Order$
forall *ordered Children* i **do**
  if *first Child* **then**
    $this.Children_{order_0}.drawTree(this, x + dx, y, dx, dy, ix + 1, iy, b)$
    $b_{ix,iy} = this.ID$
    if $y == this.Children_{order_0}.y$ **then**
    $drawline(x + 10, y, this.Children_{order_0}.x - 10, y)$
    else
      $drawline(x + 10, y, this.Children_{order_0}.x - (dx - 10), y)$
      $drawlineto(this.Children_{order_0}.x - 10, this.Children_{order_0}.y)$
      for $k = ix + 1$ **to** $this.Children_{order_0}.ix - 1$ **do** $b_{k,iy} = this.ID$
  else
    $j = Compute\_Where\_to\_Draw$
    $this.Children_{order_i}.drawTree(self, x + dx, y + dy * (j -$
    $iy), dx, dy, ix + 1, j, b)$
    if $this.Children_{order_i}.iy \neq j$ **then**
      $drawlinefrom(Children_{order_i}.x - 10, this.Children_{order_i}.y)$
      $drawlineto(this.Children_{order_i}.x - (dx - 10), y + dy * (j - iy))$
      $drawlineto(x + (dx - 10), y + dy * (j - iy))$
      $drawlineto(x + 10, y)$
      for $k = ix + 1$ **to** $a - 1$ **do** $b_{k,j} = this.ID$
    else
      $drawlinefrom(Children_{order_i}.x - 10, this.Children_{order_i}.y)$
      $drawlineto(x + (dx - 10), this.Children_{order_i}.y)$
      $drawlineto(x + 10, y)$
draw a circle at (x, y)
write the ID of the state in the circle
if $this.selftransition > 0$ **then** draw an arc above the circle

Pseudocode 6.15 describes the recursive procedure for drawing the tree. The tree is drawn starting with the "final state", splitting into the states that have transitions to the "final state" and so forth. The tree is fanned out from the back in an attempt to have the longest paths in the middle of the illustration. Several variables are necessary:

Parent a pointer to the state's predecessor

x, y the position to draw the state on the underlying canvas in pixels

dx, dy space between states on the canvas in pixels

ix, iy position in states on the grid

b the grid

drawn indicates if the state has already been drawn

order a sorted array of children; the criterion is the length of the way to the final state

space a 2D array of size $2 \times max(WayLength)$; it is used to compute free ways overhead und underneath the current state to draw a transition to a successor

The procedure starts with incrementing *drawn* and comparing it with *Parent.drawn* to ensure that the same subpath is not drawn again.
If the state is the final state, a circle and the ID are drawn and the recursion is left. Otherwise, the successors are ordered regarding their length of ways to the "final state".
All successors are visited in a loop.
The recursion is continued with the first one - with most states to the end. It is intended to draw the state one logical position to the right and with the same height. If the state has not already been drawn, exactly this will happen, so the transition can be drawn as a horizontal line. If not, two lines have to be drawn, one horizontal one close to the child and one skew line closing the connection.
If there are more children, a strategy for placement is necessary: For each child, I search for the immediately following and already drawn state on its way to the final state. In the worst case, this is the final state itself, or otherwise the envisaged path merges somewhere closer with another path. From the current position to this state a path has to be drawn. Therefore, I need to search in the grid for free horizontal lines between these x-positions. This is done overhead and underneath. The place with most free space is chosen.
When the placement is clear, the recursion can be continued with the new logical coordinates. When the recursion comes back, transitions can be drawn. If the successor is drawn on the foreseen place, only two lines are necessary, one from the current logical y-position to the intended, and a second - horizontal - for closing the connection. If the state has already been drawn somewhere, three lines are necessary. One to the intended y-position, a second horizontal one close to the child, and finally another skew line to the actual y-position of the child.
The last remaining action is to draw the current state.

6.4.2 Model Interpretation

In the running example, one motion detector sensor emits approximately 1000 values per day. Those were averaged during 30-min periods. If no value was emitted, 0 - being the value for "no motion" - was assumed. The obtained 48 "sensor values" were then fed into the model. As described in section 6.3.1, for the first 15 days state chains with tree-ends are created. Afterwards, the two types of merging were applied. figure 6.8 shows the result. In this example, no sensor value was emitted on weekends. In these cases I decided to input a chain of 48 "0"s. Those weekend days are represented by the state at the bottom. For interpreting the model structure a good picture of the model is the most important tool. But the presented algorithms are just able to give an overview of how the model looks like, not what the components of the model mean. Therefore, it is necessary to have the possibility to draw another picture, that shows which path an actual chain of values takes through the model .
The Viterbi algorithm presented in section 3.4.2 is used to find out the most probable path

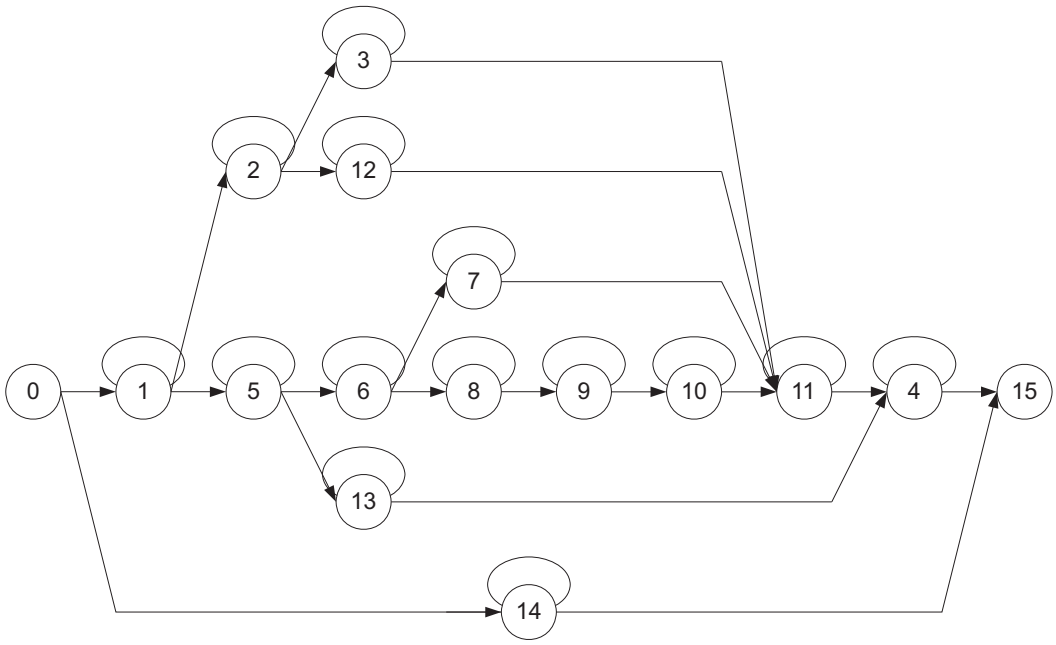


Figure 6.8: The model. States are labeled with numbers, 0 being the initial state and 15 the final one. The initial and final states appear at the start and end of every sensor value chain. The ellipses above the states show non-zero self-transitions. Ellipses and lines represent transitions with a probability greater than 0.

that generated a particular output sequence. So, for interpretation purposes, a chain of sensor values, its Viterbi path and some graphical representation are sought. The algorithms described in Pseudocode 6.17 and 6.16 are responsible for this part.

In this model every path through the model represents a particular daily routine. In this context we can talk of a semantic concept for a whole day. But, moreover, some of the states themselves also represent particular - and by humans identifiable - parts of a daily routine. In this model, all paths but one go through state 1 and end in state 4. The only exception is the transition from initial to final state with state 14 in between, which represents the weekends (and has a transition probability of 28.6%, which is 2/7). Along with the following figures of particular daily routines, state 1 can be interpreted as the morning until the first motion is detected and state 4 represents the evening after everybody already left the office (i.e. no more motion is detected). figure 6.10 shows a normal day in the "observed" office. One comment concerning the "sensor values": In this office the cleaning person comes every working day in the morning to empty the wastebasket. We can see that state 5 covers a short motion followed by a longer "break" with no motion, temporally located in the morning. This state thus represents the cleaning person. Finally, state 13 represents the period of constant activity during the day.

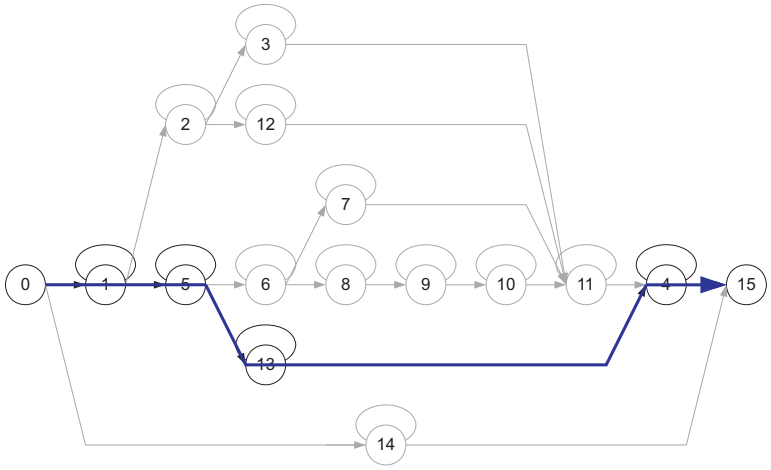


Figure 6.9: A path through the model. For a particular chain of sensor values, the Viterbi algorithm finds the most probable path. The path shown here together with its sensor values is shown in figure 6.10.

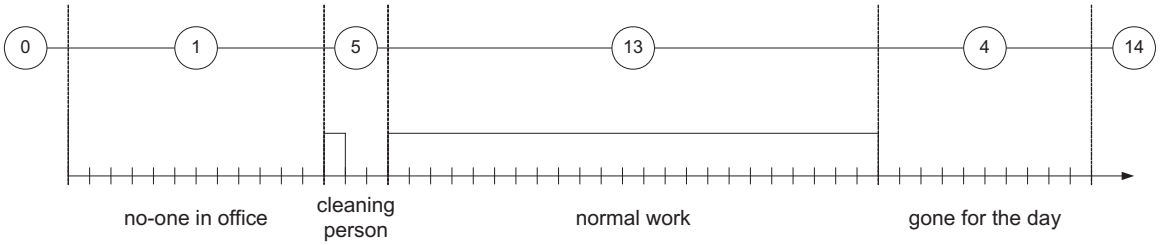


Figure 6.10: A normal day in the office. The figure shows the Viterbi path through the model and the 48 averaged sensor values for that day. Dotted lines mark changes in states.

Algorithm 6.16: DrawPath()

```
Input: VPath... Viterbi Path
Input: DSet... Sensor value chain
if empty DSet then exit
draw circle "initial state"
draw circle "final state"
draw coordinate axes
j = VPath_0
forall Path States in VPath do
    if j ≠ VPath_i then
        draw line from border to last state
        draw circle, label it
        draw line to border to next state
        draw dotted border lines
        j = VPath_i
forall Sensor Values in DSet do
    draw value and difference
```

Pseudocode 6.16 shows how to visualize a sensor value chain and its corresponding Viterbi path.

The model also saw days with discontinuous appearances of motion. An example is given in figure 6.12, where the afternoon is divided into time frames with and without motion. Each of these time frames are represented by their own states. The model assumes that there is an underlying cause for the change in behavior. In a later parameter update phase there are at least 2 possibilities for those states and the transitions to them: Either those values were singularities and the probability thereof sinks, or afternoons like this happen more often and maybe persons from that office can interpret these states as their weekly project meeting or the like.

The interpretation of states 1, 5 and 4 is as it was with the first day, state 6 represents a

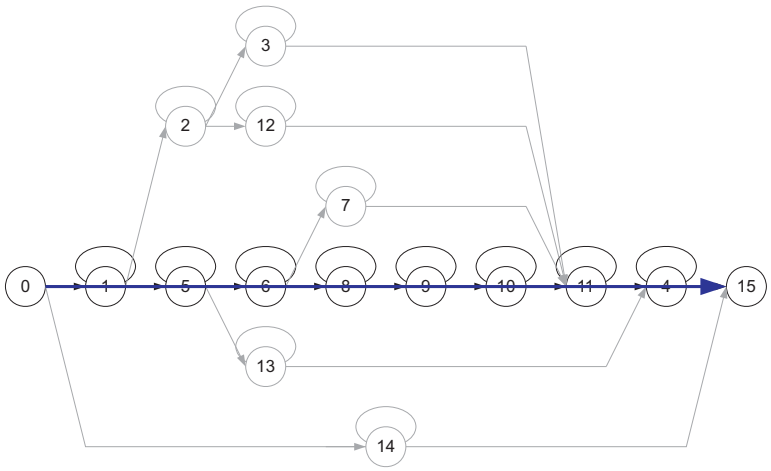


Figure 6.11: A path through the model. For a particular chain of sensor values, the Viterbi algorithm finds the most probable path. The path shown here together with its sensor values is shown in figure 6.12.

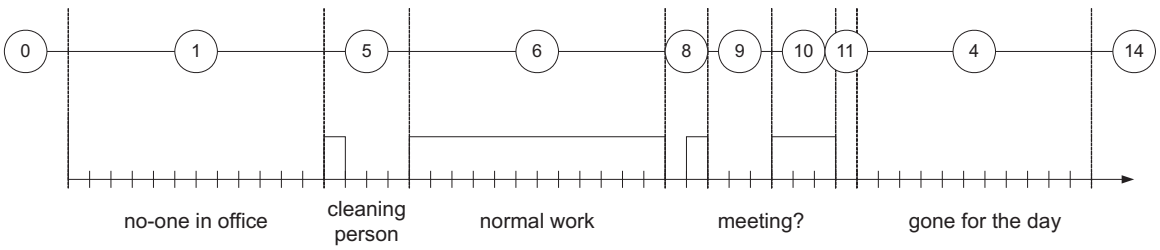


Figure 6.12: A day with breaks in activity in the afternoon. Maybe a meeting?

"shorter" working day and states 8, 9, 10 and 11 cannot be interpreted by us because of lack of particular knowledge, but could represent lunch and meetings.

Algorithm 6.17: ViterbiPath()

Input: dset; `// the data sample, array of float`
Output: VPath; `// array of states`
if *empty dset* **then exit**
`// probability of being in state` k `and emitting the first symbol`
forall k **do**
  if $dset_0 \subset Emissions_k$ **then**
    $vit1_k = Transitions_{0,k} \cdot P(dset_0 \mid Emissions_k)$
  else $vit1_k = 0$
forall k **do**
  if $vit1_k \neq 0$ **then** $Viterbi_{0,k} == 0$ **else** $Viterbi_{0,k} = -1$
`// probability of going from state` k `to state` l `and emitting the` j^{th} `symbol`
forall j **do**
  forall k **do**
    forall l **do**
      if $dset_j \subset Emissions_l$ **then**
        $vit2_{k,l} = vit1_k \cdot Transitions_{k,l} \cdot P(dset_j \mid Emissions_l)$
      else $vit2_{k,l} = 0$
  forall l **do**
    $vi = \max_k(vit2_{k,l})$
    $m = k \mid vi$
    $Viterbi_{j,l} = m$
    $vit1_l = vi$
forall k **do**
  if $Transitions_{k,last\ state} == 0$ **then** $vit1_k = 0$
`//` $vit1_k$ `is the probability (including the whole path) of transmitting from state` k `to the ''final state''. The highest value is taken as starting point for backtracking.`
$vi = \max_k(vit1_k)$
$m = k \mid vi$
if $vi == 0$ **then return** *empty Path*
forall j **do**
  `// reverse order`
  $Result_j = m$
  $m = Viterbi_{j,m}$

Pseudocode 6.17 shows the implementation of the Viterbi algorithm.

Chapter 7

Case Study: Statistical detection of Alarm Conditions in BAS

As indicated in chapter 2, I implemented a system for the automatic detection of abnormal behavior in a building automation system. Additionally, I compared it to a standard system for problem detection, using the same data set. The automated method is based on statistical models of sensor behavior. A model of normal behavior is automatically constructed. Model parameters are optimized using an on-line maximum-likelihood algorithm (see section 2.2.1). Incoming sensor values are then compared to the model, and an alarm is generated when the sensor value has a low probability under the model. The alarms generated by the automated system are compared to alarms generated by pre-defined rules in a standard automation system. Finally, the performance, strengths and weaknesses of the automated detection system are discussed.
The techniques utilized in this case study are the SGMs introduced in section 5.2 in combination with the learning mechanisms presented in chapter 2. The semantic interpretation of learned symbols was not the scope of this particular work.

7.1 Why automatic systems?

The whole thesis deals with enhancing standard - manually initialized and monitored - building automation systems. I just want to state here again their major drawbacks. Or, in other words, I want to demonstrate, where a further step of automation can help saving time and effort and increasing safety, security and user comfort.
Automation systems have seen widespread deployment in modern buildings, and include systems for environmental control, energy management, safety, security, access control, and remote monitoring. As the cost of automation systems falls, and the technology converges towards standardized protocols, we can expect automation to move from the office into the home. It will also encompass not just building management technology, but also entertainment, kitchen appliances and communications devices.

Today's building sensor and control systems are primarily based on the processing of sensor information using predefined rules. The user or operator defines, for example, the range of valid temperatures for a room by a rule – when the temperature value in that room is out

of range (e.g. caused by a defect), the system reacts (for example, with an error message). More complicated diagnostics require an experienced operator who can observe and interpret real-time sensor values. However, as systems become larger, are deployed in a wider variety of environments, and are targeted at technically less-sophisticated users, both possibilities (rule-based systems and expert users) become more and more uneconomic. The control system would require comprehensive prior knowledge of possible operating conditions, ranges of values and error conditions. This knowledge may not be readily available, and will be difficult for an unsophisticated user to input. It is impractical for experienced operators to directly observe large systems, and inexperienced users can not interpret sensor values.
In this case study I will present a system that automatically recognizes error conditions specific to a given sensor, actuator or system without the need of pre-programmed error conditions, user-entered parameters, or experienced operators. The system observes sensor and actuator data over a period of time, constructs a model of "normality", and issues error alerts when sensor or actuator values vary from normal. The result is a system that can recognize sensor errors or abnormal sensor or actuator readings, with a minimal manual configuration of the system. Further, if sensor readings vary or drift over time, the system can automatically adapt itself to the new "normal" conditions, adjusting its error criteria accordingly.

The system, called BASE (Building Automation system for Safety and Energy efficiency) (SBR05; SBR06), was tested in a building automation system consisting of 248 sensors spread across four systems (a heating and three ventilation systems). The data was collected over a time period of several months. This work presents the results of this trial, highlights the strengths and weaknesses of the automated system, and suggests future areas of improvement.

The sensor data included forced air temperatures, room temperatures, air pressures, the status of control valves, and so on. The data was collected over a period of five months (mid December to mid May), thus including the seasonal transition from winter to summer.
The BASE system was allowed to adapt models to each of the 248 sensors. Each sensor model consisted of 12 mixtures of Gaussians models, one model for every two hours of a 24-hour day. During this time period, the alarm messages from the standard building automation system were also recorded. Because of the relatively simple nature of the individual models, I was able to simultaneously fit a large number of models in real time. In this case, in any particular 2-hour period, 248 sensor models were being fit simultaneously, and the system optimized $248 \times 12 = 2976$ sensor models in total.
In the following sections I describe and show results of model parameter optimization, optimized models, and compare alarms delivered by BASE and by the traditional building automation system.

7.2 Parameter Optimization

The model log-likelihood, given by equation 2.12, is a measure of model quality. As the parameter values are optimized on-line, the log-likelihood of the sensor models increases. figure 7.1 shows the average log-likelihood during parameter optimization of the sensor models in the test system. The log-likelihood increases with time, indicating that the models improve over time. The log-likelihood does not increase consistently, however, due to the on-line fitting of parameters and simultaneous reporting of abnormal sensor values. If sensors receive

abnormal values, the log-likelihood decreases, until the values return to their normal range or the sensor model adapts to the new range of values.

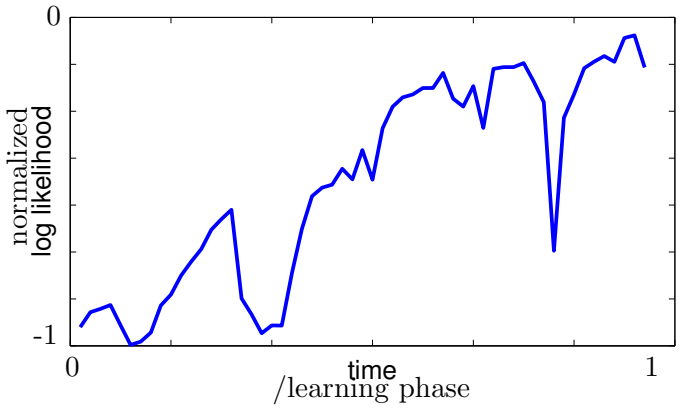


Figure 7.1: Average learning curve of sensor models. On average the log-likelihood of the model improves over time. The three large drops in average log-likelihood correspond to large modifications of the system (addition of equipment, changes in system parameters). Smaller drops correspond to system-wide disturbances (such as a power outage). The learning phase is application dependent and varies typically from hours to weeks.

Fig. 7.2 shows an example of a single sensor after the learning phase. The upper figure shows the sensor value, and the lower shows the corresponding log-likelihood as a function of time. The large disturbance in the center (a power fluctuation) registers an alarm. So do the two small "spikes" near the end of the graph.

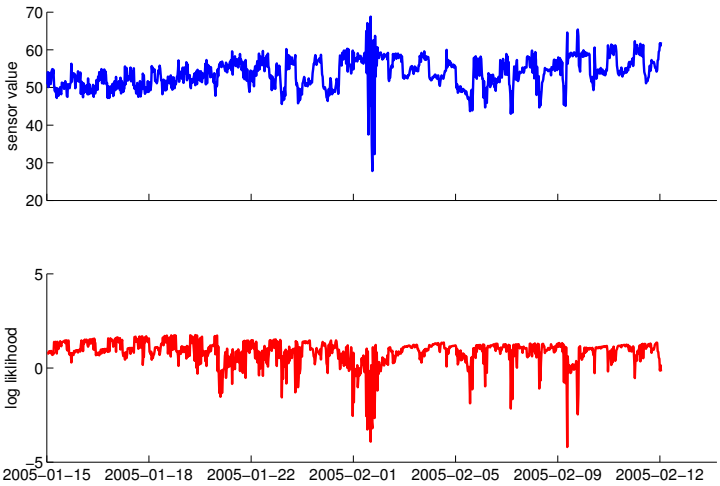


Figure 7.2: Sensor value and log-likelihood of a single sensor from the system. Unusual sensor values register as drops in the log-likelihood, causing alarms.

7.3 Comparison to Standard System

During the test period, each sensor model's log-likelihood was computed for each new sensor value. This log-likelihood was compared to a threshold. If the log-likelihood fell under

threshold, an alarm was emitted. These alarms were then analyzed to discover if they corresponded to alarms emitted by the standard system. These were provided to us by the system operators.

Over the 5-month test period, the traditional system emitted 2521 alarms (excluding the lowest priority informational messages). The number of alarms delivered by BASE depends on the alarm threshold selected by the user. The number of BASE alarms ranged from 198 at the lowest alarm threshold to 1599 at the highest tested alarm threshold.

The BASE system, or any automatic system for alarm detection, can deliver "false" alarms, that is, alarms that are considered unimportant by the operator. This can occur if the sensor value makes a statistically significant but in reality unimportant deviation from normality, or if the sensor model does not sufficiently capture the true variation in sensor values. With the standard system, alarms are by definition not "false", because they occur in exactly defined situations. However, alarms of standard systems can also be "false" in the sense that they are considered unimportant by the operator. In order to compare the BASE and standard alarms, an equivalent to "false" alarms must be defined for the standard building automation system. In this work, an alarm from the standard system is considered unimportant if it is canceled by the operator within five minutes of its occurrence. I call these alarms "quick" alarms. I also label BASE alarms that are considered unimportant by the operator "quick" alarms.

Concerning BASE alarms, a subset of approximately 10% of alarms was classified by hand into "quick" or "normal" alarms. In order to be compared to the standard alarms, a subset of approximately 10% of the standard alarms were also classified "quick" and "normal", using timing information from the alarm log. In both cases, the subsampled alarms were uniformly and randomly selected, and it was assumed that the subsample of classified alarms were representative of the set of all alarms.

Fig. 7.3 shows an analysis of alarms emitted by both systems. The bar graph shows clusters of bars, one cluster for each of 9 different alarm thresholds. The error bars indicate confidence values, taking the subsampling of hand-analyzed alarms into account.

As the threshold increases, the number of alarms delivered by BASE increases. After a threshold value of -4.2, the number of BASE alarms does not increase any more, indicating that the data from the automation system is either classified by BASE as quite unlikely (probability less than 0.05) or likely (probability greater than 0.15) with little in between. This suggests that the BASE models describe the observed data well. Also, the number of BASE alarms which correspond to alarms from the standard system, the number of unique good BASE alarms, and the number of BASE false alarms all increase as the threshold increases.

In general, the standard system delivers a surprising number of alarms that are quickly dismissed, much more so than BASE. This may simply be because of greater familiarity with the standard system, so the operator can quickly decide between alarms that they often see and new, problematic situations. The number of BASE false alarms is low in comparison, indicating that the BASE system does deliver useful information. The large but not complete overlap between BASE and standards alarms suggest that true alarms can be sorted into three categories: Alarms where a value crosses a pre-defined threshold, without deviating from historical values (for example, an alarm indicating that a fuel tank is nearly empty); alarms that deviate from normality but do not cross the threshold (see the alarms in figure

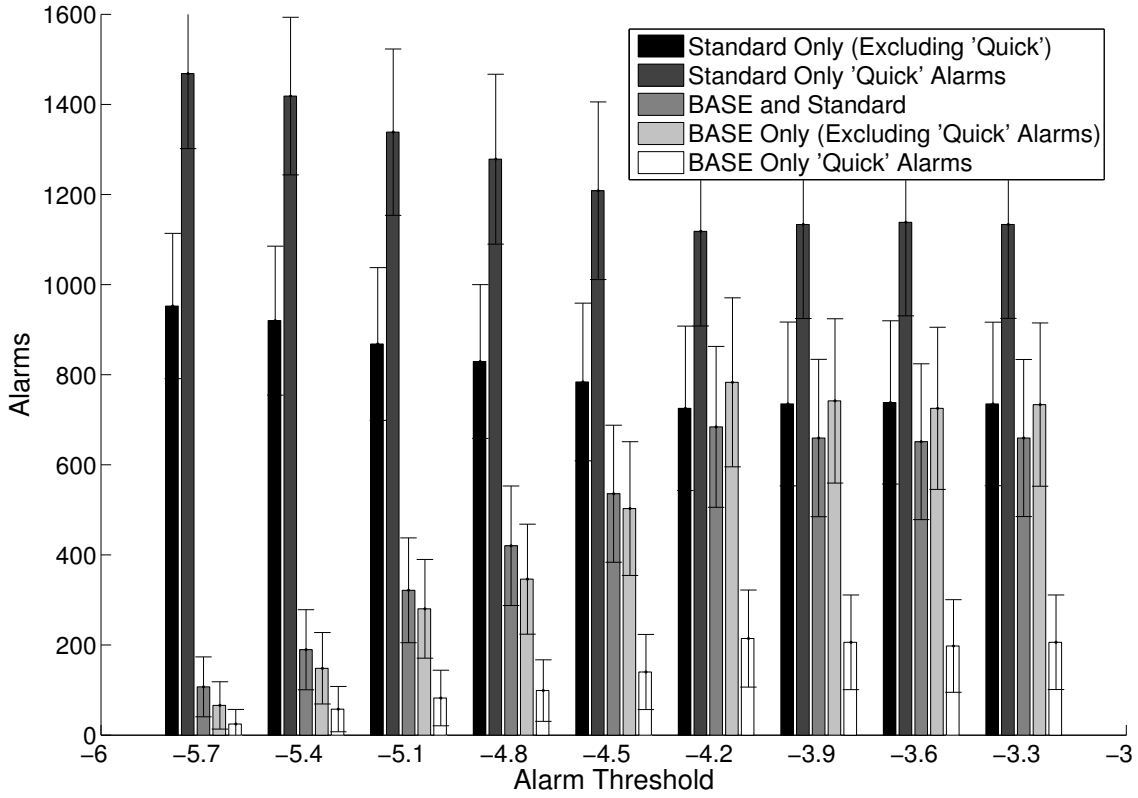


Figure 7.3: Analysis of BASE alarms and alarms from the Standard building automation system. Both systems deliver alarms that are non-critical (those labeled "Quick" alarms) as well as alarms that are considered important by the user. As the threshold increases, the number of good alarms, but also the number of false alarms delivered by BASE increases.

7.2 or figure 7.4 for example); and alarms that do both. The BASE and standard systems largely complement one another and give the operator additional information which they would not otherwise have.

7.4 User Comfort

One case in which BASE can deliver useful information is to enhance user comfort. In the test environment, users receive heated air from the central heating system, and are then able to increase air temperature with an additional local system. It is therefore difficult to set global thresholds indicating a local heating problem. The BASE system detects local problems quickly, since each sensor model is adapted to the local environment. figure 7.4 shows an example of local problem detection. The upper figure shows outside temperature, and the lower figure shows local room temperature. The dark stripes indicate normal working hours, and the number indicates average temperature over a 12-hour period. On the third day the indoor temperature drops to an abnormal (and uncomfortable) 18°C, triggering a BASE alarm (vertical line). Without this alarm, the occupants of the room must alert the building managers themselves.

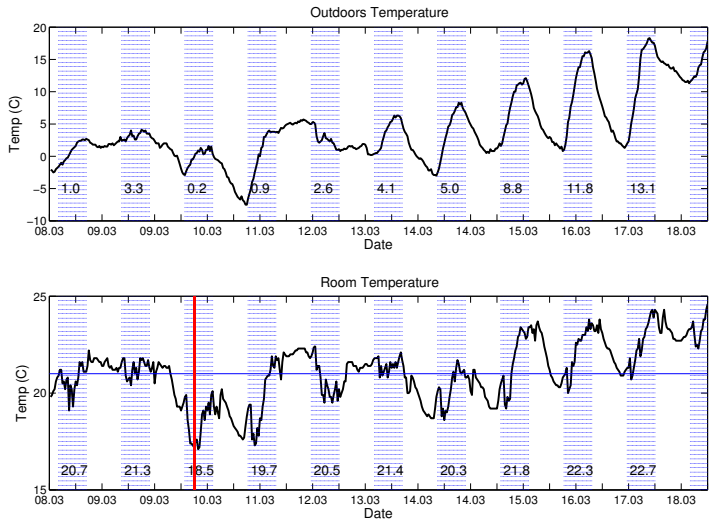


Figure 7.4: An example of detecting a change in local conditions. On the third day (third dark stripe) average indoor temperature drops, triggering an alarm.

7.5 System Adaptation

The BASE system adapts model parameters to changes in the environment. That is, when there are repeated examples of an abnormal situation, BASE will adapt the sensor models in such a way that the new situation is no longer considered abnormal. figure 7.5 shows an example of this behavior. When the temperature in a room is abnormally high (on date 11.02), it is detected by the system and an alarm is delivered (vertical line). However, the next time this temperature is reached (on 14.02), the system has already adapted to this circumstance, and does not deliver an alarm. How many repetitions of a situation are required before an alarm is no longer generated depends on the system's learning rate, and how unusual the situation is. More unusual situations require more repetitions before the system adapts.

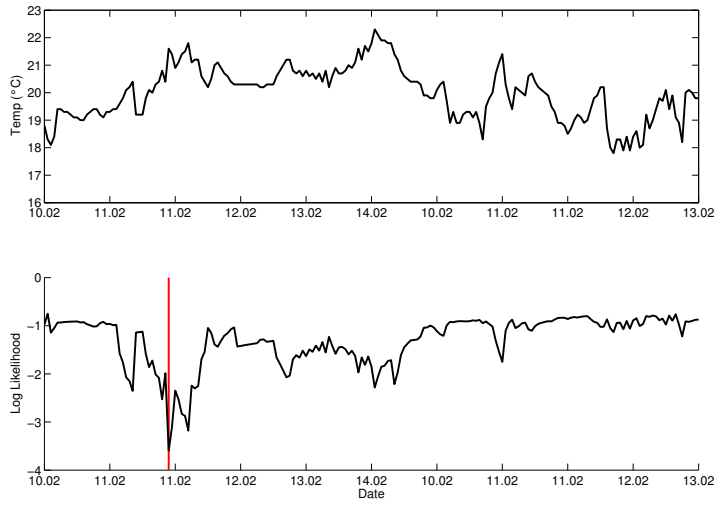


Figure 7.5: An example of system adaptation. On 11.02, the temperature is abnormally high, and an alarm is generated. On 14.02, a similar temperature is reached, and no alarm is generated.

Chapter 8

Case Study: HMMs for Traffic Observation

As announced earlier, I implemented and tested a model structure based on HMMs with mixture of Gaussians as emission models. The traffic situation of tunnels is used as the test environment. Unusual traffic situation should be automatically detected.
The automated method is based on statistical models of sensor behavior in combination with a Markov model of monitoring points. A model of normal traffic flow is automatically constructed. The model's structure and parameters are optimized using a mini-batch model-merging and parameter-updating algorithm. Incoming velocity vectors are conveyed to the model and the most probable path through the tunnel's Markov model is computed. An alarm is generated when the sensor values have a low probability under the model. The performance, strengths and weaknesses of the automated traffic flow analysis system are discussed below.
In this case study a slightly modified scenario concept is used. The whole traffic through the tunnel is seen as the scenario. The scenario consists of a number of paths - sub scenarios - as presented in section 5.3. These paths - generated from the actual data - can be more or less usual over time. The main advance compared to systems that observe the velocities at a particular point in the tunnel (seen from a single camera) is that this system has the overview of the traffic situation of the whole tunnel and can therefore give more informative statements.

8.1 Surveilance Systems for Tunnels

Tunnels play a crucial role in the importance of the transport sector for Europe's economy, therefore various research activities targeted to better control of tunnels have been started (BS06). In recent years the risks have increased with the ageing of tunnels because they are used more and more intensively. Controlling traffic tunnels is a complex and challenging task with very serious requirements, due to special tunnel conditions (illumination, environment, intrinsic characteristics of the tunnel) and the small timeframe to respond accordingly. When an incident occurs, tunnel operators have a tight timeframe for recognizing the incident, proceed to its verification and react properly. Incidents like driving in the wrong direction, presence of fire or smoke, or crashes between vehicles might cause huge damage depending

on the actors involved in the incident and its magnitude. Depending on the magnitude of the incident and its emergency level, tunnel operators have to notify the proper channels (police, roadway authorities, drivers, etc.), start the standard procedures, and activate alarm signals among many other tasks. Measures like stopping all vehicles at tunnel entrances, coordinating with the emergency services, dispatching the first rescue team to the incident scene, and alerting tunnel users of the emergency situation through tunnel radio broadcasting are important actions to be taken when an incident occurs. Taking the correct measures during the first few minutes after an incident is crucial to ensure the safety of the people involved. As a consequence, it is required that operators pay careful attention during the monitoring task. To facilitate this task, many road tunnels are already equipped with video systems allowing tunnel operators to supervizen tunnel activities. These video systems are operating 24/7, generating a huge amount of information, which cannot be completely supervised by the operators the whole time. Besides, the length of the tunnel complicates the problem even more because a longer tunnel implies more cameras, and therefore a larger amount of information.

Aforementioned facts result in an increase in the demand for automatic or semi-automatic tools to aid tunnel operators in detecting and managing abnormal behaviors and unexpected events. Automatic traffic scene analysis has gained great interest in the context of advanced transportation management systems and it has become essential in many areas of traffic analysis. As the development and implementation of Intelligent Transportation Systems (ITS) progresses, quick and accurate incident detection and effective emergency response become critical parts of advanced transportation management systems. In recent years, and as a result of advancements in hardware and software, many potentially reliable and efficient new incident detection methods have emerged. Automatic incident detection (AID) has received more attention, and different areas such as artificial intelligence, computer vision, neural networks, fuzzy logic, and video image processing contribute with a variety of algorithms to AID. Major improvements in performance and quality of results of machine-vision-based traffic surveillance systems demonstrate a great potential to attain the desired level of accuracy and reliability.

The research in Computer Vision applied to intelligent transportation systems is mainly devoted to providing them with situational awareness. A combination of computer vision methods with video technology is able to detect all major incidents: interrupted traffic flow or slow-moving traffic, and statistical information such as speed and vehicle classification. Advantages of video-based systems are a higher detection rate with a shorter mean detection time and the simple recording of raw data, among many others. However, problems like traffic lights, reflections, and varying weather conditions are the main challenges in video image analysis systems. In tunnels, reflections and low illumination conditions are intrinsic problems the need to be overcome. The abilities to detect all major incidents like slow-moving traffic, traffic jam, or classification of moving objects is demonstrated by previous research work (BBF+04; CPPS99; Rem97; VJ01) and commercial systems like ABT2000 [1], INVIS [2], VisioPad [3], Traffic Analysis System [4], Autoscope [5], Video Trak 910 [6], SiADS - SITRAFFIC [7],

[1] `www.artibrain.at`
[2] `www.invis-security.com`
[3] `www.citilog.fr`
[4] `www.crs-its.com/main.htm`
[5] `www.imagesensing.com`, `www.autoscope.com`
[6] `www.peek-traffic.com`
[7] `www.siemens.com/page/1,3771,1129794-0-14_0_0-10,00.html`

and Traficam [8] among many others - this list does not presume to be comprehensive. However, to the best of my knowledge, no work has been reported on digital image video analysis with SGMs and HMMs in tunnels. The goal of this work is to automatically recognize unusual traffic-flow conditions without the need of pre-programmed rules, user-entered parameters, or experienced operators. The system observes sensor data over time, constructs a model of "normality", and issues error alerts when sensor values - or combinations thereof - vary from the normal. The result is a system that can recognize abnormal traffic activity with minimal manual configuration of the system. Further, if sensor readings vary or drift over time, the system can automatically adapt itself to the new "normal" conditions, adjusting its error criteria accordingly.

The system, called SCRS (Semantic Concept Recognition System) (BSR06), which was initially investigated for use in building automation systems to detect human behavior, was tested with data generated from a traffic simulator. This chapter presents the results of this trial, highlights the strengths and weaknesses of the automated system, and suggests future areas of improvement.

8.2 System Structure

The goal of the SCRS model is to automatically discriminate system behavior in a running automation system. It does this by learning about the behavior of the automation system and by observing data flowing through the system. The SCRS builds a model of the sensor data in the underlying automation system, based on the data flow. The model comprises not only SGMs describing the possible sensor values but also a model of the underlying events that cause a change in system behavior. From that model, a HMM, the system can identify recurring scenarios - patterns within the sensor values - with slightly varying sensor values represented by the SGMs that model the emission probability distributions. The system is also capable of launching an alarm in case of occurence of new scenarios or variations within scenarios with very low probability under the model.
We use a set of statistical generative models to represent knowledge about the automation system. A statistical generative model takes as input a sensor value, status indicator, time of day, etc., and returns a probability between zero and one. Additionally, HMMs can be queried to deliver the most probable path through the model. In our case the most probable path can be interpreted as a similar (the best matching) traffic situation compared to the current one, which has already been learned.
Using SGMs has several advantages. First, because the model encodes the probability of a sensor value occurring, it provides a quantitative measure of "normality", which can be monitored to detect abnormal situations. Second, the model can be queried as to what the "normal" state of the system would be, given an arbitrary subset of sensor readings. In other words, the model can "fill in" or predict sensor values, which can help to identify the source of system behavior, the unusual traffic situation. Third, the model can be continuously updated to adapt to sensor drift or to slightly changing operation conditions - varying speed limits due to road works for example - of the system.
For the application described in this case study, HMMs were used with Gaussian models as emission probability distributions. It is not necessary to use a mixture of Gaussians for the emissions, because each Gaussian belongs to a certain state. Under this point of view,

[8] www.traficon.com

the whole system behaves like a set of mixtures of Gaussians, but the priors of the mixture distribution - coming from the transitions - vary with respect to the past.

In this surveillance application the idea of scenarios is used in a different way than in

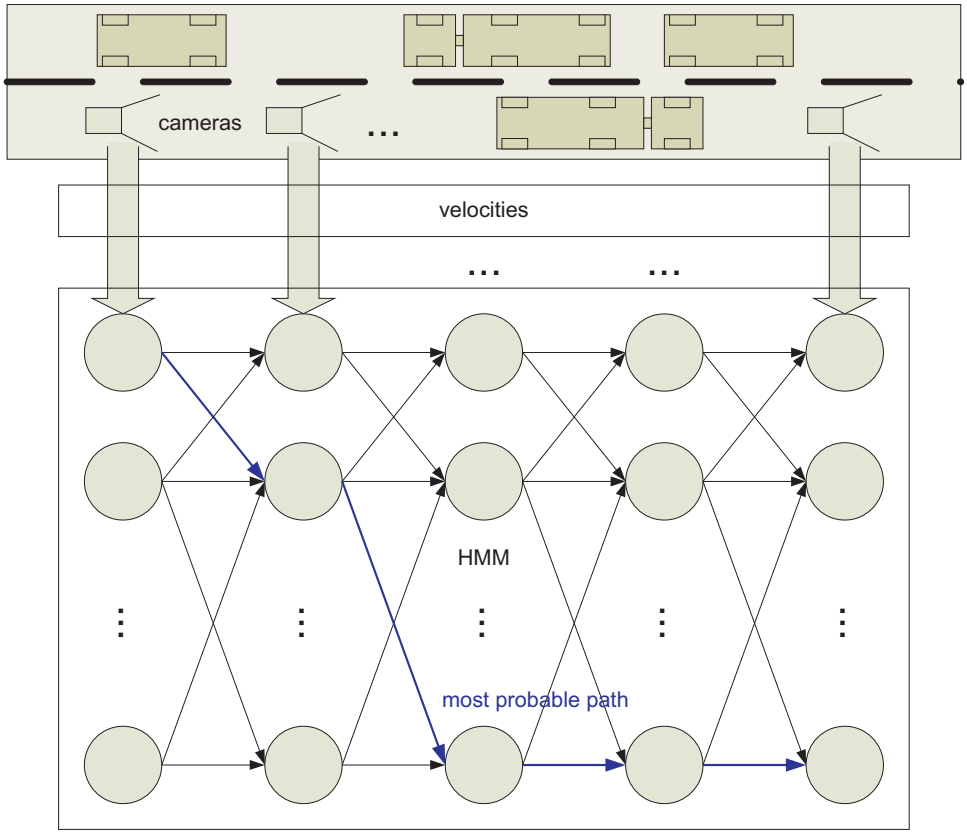


Figure 8.1: System Structure of the Surveillance System: Cameras are mounted equidistantly on the ceiling of the tunnel. Each time a camera recognizes a new vehicle and computes its velocity, a vector with the velocities of all cameras is passed to the HMM. The HMM on the one hand computes the most probable path and its log-likelihood and on the other it saves the vector for later incorporating.

building automation systems discussed earlier. Here the length of the state chain is fixed with the number of cameras C in the tunnel[9]. Each "snapshot" of velocity values (one per camera, even if it is not new) is said to be a sensor value chain. The chains of C values are then fed into the (empty) model and during a procedure of 3 steps (see also section 6.3) the parameters of the model are learned:

1) Comparison of the chain's beginning/end

2) Merging of identical states

3) Merging of consecutive states

The system architecture is depicted in figure 8.1.

[9]The number of cameras in the tunnel is specified by national authorities. In Austria, the distance between cameras is fixed to 200m for short tunnels (<500m) and 120m otherwise (ASF05).

8.3 System Adaptation

One important paramter of the SCRS for the analysis of the traffic situation is the number of velocity vectors to learn. In the current implementation the system takes the first 10 velocity vectors to form an initial model. Afterwards each new vector is compared to this model. The SCRS computes the Viterbi path for the new vector and the path's likelihood. After additional 10 vectors are received, the model computes a second model for the latest values. Afterwards, these two models are merged according to their priors. This procedure ensures that the model will learn new traffic situations, but with a low prior. The prior is the ratio of the number of new values to the number of values already seen. figure 8.2 shows the outcome of the just presented procedure for the first 450 velocity vectors. In a later phase - when the model is assumed to already learned all possible states - the baum-welsh algorithm can be used to adapt the model parameters (see section 3.4.3, the re-estimation formulas are given in equation 3.5 and equation 3.6). In the following 3 Figures the impact of the learning rate

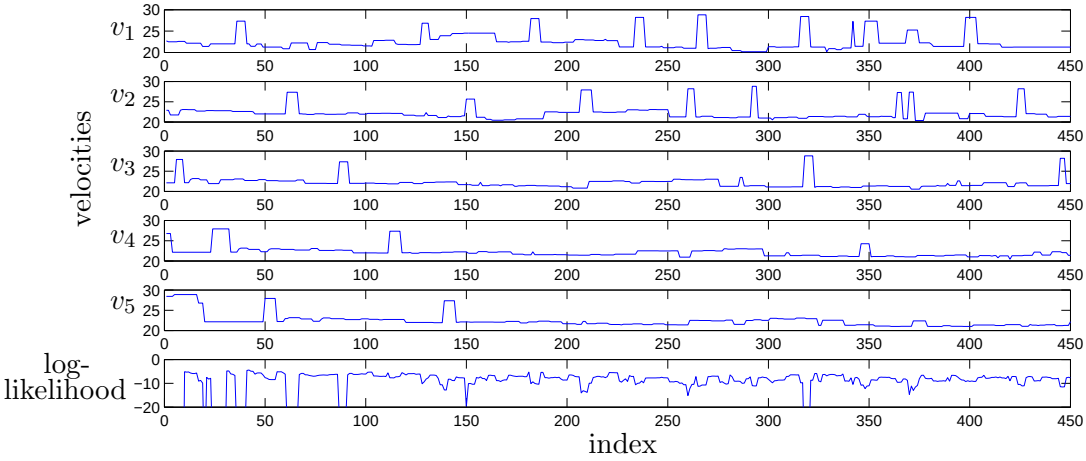


Figure 8.2: Simulated velocity values of a tunnel with 5 cameras. The bottom part of the figure is the overall likelihood of the model. At the beginning each new situation causes the system to drop in the likelihood, while it stays around -7 to -10 during normal conditions. The large drop of the likelihod at index 320 is caused by the coincident occurence of large velocity values at camera 1 and 3.

is investigated. figure 8.3 depicts a simulation run where every 2 new values are merged. The likelihood is therefor not as "flat" as before, but peaks are not as broad, because the model adapts very quickly to new conditions. The simulator starts to simulate a traffic jam after about 450 vectors, therefore the likelihood decreases dramatically (about -30 per camera). In the first figure, the likelihood raises up to about -13 after the first few vectors, indicating a degradation of the likelihood caused by a traffic jam of about -6.
The next simulation run had a lower learning rate. Each new 20 models are merged into the system. The result is shown in figure 8.4. We recognize a longer learning phase with likelihood drops, but a higher likelihood in the later phase for common situations (about -4 to -5). This observation is continued in figure 8.5. The average likelihood sinks, but the values for common situations can be very high (up to -1,5).
This behavior can be explained when having the structure of the model in mind: an HMM with Gaussians to model the velocities. When many values are used to learn each of the

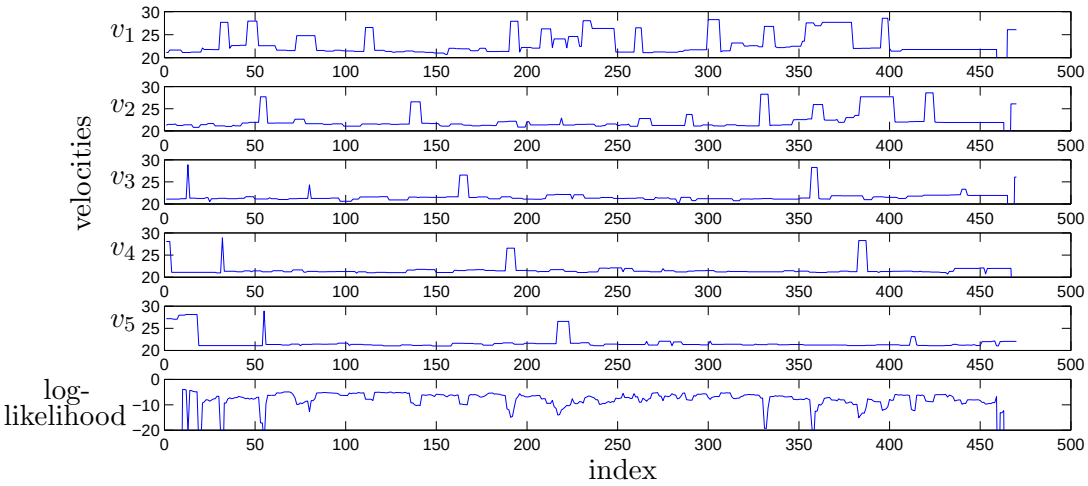


Figure 8.3: Simulated velocity vectors for the SCRS with a very high learning rate. The system adapts very quickly to new conditions.

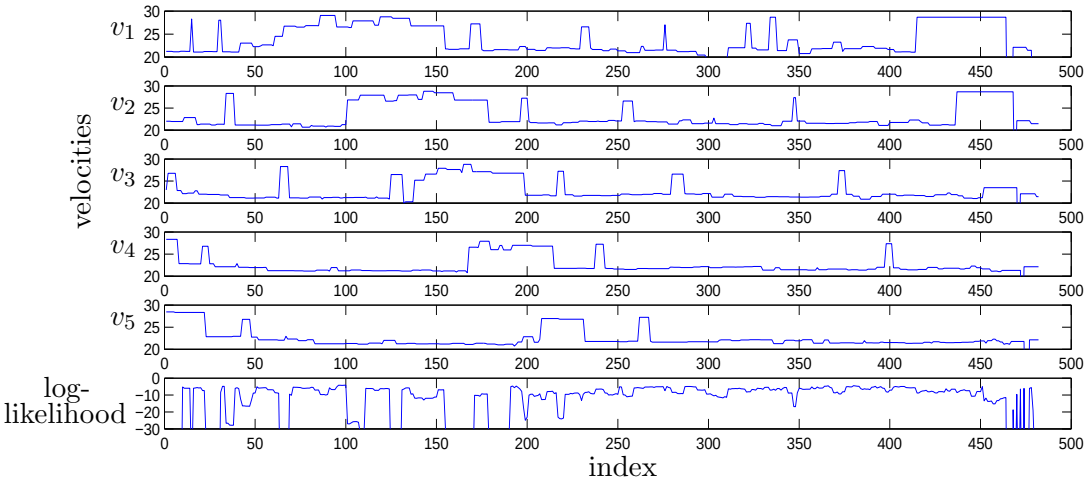


Figure 8.4: Simulated velocity vectors for the SCRS with a medium learning rate. The time in which the likelihood drops because of new situations is very long. On the other hand the likelihood for common situations is higher than in the case of faster learning rates.

Gaussians, the variance becomes low and therefore the Gaussians become narrow. A narrow Gaussian gives very high values for "fitting" velocities and low for unusal ones. However, for ensuring correct detection of situations the model needs many sensor values to learn about all possible combinations thereof. This is because it learns not just about parameters of one camera, but the overall situation. Unfortunately, the simulator was only capable of delivering roughly 500 values, which is enough to make observations about the system behavior, but far too few to fully learn the model. When thinking of a simulator capable of delivering much more values, it would be also of interest to have actual data from real tunnels, because the "normal" traffic situation in a particular tunnel may not be usual for another one.

The simulator started to simulate a traffic jam after 450 values. The model was capable - independent of the learning rate - to successfully detect that incident.

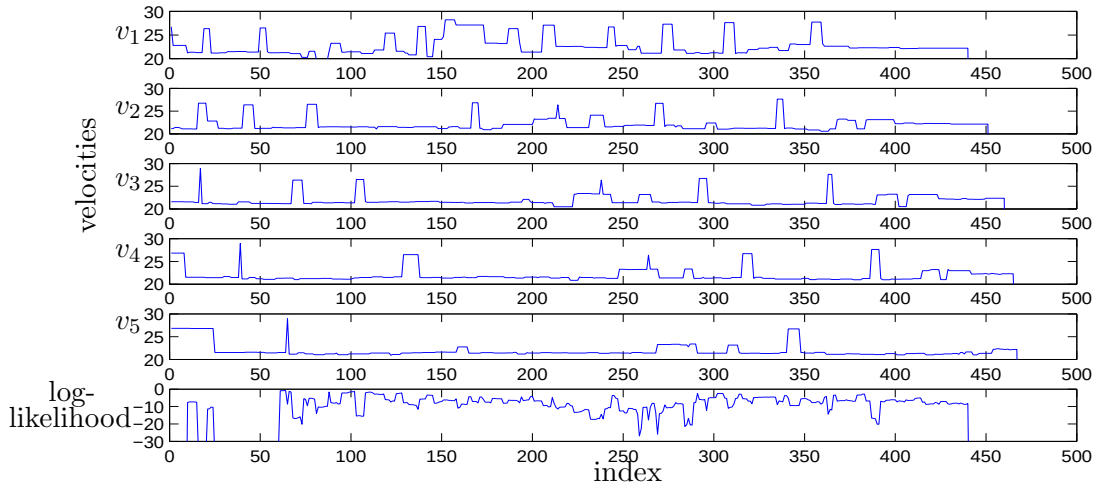


Figure 8.5: Simulated velocity vectors for the SCRS with a slow learning rate. The differentiation between very common and uncommon situations becomes sharper. The likelihood for common situations reaches -1,5 several times, but is lower in average than in the case of faster learning.

Because the model uses the Viterbi algorithm to choose from already seen traffic situations, it would be easy - as it is a built-in function - to detect at which camera the problem occured (in the sense that it finds the adequate path). If - during some negotiation procedure - the user already defined a reaction on such an incident, the system could automatically direct the user's attention to that camera.
Another advantage of the system comes to light in case the velocities at all cameras are within the values seen before, but the simultaneous occurance of groups of values or the sudden change of values can indicate an unusual situation and therefore launch an alarm.

Chapter 9

Discussion and Outlook

A method to automatically condense semantic information from building automation sensor data is described. Representations of the behavior of humans are presented and a way of learning the structure and parameters thereof is illustrated, along with descriptive examples. Finally, case studies to illustrate the results are presented. One case study deals with how an error detection system models functions compared to standard rule-based methods in a large building. The second shows the application of the HMM-based system to observe traffic in tunnels and to launch alarms in case of traffic jam, fire or other problems.

In chapter 5 I present ways to represent sensor values in order to combine them to form scenarios. First, a definition of scenarios is given, then models for the lower levels of the model structure are introduced. These models can be used to fit the data of sensors with different time frames as discussed in section 6.1.

The example implementation presented in chapter 6 describes a test of the utilization of HMMs to model sensor values in an office building and the attempt of constructing symbols with semantic meaning, semantic concepts. The system can automatically build a model of various daily routines in an office environment. It does this by using a batch-learning algorithm to create the model and an on-line mini-batch version during normal operation. Learning incoming value chains - daily routines - is used to create a new path through the model or - if parts of it already exist - just to create a new smaller piece of the model. After a predefined number of examples are seen, some state merge procedures are applied, which produce a model with a manageable number of states to be interpreted by humans.

I found that the system does create models with paths and states that do have a meaning. In some cases a state has a meaning that can be directly related to some particular event like the appearance and disappearance of the cleaning person, the whole morning and evening in the office, as well as state 13 that represents the normal working day. These states, or even whole paths like 1-5-13-4, can be seen as high-level semantic concepts, and can be easily interpreted by human operators. The learned semantic concepts divide the daily routine in the office environment into similar segments as would be intuitively used by a human operator.

The work presented in chapter 7 describes a test of statistical methods for the automatic detection of abnormal sensor values. The BASE system can automatically build a model of normal sensor behavior. It does this by optimizing model parameters using an on-line maximum-likelihood algorithm. Incoming sensor values are then compared to the model, and

an alarm is generated when the sensor value has a low probability under the model. The model parameters are continuously adapted on-line.

I found that the traditional system and the BASE system complement one another. While the traditional system was able to detect statistically insignificant but important deviations using thresholds, BASE was able to detect deviations that were within the thresholds used by the traditional system, but were nevertheless significant deviations from normality. These alarms have implications for system reliability, safety, security, efficiency and user comfort.

The work presented in chapter 8 describes a test of the HMM approach for the automatic detection of abnormal traffic situations in tunnels. The SCRS system can automatically build a model of normal traffic situations. Each path through the model represents a particular traffic situation of the whole tunnel. To ensure that even unusual situations are represented correctly, auxilary models of newly arrived sensor values are built, and these are merged with the original model. The SCRS possesses the clear advantage of providing an overview of the whole tunnel, while normal systems use the values of each camera independently.

The case study showed that the SCRS is capable of differentiating between usual and unusual traffic situations. With the limited data available, we have shown that the method can detect system-wide disturbances, and offers a clear advantage over systems which consider each camera alone. The system has shown its abilities to give a quantitative measure of the likelihood of the traffic situation to be used by other systems like fire or smoke detection systems, systems for counting vehicles or systems to detect crashes and jams. The detailed study of the system using a comprehensive set of simulations for various alarm conditions must be left as future work.

In the near or farther future, ambient intelligence systems will be used to observe several items. Therefore, improvements on context awareness, disappearing computing and many other fields have to be carried out. Also, effort in symbolic processing and representation of system inputs, system states and context has to be increased. This includes higher sophisticated methods for learning, storing and recognizing meaningful scenarios. This work is a basic proof of concept and - with the words of Weizenbaum - shall be a small light of a new lantern a few steps away from the way that is currently illuminated.

In this thesis I have shown that statistical models are worth being considered when trying to improve building automation systems. They are capable of adapting to real conditions in a way that is impossible to pre-define by a system operator. When using a second level of abstraction - in the sense of not just finding patterns within the output values of some sensor, but also trying to find patterns over time - a system can be constructed that learns to adapt to recurring patterns in daily routines (or other time bases). These patterns are modelled and represented in a way that a human operator is able to label (most of) the states in an intuitive way.
On the other hand, the method presented here still lacks functionality. Possible improvements and further investigations could be made in the following topics:

- There are already systems dealing with pre-defined scenarios. A comparison of the output would be interesting to see, if the results have similarities. In a further step, a method to compare scenarios - and so have a possibility to label the learned ones - is necessary for future deployment[1].

[1] (NHG06) for example proposed a method to compute the semantic similarity of concepts from different

- Learning of higher levels in the hierarchy. Currently, the system is able to deal with states, sub-scenarios and scenarios. But, there are still correlations in a higher level. A first step could be the introduction of transitions between (sub-)scenarios. So could the system with the motion detector presented in chapter 6 automatically learn the probability of having a "weekend day" followed by a "normal day".

- The whole question of actuators is left open. In this implementation, a drop in the likelihood can be used to launch an alarm. For future deployment, policies for controling actuators have to be investigated. A valuable hint on the necessity to start an action could be a change in the Viterbi path over time. This happens when the system recognizes that another path through the model matches the seen data better than the one expected up to now.

- Another important aspect is the data type of the sensors. I already mentioned what type and how often several sensors generate data. A procedure that automatically detects which type of sensor it is, and automatically chooses the right model would enhance the system.

When we think about a later product, the questions on accaptance and social, ethical, and legal implications rise. It is important to point out here that the methods presented in this and related work have the possibility to enhance privacy for the observed users in several respects. Among others the two most important are:

1) People are observed by machines. Only unusual or dangerous situations are reported to other people. Therefore, the user gains freedom. He is not observed in the sense of a surveillance system with a security guard behind that watches every movement, but protected in the sense that only in case of a need someone else is informed.

2) Sensor values from e.g. cameras are processed locally, only the results are distributed over the network. This means it is impossible to gain access to sensitive video data by an intruder, because it is not transmitted.

It will be interesting to see if there will be discussions in the public on what and how and how much may be observed. Currently, I see two groups of people, the one who are totally pro and the one who are or totally against observation. The former invoke security reasons to support their opinions while the latter state that their personal freedom of not being observed is more important than society's need for security. Well, everybody wants to be secure[2], but the borderline between an effort to make people safe and the same effort dissipating in the administration thereof without actually enhancing the security level is narrow. On the other

ontologies based on five categories: syntactic, properties, neighborhood, context similarity and equivalent similarity.

[2]According to Maslow's hierarchy of needs, the need for security is on the second level, which he calls the safety level. Prior to this, the physical level, which consists of breathing, food, water, sex, sleep, homeostasis and excretion, has to be satisfied. If we consider a society that fulfills at least level one and has the feeling of fulfilling much more, the trust of the people in the security of the whole society can be concussed easily by shocking events like terror attacks. At this point, there are two extreme ways of dealing with the psychology of mass population: to explain that terror attacks cannot be prevented or to demand some security-enhancing activities, which also carry the message: the representatives know what to do. Either way, my point is that the security a society wants cannot be quantified, has often nothing to do with the actual imminence of danger but can be influenced by predicting danger or even by predicting safety.

hand: what is personal freedom? In my opinion, the freedom of each individual - or each right guaranteed to individuals by the society through its legislatures - limits the freedom of everybody else. So, if somebody demands the right for people to take some action, he demands at the same time that society be prohibited from banning exactly this action. Therefore, society - in most cases through its democratically elected representatives - has to find a level with which everybody can live.
The implementation of the ubiquitous computing vision opens up new possibilities to rebalance the freedom of individuals and the freedom or needs of society. Maybe this act of rebalancing will reach a level - or, even earlier, the anticipation of a possibility to rebalance - that affects people to really start with fruitful discussions on a broad basis. I will be there, see you.

List of Algorithms

List of Figures

Bibliography

ABD+01 Allen, J. ; Byron, D. ; Dzikovska, M. ; Fergusen, G. ; Galescu, L. ; Stent, A.: Towards conversational human-computer interaction. In: *AI Magazine* 22 (2001), Nr. 4, S. 27–37 2

ACGHW Austin, J. ; Cliff, D. ; Ghanea-Hercock, R. ; Wright, A. *Large-Scale, Small-Scale Systems: Foresight Cognitive Systems Projects Research Review* 38

AH03 Aebischer, B. ; Huser, A.: Energy analyses of the FutureLife-House. (2003) 2

ASF05 ASFINAG: Videosysteme. (2005), S. 76–77 102

BBF+04 Bertozzi, M. ; Broggi, A. ; Fascioli, A. ; a. Tibaldi ; Chapuis, R. ; Chausse, F.: Pedestrian Localization and Tracking System with Kalman Filtering. In: *IEEE Intelligent Vehicles Symposium*, 2004 100

Ber99 v. Berlo, A.: Design Guidlines on Smart Homes. In: *COST 219bis Guidebook, European Commission* (1999) 41

BFD05 Bartolomeu, P. ; Fonseca, J. A. ; Duarte, P.: MIDI over Bluetooth. In: *Proc. of the 10th IEEE EFTA, Catania, Italy* (2005) 45

Bis95 Bishop, C. M.: *Neural Networks for Pattern Recognition*. New York NY. : Oxford University Press Inc., 1995 9, 19, 20, 22

BIT04 Beaudin, J. ; Intille, S. S. ; Tapia, E. M.: Lessons Learned Using Ubiquitous Sensors for Data Collection in Real Homes. In: *Extended Abstracts of the 2004 Conf. on Human Factors in Computing Systems* (2004), S. 1359–1362 41

BJM83 Bahl, L. R. ; Jelinek, F. ; Mercer, R. L.: A maximum likelihood approach to speech recognition. (1983), S. 308–319 32

BP01 Bodmer, J. ; Pfeiffer, W. K.: *Home and building automation system*. 2001. – Patent Number CH690875, WO9744720 (A1), EP0900417 (A1), US6263260 (B1), EP0900417 (B1), CN1179256C (C) 44

BRT02 Borodulkin, L. ; Ruser, H. ; Traenkler, H-R.: 3D Virtual "Smart Home" User Interface. (2002) 2

BS06 B. Strobl, et. a.: Vitus - Tunnel Safety through video-based Analysis. In: *Proceedings of the 13th World Congress and Exhibition on Intelligent Transport Systems and Services*, 2006 99

BSR06 Bruckner, D. ; Sallans, B. ; Russ, G.: Probabilistic Construction of Semantic Symbols in Building Automation Systems. In: *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, 2006 73, 101

BSS02 vom Boegel, G. ; Schliepkorte, H-J. ; Scherer, K.: Operation of Appliances controlled by Smart Labels. (2002) 2

CGMR05 Cohignac, T. ; Guichard, F. ; Migliorini, C. ; Rousson, F.: *Method and System for Detecting a Body in a Zone Located Proximate an Interface.* 2005. – Patent Number WO2005013226, FR2858450 (A1) 44

CPPS99 Cucchiara, R. ; Piccardi, M. ; Prati, A. ; Scarabottolo, N.: Real-time Detection of Moving Vehicles. In: *Proc. of 10th International Conf. on Image Analysis and Processing*, 1999, S. 618–623 100

CV95 Cortes, C. ; Vapnik, V.: Support Vector Networks. In: *Machine Learning* 20 (1995), S. 273–297 44

CWG66 Colby, K. M. ; Watt, J. B. ; Gilbert, J. P.: A Computer Method of Psychotherapy. Preliminary Communication. In: *The Journal of Nervous and Mental Disease* (1966), Nr. 142, S. 148–152 7

Day69 Day, N. E.: Estimating the components of a mixture of normal distributions. In: *Biometrika* 56 (1969), S. 463–474 18

Dec02 Decotignie, J. D.: Wireless fieldbusses - a survey of issues and solutions. In: *IFAC World Congress* (2002) 45

Die00 Dietrich, D.: Evolution potentials for fieldbus systems. In: *IEEE Int. Workshop on Factory Communication Systems WFCS 2000* (2000) 4

DLP+06 Deutsch, T. ; Lang, R. ; Pratl, G. ; Brainin, E. ; Teicher, S.: Applying Psychoanalytic and Neuro-Scientific Models to Automation. In: *Proceedings of the 2nd International Conference on Intelligent Environments*, 2006, S. 111–118 49

DLR77 Dempster, A. P. ; Laird, N. M. ; Rubin, D. B.: Maximum likelihood from incomplete data via the EM algorithm. In: *J. Royal Statistical Society Series B* 39 (1977), S. 1–38 19

DRA04 Demiris, G. ; Rantz, M. J. ; Aud, M. A.: Older adults attitudes towards and perception of smart home technologies: a pilot study. In: *Med. Inform.* 29 (2004), Nr. 2, S. 87–94 41

DRT+01 Dietrich, D. ; Russ, G. ; Tamarit, C. ; Koller, G. ; Ponweiser, M. ; Vincze, M.: Modellierung des technischen Wahrnehmungsbewusstseins für den Bereich Home Automation. In: *e&i* Bd. 11, 2001, S. 454–455 4

EBM05 Endres, Christoph ; Butz, Andreas ; MacWilliams, Asa: A Survey of Software Infrastructures and Frameworks for Ubiquitous Computing. In: *Mobile Information Systems Journal* 1 (2005), January–March, Nr. 1 1

EH81 Everitt, B. S. ; Hand, D. J.: *Finite Mixture Distributions*. London : Chapman and Hall, 1981 14

Eie99 Eier, R.: Maped Markov chains for modeling non-Markovian processes. In: *Proceedings Volume 5, Computer Science and Engineering*, 1999, S. 348–355 26

FH99 Fellbaum, K. ; Hampicke, M.: Integration of Smart Home Components into Existing Residences. In: *Proc. of AAATE* (1999) 41

FMM+03 Friesdorf, W. ; Meyer, S. ; Mollenkopf, H. ; Fellbaum, K. ; Dienel, L. ; Blessing, L. ; Heine, A.: Sentha-Projekt - Wissenschaftliche Erkenntnisse. In: *SENTHA Publikation* (2003) 41

FS94 Fasolo, Paul ; Seborg, Dale E.: An SQC Approach to Monitoring and Fault Detection in HVAC Control Systems. In: *Proceedings of the American Control Conference*. Baltimore, Maryland, 1994 20

Fue03 Fuertes, C. T.: *Automation System Perseption*. Vienna, Austria : Institute of Computer Technology, Technical University of Vienna, 2003. – Dissertation thesis 4, 5

GLS+00 Goulding, P.R. ; Lennox, B. ; Sandoz, D.J. ; Smith, K. ; Marjanovic, O.: Fault Detection in Continuous Processes Using Multivariate Statistical Methods. In: *International Journal of Systems Science* 31 (2000), Nr. 11, S. 1459–1471 20

GSB02 Gellersen, H-W. ; Schmidt, A. ; Beigl, M.: Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts. In: *Mobile Networks and Applications* 7 (2002), S. 341–351 2

Hai06 Hainich, R. R.: *The End of Hardware, A Novel approach to Augmented Reality*. Booksurge, 2006 1

HJ97 Hood, Cynthia S. ; Ji, Chuanyi: Proactive Network Fault Detection. In: *INFOCOM (3)*, 1997, S. 1147–1155 20

HLS99 House, J.M. ; Lee, W. Y. ; Shin, D. R.: Classification Techniques for Fault Detection and Diagnosis of an Air-Handling Unit. In: *ASHRAE Transactions* 105 (1999), Nr. 1, S. 1987–1997 20

HMNS03 Hansmann, U. ; Merk, L. ; Nicklous, M.S. ; Stober, T.: *Pervasive Computer, The Mobile World*. Springer Professional Computing, 2003 1

HSG98 Hinton, G. E. ; Sallans, B. ; Ghahramani, Z.: A Hierarchical Community of Experts. In: Jordan, M. I. (Hrsg.): *Learning in Graphical Models*. Kluwer Academic Publishers, 1998, S. 479–494 19

HY02 Haigh, K. Z. ; Yanco, H. A.: Automation as Caregiver - A Survey of Issues and Technologies. In: *AAAI Workshop Automation as Caregiver* (2002), S. 39–53 41

Int02 Intille, S. S.: Designing a Home for the Future. In: *IEEE Pervasive Computing* (2002), April–June, S. 80–86 2

Ise84 Isermann, R.: Process Fault Detection Based on Modeling and Estimation Methods – A Survey. In: *Automatica* 20 (1984), Nr. 4, S. 387–404 20

Jaa97 Jaakkola, T. S.: *Variational Methods for Inference and Estimation in Graphical Models*. Cambridge, MA : Department of Brain and Cognitive Sciences, MIT, 1997. – Ph.D. thesis 20

JGJS99 Jordan, M. I. ; Ghahramani, Z. ; Jaakkola, T. S. ; Saul, L. K.: An introduction to variational methods for graphical models. In: *Machine Learning* 37 (1999), S. 183:233 20

Kal60 Kalman, R. E.: A new approach to linear filtering and prediction problems. In: *Trans. ASME, Series D, Journal of Basis Engineering* 82 (1960), March, S. 35–45 20

KDP02 Kabitzsch, K. ; Dietrich, D. ; Pratl, G. *LonWorks Gewerkeübergreifende Systeme*. 2002 19

KNSN05 Kastner, W. ; Neugschwandtner, G. ; Soucek, S ; Newmann, H. M.: Communication System for Building Automation and Control. In: *Proc. of the IEEE, Special Issue on Industrial Communication Systems* 93 (2005), S. 1178–1203 19, 44

Lag00 Lagendijk, R. L. *The TU-Delft Research Program "Ubiquitous Communications"*. 2000 2

LBD+ Litzenberger, M. ; Bauer, D. ; Donath, N. ; Garn, H. ; Kohn, B. ; Posch, C. ; Schön, P.: Embedded Vehicle Counting System With 'Silicon Retina', Optical Sensor. In: *AIP Conference Proceedings* 71

LDS01 Loy, D. (Hrsg.) ; Dietrich, D. (Hrsg.) ; Scheinzer, H.J. (Hrsg.): *Open Control Networks: LonWorks/EIA 709 Technology*. Boston, Dordrecht, London : Kluwer Academic Publishers Boston, 2001 19

LGS99 Lee, T-W ; Girolami, M. ; Sejnowski, T.: Independent Component Analysis using an Extended Infomax Algorithm for Mixed Sub-Gaussian and Super-Gaussian Sources. In: *Neural Computation* 11 (1999), Nr. 2, S. 417–441 19

LGWA05 Lindeloef, D. ; Guillemin, A. ; Wilhelm, L. ; Altherr, R.: AHeart: a hardware-independent building control API. (2005), S. 598–600 2

LMB+03 Lindwer, M. ; Marculescu, D. ; Basten, T. ; Zimmermann, R. ; Marculescu, R. ; Jung, S. ; Cantatore, E. *Ambient Intelligence Visions and Achievements; Linking abstract ideas to real-world concepts*. 2003 1

LPD Lichtensteiner, P. ; Posch, C. ; Delbruck, T.: A 128 x 128 120dB 30mW Asynchronous Vision Sensor that responds to Relative Intensity Change. In: *Proceedings of the 2006 IEEE International Solid-State Circuits Conference* 71

LS03 Lohse, M. ; Slusallek, P.: Middleware Support for Seamless Multimedia Home Entertainment for Mobile Users and Heterogeneous Environments. (2003), S. 217–222 2

Mah04 Mahlknecht, S.: *Energy-Self-Sufficient Wireless Sensor Networks for the Home and Building Environment*. Vienna, Austria : Institute of Computer Technology, Technical University of Vienna, 2004. – Dissertation thesis 70

Mat04 Mattern, F.: Ubiquitous Computing: Schlaue Alltagsgegenstände - Die Vision von der Informatisierung des Alltags. In: *Bulletin SEV/VSE* (2004), Nr. 19, S. 9–13 1

MB88a MacLachlan, G. J. ; Basford, K. E.: *Mixture Models.* New York : Marcel Dekker, Inc., 1988 14

MB88b McLachlan, G. J. ; Basford, K. E.: Mixture Models: Inference and Applications to Clustering. In: *New York: Marcel Dekker* (1988) 16

MB04 Mahlknecht, S. ; Boeck, M.: CSMA-MPS: A Minimum Preamble Sampling MAC Protocol for Low Power Wireless Sensor Networks. In: *IEEE WFCS 2004*, 2004 70

MHDP05 M., Wollschlaeger ; H., Kulzer ; D., Nuebling ; P., Wenzel: A Common Model for XML Descriptions in Automation. In: *Proceedings of the IFAC World Congress 2005, Prag*, 2005 20

MK03 Mavrommati, I. ; Kameras, A.: The evolution of objects into Hyper-objects. In: *Personal and Ubiquitous Computing* 7 (2003), Nr. 1, S. 176–181 2

MM02 Marples, D. ; Moyer, S.: Guest editorial: in-home networking. In: *IEEE Communications Magazine* 40 (2002) 44

Moz98 Mozer, M. C.: The neural network house: An environment that adapts to its inhabitants. (1998), S. 110–114 2

Moz05 Mozer, M. C.: Lessons from the adaptive house. In: *Cook & Das, Smart Environments: Technologies, protocols and applications* (2005), S. 273–294 44

Neu99 Neuhold, H.: Die Wohnbedürfnisse älterer Menschen. In: *G. Schöpfer: Seniorenreport Steiermark, Graz* (1999) 41

NH98 Neal, R. M. ; Hinton, G. E.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Jordan, M. I. (Hrsg.): *Learning in Graphical Models.* Kluwer Academic Publishers, 1998, S. 355–368 20

NHG06 Ngan, Le D. ; Hang, Tran M. ; Goh, Angela Eck S.: Semantic Similarity between Concepts from Different OWL Ontologies. In: *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, 2006 107

Ni05 Ni, Q.: Performance analysis and enhancements for IEEE 802.11e wireless networks. In: *IEEE Network* 19 (2005), S. 21–27 45

ODK96 Ostendorf, M. ; Digalakis, V. ; Kimball, O. A.: From HMMs to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition. (1996), S. 360–378 32

OH05 Oliver, N. ; Horvitz, E.: S-SEER: Selective Perception in a Multimodal Office Activity System. In: *Int. Journal on Computer Vision and Image Understanding* (2005) 44

ORP00 Oliver, N. ; Rosario, B. ; Pentland, A.: A Bayesian Computer Vision System for Modeling Human Interactions. In: *PAMI Special Issue on Visual Surveillance and Monitoring* (2000) 44, 67

Pal01 Palensky, P.: *Distributed Reactive Energy Management*. Vienna, Austria : Institute of Computer Technology, Technical University of Vienna, 2001. – Dissertation thesis 5

Pea86 Pearl, J.: Fusion, propagation and structure in belief networks. In: *Artificial Intelligence* 29 (1986), Nr. 3, S. 241–288 44

PL03 Pontoppidan, N.H. ; Larsen, J.: Unsupervised Condition Change Detection in Large Diesel Engines. In: Molina, C. (Hrsg.) ; Adali, T. (Hrsg.) ; Larsen, J. (Hrsg.) ; Hulle, M. V. (Hrsg.) ; Douoglas, S. (Hrsg.) ; Rouat, J. (Hrsg.): *IEEE Workshop on Neural Networks for Signal Processing*. Piscataway, New Jersey : IEEE Press, 2003, S. 565–574 20

PP05 Pratl, G. ; Palensky, P.: Project ARS - The next step towards an intelligent environment. (2005) 4

PPDB05 Pratl, G. ; Penzhorn, W. ; Dietrich, D. ; Burgstaller, W.: Perceptive Awareness in Building Automation. In: *Proceedings of the 3rd International Conference on Computational Cybernetics*, 2005, S. 259–264 49

Pra06 Pratl, G.: *Symbolization and Prozessing of Ambient Sensor Data*. Vienna, Austria : Institute of Computer Technology, Technical University of Vienna, 2006. – Dissertation thesis 4, 49

Rab89 Rabiner, L. R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Proceedings of the IEEE* (1989), S. 257–286 32, 44

Rem97 Remagnino, P.: An Integrated Traffic and Pedestrian Model-Based Vision System. In: *Proceedings of the Eight British Machine Vision Conference*, 1997, S. 380–389 100

RG99 Roweis, S. ; Ghahramani, Z.: A Unifying Review of Linear Gaussian Models. In: *Neural Computation* 11 (1999), Nr. 2, S. 305–345 24

RHC+02 Roman, M. ; Hess, C. K. ; Cerqueira, R. ; Ranganathan, A. ; Campbell, R. H. ; Nahrstedt, K.: Gaia: A Middleware Infrastructure to Enable Active Spaces. In: *IEEE Pervasive Computing* (2002), Oct–Dec, S. 74–83 2

RHW88 Rumpelhart, D. ; Hinton, G. ; Williams, R.: Learning internal representations by error propagation. In: *Neurocomputing* (1988), S. 675–695 44

RJ86 Rabiner, Lawrence R. ; Juang, Biing-Hwang: An Introduction to Hidden Markov Models. In: *IEEE ASSAP Magazine* 3 (1986), January, S. 4–16 20, 27

RLFV06 Roesener, C. ; Lorenz, B. ; Fodor, G. ; Vock, K.: Emotional Behavior Arbitration for Automation and Robotic Systems. In: *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, 2006 49

RM85 Russell, M.J. ; Moore, R.K.: Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition. In: *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. Tampa, 1985, S. 5–8 32

Rus03 Russ, G.: *Situation Dependent Behaviour in Building Automation*. Vienna, Austria : Institute of Computer Technology, Technical University of Vienna, 2003. – Dissertation thesis 4, 5

Sak99 Sakamura, K.: TRON-Specification VLSI CPU. In: *Journal of Information Processing Society of Japan* 40 (1999), Nr. 3, S. 252–258 2

Sal00 Sallans, B.: Learning Factored Representations for Partially Observable Markov Decision Processes. In: Solla, S. A. (Hrsg.) ; Leen, T. K. (Hrsg.) ; Müller, K-R (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 12, The MIT Press, Cambridge, 2000, S. 1050–1056 20

SBR05 Sallans, B. ; Bruckner, D. ; Russ, G.: Statistical Model-based Sensor Diagnostics for Automation Systems. In: *Proceedings of the 6th IFAC International Conference on Fieldbus Systems and their Applications*, 2005 94

SBR06 Sallans, B. ; Bruckner, D. ; Russ, G.: Statistical Detection of Alarm Conditions in Building Automation Systems. In: *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, 2006 94

Sch98 Schneider, S.: Ältere Bundesbürger in Privathaushalten und Heimen - Lebenssituation und Heimeintrittsgründe. (1998) 41

SH03 Schein, J. ; House, J.M.: Application of Control Charts for Detecting Faults in Variable-Air-Volume Boxes. In: *ASHRAE Transactions* Bd. 109, 2003, S. 671–682 20

Sil86 Silverman, B. W. *Density Estimation for Statistics and Data Analysis*. 1986 16, 17

Ski02 Skiczuk, Peter: *Network Protocol Architecture for Home Access Points*, Vienna University of Technology, Ph. D. dissertation, 2002 46

SO93 Stolcke, Andreas ; Omohundro, Stephen: Hidden Markov Model Induction by Bayesian Model Merging. In: Hanson, Stephen J. (Hrsg.) ; Cowan, Jack D. (Hrsg.) ; Giles, C. L. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 5, Morgan Kaufmann, San Mateo, 1993, S. 11–18 73

SZB04 Stefanov, D. H. ; Zeungnam, B. ; Bang, W-C: The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 12 (2004), S. 228–250 44

TIL04 Tapia, E. M. ; Intille, S. S. ; Larson, K.: Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In: *Pervasive* (2004), S. 158–175 41

TMIL04 Tapia, E. M. ; Marmasse, N. ; Intille, S. S. ; Larson, K.: MITes: Wireless Portable Sensors for Studying Behavior. In: *Proc. of Extended Abstracts Ubicomp* (2004) 41

TSM85 Titterington, D. M. ; Smith, A. F. M. ; Makov, U. E.: Statistical Analysis of Finite Mixture Distributions. In: *Wiley Series in Probability and Mathematical Statistics* (1985) 16

VJ01 Viola, P. ; Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple. In: *IEEE CVPR*, 2001 100

Weg98 Wegmann, M.: *Method for Surveying a Predetermined Surveillance Area*. 1998. – Patent Number WO9856182, EP0986912 (A1), US6628323 (B1), CA2290383 (A1), EP0986912 (B1), ES2166162T (T3) 44

Wei66 Weizenbaum, J.: ELIZA - A Computer Program For the Study of Natural Language Communication Between Man And Machine. In: *Communications of the ACM* (1966) 6

Wei76 Weizenbaum, J.: *Computer Power and Human Reason: From Judgement to Calculation*. W. H. Freeman, 1976 6

Wei91 Weiser, M.: The Computer for the 21st Century. In: *Scientific American* 265 (1991), Nr. 3, S. 66–75 1

Wei06 Weihs, Manfred: *Convergence of Real-time Audio and Video Streaming Technologies*, Vienna University of Technology, Ph. D. dissertation, 2006 46

WG96 Wise, B.M. ; Gallagher, N.B.: The Process Chemometrics Approach to Chemical Process Fault Detection and Supervision. In: *Journal of Process Control* 6 (1996), Nr. 6, S. 329–348 20

Wil03 Wildberger, M.: AI in Elder Care, Modeling & Simulation. In: *The Society for Modeling and Simulation International* 1 (2003), Nr. 4 44

WMW05 Willig, A. ; Matheus, K. ; Wolisz, A.: Wireless technology in industrial networks. In: *Proc. of the IEEE, Special Issue on Industrial Communication Systems* 93 (2005), S. 1130–1151 45

WSA+95 Want, R. ; Schilit, B. ; Adams, N. ; Gold, R. ; Petersen, K. ; Ellis, J. ; Goldberg, D. ; Weiser, M.: The PARCTAB ubiquitous computing experiment. (1995) 2

Zur05 Zurawski, R.: The Industrial Communications Handbook. In: *CRC Press* (2005) 44

Curriculum Vitae

Dipl.-Ing. Dietmar Bruckner

Personal Information:

Birth	22. 11. 1979, Zwettl, Austria
Marital Status	Single
Nationality	Austria

Education:

1998	school leaving examination (HTBLA Karlstein)
1999 – 2004	study Electrical Engineering, TU Vienna
2004	Master of Science (Dipl.-Ing.) Vienna University of Technology, passed with honors

Professional Experience:

01/04 – 10/04	Project ADI: design and implementation of an MPEG-4 encoder for a Blackfin DSP
11/04 – 10/06	Project ARS I and II: research project in cooperation with ARC Seibersdorf; applications of probabilistic models in automation networks; creation of funding proposals on national and European level
Since 11/06	Project SENSE: EU-funded research project on a network of cooperating sensor nodes to observe public accessible areas in buildings

Publications:

D. Bruckner, B. Sallans, G. Russ: Probabilistic Construction of Semantic Symbols in Building Automation Systems. In: Proceedings of 2006 IEEE INDIN´06, 2006.

B. Sallans, D. Bruckner, G. Russ: Statistical Detection of Alarm Conditions in Building Automation Systems. In: Proceedings of 2006 IEEE INDIN´06, 2006.

H. Hareter, G. Pratl, D. Bruckner: Simulation and Visualization System for Sensor and Actuator Data Generation. In: Proceedings of 6th IFAC FET'05, 2005, S. 56 - 63.

B. Sallans, D. Bruckner, G. Russ: Statistical Model-Based Sensor Diagnostics for Automation Systems. In: Proceedings of 6th IFAC FET'05, 2005, S. 239 - 246.

D. Bruckner: Mobile Plattform zur digitalen Bildbearbeitung für batteriebetriebene Roboter (German), Betreuer: S. Mahlknecht, Institut für Computertechnik, 2004.