Learning Situation Models for Providing Context-Aware Services

Oliver Brdiczka

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Numéro attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INP GRENOBLE

Spécialité: Imagerie, Vision et Robotique
préparée au laboratoire LIG au sein de l'équipe PRIMA
dans le cadre de **l'Ecole Doctorale**: Mathématiques, Sciences et Technologie de l'Information

présentée et soutenue publiquement

par

Oliver BRDICZKA

le 25 Mai 2007

LEARNING SITUATION MODELS FOR PROVIDING CONTEXT-AWARE SERVICES

Directeur de thèse : M. James L. CROWLEY
Co-Directeur de thèse : M. Patrick REIGNIER

JURY

Mme Marie-Christine ROUSSET, Présidente
M. Rüdiger DILLMANN, Rapporteur
M. Jean VANDERDONCKT, Rapporteur
M. Gilles PRIVAT, Examinateur
M. James L. CROWLEY, Directeur
M. Patrick REIGNIER, Co-directeur

Abstract

During the last decade, “intelligent” environments augmented by multiple sensors, including cameras, microphones, and interaction devices have enabled the computer observation of human activity. In order to detect and respond to human activity, a context model describing the environment, its users and their activities must be maintained. In this thesis, context is represented by situation models. A situation is a kind of state described by a number of characteristic roles and relations. A role is played by an entity and entities can be in relations. An entity refers to a person or object. A situation model can be defined and implemented by experts. Two example implementations of this kind are presented in this thesis. However, human behavior evolves over time. A situation model must be constructed and adapted automatically. This thesis addresses the problem by proposing a framework for the automatic acquisition and evolution of different layers of a situation model. Different learning methods are proposed as part of this framework. The situation model serves as frame and support for the different methods, permitting to stay in an intuitive declarative framework. The first layer of the framework concerns role learning and detection. Bayesian classifier, support vector machines (SVMs) and a novel hybrid classifier combining SVMs and Bayesian classifier are presented and compared. Based on the results of role detection, a method for unsupervised situation discovery is then proposed. This method takes a multimodal observation stream as input and generates a first segmentation into distinct observation sequences as output. This segmentation and associated situation labels given by an expert or user are the input for supervised situation learning. A supervised situation learning scheme is proposed. This scheme can be applied to different learner classes, generating a situation representation for each situation label and associated observation sequences. Based on the learned situation model, a method for the integration of user preferences is presented. This method adapts the learned situation model by splitting situations according to user feedback given on executed system services. The methods of the framework have been evaluated separately on data sets coming from different augmented environments. The complete framework has been integrated into an intelligent home environment. The implementation has been evaluated and the obtained results validate the proposed approach.

Keywords: context modeling, situation model, deterministic and probabilistic implementation of situation models, automatic acquisition and evolution of situation models, role detection, unsupervised situation discovery, supervised situation learning scheme, situation split.

Acknowledgments

First of all, I would like to thank my supervisor Prof. James L. Crowley and my co-supervisor Dr. Patrick Reignier for giving me the opportunity to enter into the world of research. I grateful thank them for providing me with the necessary freedom and support to realize my ideas during the last three years. Further, I am grateful to Prof. Rüdiger Dillmann, Prof. Jean Vanderdonckt, Dr. Gilles Privat and Prof. Marie-Christine Rousset for their interest in my work and for agreeing to be in the jury of this thesis.

I would also like to thank all members of the PRIMA research team for their support and fun during the last three years. Special thanks go to my colleague Jérôme Maisonnasse for many fruitful discussions and collaboration. I further thank Prof. Pong-Chi Yuen for interesting discussions and collaboration about context modeling during his sabbatical at INRIA Rhône-Alpes.

The work presented in this thesis is part of the CHIL project which is an Integrated Project (IP 506909) under the European Commission's Sixth Framework Program. The project gave me the opportunity to meet and work with many interesting people. France Télécom Project HARP provided further data sets and support for the final evaluations during this thesis.

Last, but not least, I owe grateful thanks to my parents for their unconditional support and encouragements during many difficult periods of this thesis.

Contents

Chapter 1

Introduction

Ubiquitous computing [94] integrates computation into every-day environments. The technological progress of the last decade has enabled computerized spaces equipped with multiple sensor arrays, including microphones or cameras, and multiple human-computer interaction devices. An early example is the Intelligent Room [33], a conference room augmented by several cameras, video projectors and displays as well as audio devices and microphones (Figure 1.1).

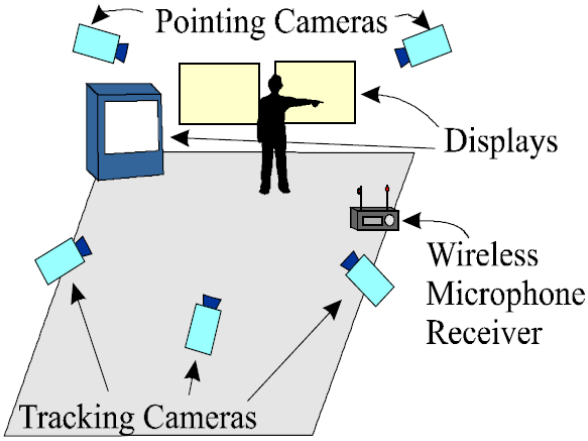


Figure 1.1: A simplified layout of the Intelligent Room at MIT AI Lab (picture from [34])

Smart home environments [22] and even complete apartments equipped with multiple sensors [35] have been realized. The major goal of these augmented environments is to enable devices to sense changes in the environment and to automatically act based on these changes. A main focus is sensing and responding to human activity. Human actors must be identified and their current activity needs to be recognized. Addressing the right user at the correct moment, while

perceiving his correct activity, is essential for correct human-computer interaction in augmented environments.

1.1 Problem Definition

Augmented “intelligent” environments have enabled the computer observation of human (inter)action within the environment. The analysis of (inter)actions of two or more individuals is of particular interest here because it provides information about social context and relations and it further enables computer systems to follow and anticipate human (inter)action. The latter is a difficult task given the fact that human activity is situation dependent [90] and does not necessarily follow plans. Computerized spaces and their devices require this situational information, i.e. context [41], to respond correctly to human activity. In order to become context-aware, computer systems must thus maintain a model describing the environment, its occupants and their activities. In order for user to trust these systems, system reasoning and behavior must, however, be kept transparent for the users. A human understandable context model is hence essential, representing user behavior and needs as well as system service execution.

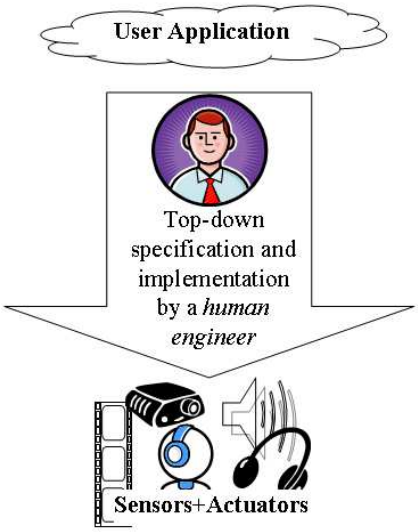


Figure 1.2: Top-down manual specification and implementation of a context model

Experts normally define and implement context models according to the needs of users and application (Figure 1.2). Based on user needs and envisaged application, a human engineer specifies and implements the context model. Sensor perceptions, context model and system services to be provided are associated manually.

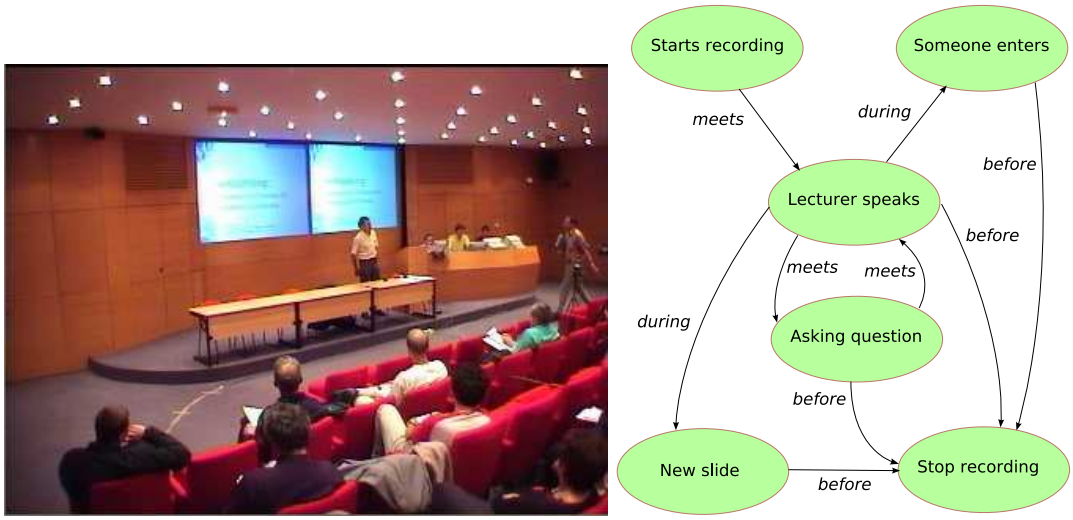


Figure 1.3: Wide-angle camera image of a lecture in the amphitheater at INRIA Rhône-Alpes (left) and a corresponding hand-crafted context model (in form of a situation graph) (right)

Figure 1.3 shows an example of such a top-down implementation. The wide-angle camera view of a lecture and a corresponding context model in form of a situation graph are depicted (see chapter 4 for details). The context model has been defined and implemented by engineers knowing the needs of users and application. The sensor perceptions are associated to the situations manually.

Human behavior evolves over time. New activities and scenarios emerge in an intelligent environment, others disappear. New services must be integrated into the environment, while obsolete services should be deleted. A fixed context model is thus not sufficient. Experts normally define, implement and adapt context models according to changing needs of users and application. However, experts are expensive and not always available. Moreover, the environment's intelligence lies in its ability to adapt its operation to accommodate the users. The research challenge is thus to develop machine learning methods for this process, making it possible to automatically acquire and evolve context models reflecting user behavior and needs in an intelligent environment (Figure 1.4).

We can distinguish two different motivations for learning context models:

1. Knowledge engineering (acquisition of a context model)
2. User preferences integration (adaptation of a context model)

A key requirement for context-aware intelligent environments is the automatic acquisition and evolution of an intuitive, comprehensible context model. Based on computer observation of the

environment, this context model must first be constructed as automatically as possible using machine learning methods (knowledge engineering), while staying transparent for the user. The user then should be able to integrate his preferences into this model, and adapt it constantly according to the evolution of his behavior and needs (user preferences integration). Intelligibility [12] of the context model and the reasoning process is important in order to permit the users to trust the system.

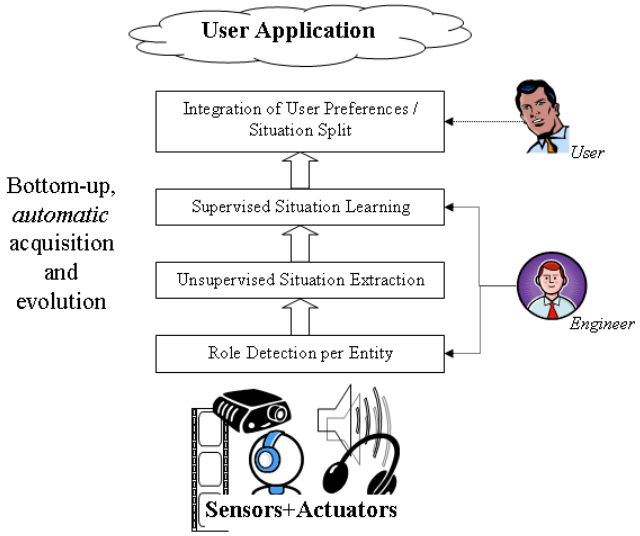


Figure 1.4: Bottum-up automatic acquisition and evolution of a context model

1.2 Approach

The proposed approach addresses the problem by providing an intelligible framework for acquiring and evolving a context model, called *situation model*. The methods proposed as part of this framework acquire different layers of the situation model, with different levels of supervision. The situation model serves as frame and support for the different learning methods, permitting to stay in an intuitive declarative framework.

The situation model and the underlying concepts are motivated by models of human perception of behavior in an augmented environment. Human behavior is described by a finite number of states, called *situations*. These situations are characterized by entities playing particular roles and being in relation within the environment. Figure 1.5 gives an example of a situation model for a lecture room. The situations “empty”, “lecture” and “audience” are characterized by the roles “lecturer” and “audience” as well as the relation “notSameAs”. The situation model has

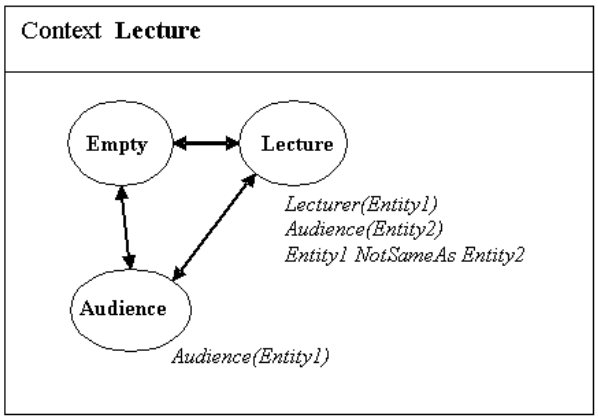


Figure 1.5: Example of a simple situation model for a lecture room. **Empty**, **Audience** and **Lecture** are the available situations. *Lecturer*, *Audience* are the available roles and *NotSameAs* the available relation

been used to implement different applications like an automatic cameraman, or an interaction group detector.

Figure 1.6 illustrates the framework for acquiring and evolving a situation model. First, roles are learned and detected based on collected data labeled by an expert. Situations are then extracted in an unsupervised manner from observation data. The extracted situation segments can then be used to learn situation labels with user or expert input. The resulting situation model can finally be evolved according to user feedback using the situation split.

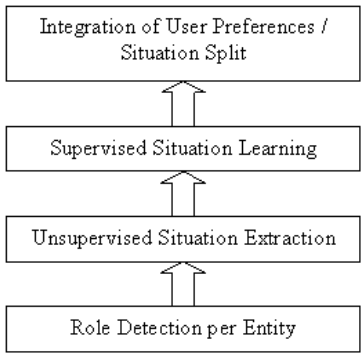


Figure 1.6: Framework for automatic acquisition and evolution of a situation model

Role learning and detection is based on event streams. These event streams contain the available entities as well as their properties. An acceptance test for a role is constructed by learning a role label from these entity event streams. Role labels refer to the abstract events necessary for role assignment. For this, we compare methods based on a Bayesian classifier, support vector

machines (SVMs) and a novel hybrid classifier combining Bayesian methods and SVMs.

The unsupervised extraction of situations is based on a stream of multimodal observations. The method that we propose detects change in the observation distribution by measuring the Jeffrey divergence between adjacent histograms of observations. These observation distributions are represented by histograms containing the frequency of these observations. To separate distinct distributions of observations, two adjacent windows are slid from the beginning to the end of the meeting recording, while constantly calculating the Jeffrey divergence between the histograms generated from the observations within these windows. The size of the sliding adjacent windows is varied generating several Jeffrey divergence curves. The peaks of the resulting curves are detected using successive robust mean estimation. The detected peaks are merged and filtered with respect to their height and window size. The retained peaks are finally used to select the best model, i.e. the best allocation of observation distributions for the given recording. This allocation corresponds to a first unsupervised segmentation of the situations.

Segments of observations and the provided situation labels are the input for supervised situation learning. A supervised learning scheme iterates over different classes of learners and the associated parameterization in order to acquire a representation for each situation label. The best representation for each situation label is retained. These representations can then be used to detect the situations and to construct the corresponding situation model.

The input for the integration of user preferences is a learned (or predefined) situation model along with user feedback from prior use. The feedback is given on executed system services (associated to situations). An algorithmic method adapts the associations between system services and situations according to the given feedback. If necessary, a situation is split into sub-situations, refining the perception of the system. The representations of the new sub-situations are learned by using the supervised situation learning scheme.

1.3 Evaluation and Results

The methods that we propose for acquiring and evolving situation models have been integrated into a complete system for an intelligent home environment. The implementation is based on a 3D tracking system that creates and tracks targets in the scene. The extracted targets are used to detect individual roles per entity. Observations are generated based on the role values of several entities. These observations are the input for unsupervised situation extraction. The results of the extraction process are used for supervised situation learning. The learned situation model is then the basis for the integration of user preferences, i.e. associating and changing system services with user feedback.

We have conducted two different evaluations (Figure 1.7). A first evaluation was designed to

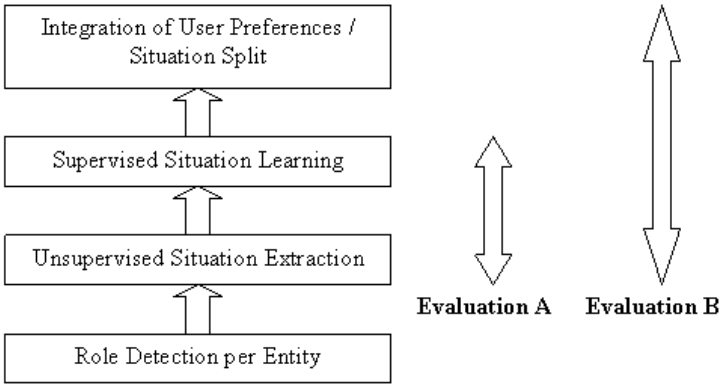


Figure 1.7: Different parts of the implementation and their evaluation: role detection per entity, unsupervised situation extraction, supervised situation learning and integration of user preferences

analyze the effects of automatic situation extraction and supervised situation learning for situation recognition. A second evaluation was then used to validate the combination of the three methods: unsupervised situation extraction, supervised situation learning and integration of user preferences.

The results of these evaluations validate our approach for an understandable framework for acquiring and evolving situation models. The results of the first evaluation show that unsupervised situation extraction and multi-person observation generation are beneficial for situation recognition. The results of the second evaluation indicate that the integration of the different methods into an intelligent home environment system is feasible. Even though the results are encouraging, the error rates are still excessive. Further improvements in detection and learning algorithms are necessary in order to provide a reliable system that could be accepted by a user in his daily life.

1.4 Thesis Outline

We give below an overview of the remainder of this thesis.

Chapter 2 analyses and defines the problem of this thesis with respect to the literature. “Intelligent” environments are defined as environments augmented with multiple sensors and interaction devices, including cameras, microphones, and video projectors. Several examples of intelligent environments are presented including smart offices, smart home environments and smart classrooms. The problem of context-aware service supply is motivated and discussed. In order to provide unobtrusive, proactive services, an augmented environment must be aware

of what the humans are doing in the environment and in which context human actions take place. Context refers not only to location information, but to any information that characterizes a situation related to the interaction between humans, application and surrounding environment. Context is further particular to each occasion of activity or action; the scope of contextual features is redefined dynamically. In order for users to trust a context-aware system, a user should be able to understand and scrutinize the context representation and reasoning of the system. Human behavior evolves over time. A context model must also evolve in order to accommodate the changing needs of the users. Instead of hiring a team of engineer to keep the context model of an augmented environment up to date, the environment itself must be adaptive. The research challenge is to develop machine learning methods for this process, making it possible to automatically acquire and evolve context models reflecting user behavior and needs in an intelligent environment. Two different motivations for learning context models are distinguished: Knowledge engineering (acquisition of a context model) and User preferences (adaptation of a context model). Knowledge engineering refers to building up a context model and context-aware system from collected data. User preferences refer to the updating process, integrating changing user preferences into an existing context model and system using user computer interaction. A user preferences approach can include a knowledge engineering step to set up an initial context model. Several examples for learning context are then presented and compared with regard to their comprehensibility for the user and their adaptivity. A description of the contributions of this thesis to the problem concludes this chapter.

Chapter 3 proposes an abstract model for representing and perceiving context in augmented environments. First, the relationship between context and human activity is discussed. According to the obtained insights, a context model should describe a closed-world motivated by application and user needs, be suitable and intuitive to provide explanations to the user, and focus on human relations and activities rather than environment settings. Following these conclusions, the situation model is defined. A situation is a form of state characterized by a set of roles and relations. Roles involve only one entity, describing its activity. An entity can be a person, place or object considered to be relevant to user and application. An entity is observed to “play” a role. Relations are defined as predicate functions on several entities, describing the relationship or interaction between entities playing roles. A situation network is defined as a composition of situations that share the same set of roles and relations. The situation model can be interfaced with perceptual components of an augmented environment. Recognition processes for roles and relations are proposed. These recognition processes are interpreted as acceptance tests applied to the properties of relevant entities. Acceptance tests for roles are applied to the properties of all available entities in order to determine their role assignment. Acceptance tests for relations are applied to the properties of entities already assigned to roles (“playing roles”). An acceptance test is based on events sent by perceptual components. Events concern entities and their properties. An acceptance test can be divided into two different phases: 1) filtering and 2) roles and relation assignment. A filter is applied to the raw entity event stream in order to extract more abstract events that reduce the dimension of the data. Role and relation assignment takes

these abstract events as input and generates the corresponding roles and relations. An example implementation of acceptance tests for the lecture scenario is presented.

Chapter 4 details two possible "top-down" implementations of the situation model and gives an example for each implementation. Situations and the underlying abstract concepts can be interpreted as finite-state machines. The finite-state machine implementation influences the control flow and how perceptions, coded as events, are finally used and interpreted to activate situations. First, a deterministic implementation of situation models is presented. This implementation is based on Petri nets implementing the situations of the model. The implementation has been used to implement an automatic cameraman. The automatic cameraman is context-aware selecting at every time, based on the current situation the appropriate camera to provide images. The automatic cameraman has been successfully evaluated. A probabilistic implementation of situation models is further proposed. This implementation is based on hidden Markov models. The states of the hidden Markov model represent the situations. The implementation has been used to implement a real-time detector for interaction groups. The objective was to detect the split and merge of small groups during a meeting. The proposed detector has been successfully evaluated. The choice of the implementation depends on the application that is envisaged. Petri nets can implement all Allen temporal operators, in particular those describing parallelism. However, Petri nets can not model erroneous perceptions and uncertain situations (uncertain expectations of perceptions for a situation and uncertain situation transitions). Hidden Markov models are less rich in modeling temporal constraints (in particular parallelism), but HMMs permit to model erroneous input and uncertain situations.

Chapter 5 explores several methods for learning role acceptance tests. These methods are part of the framework for the automatic acquisition and evolution of situation models. A role acceptance test is divided into a filtering phase and role and relation assignment. The filtering phase is considered to be the key process, isolating abstract events necessary to activate a role. These abstract events fuse and filter event streams coming from perceptual components. The event streams contain the available entities as well as their properties. Three methods for learning different role labels from entity event streams are presented. Role labels refer to the abstract events necessary for role assignment. The proposed methods are based on a Bayesian classifier, support vector machines (SVMs) and a hybrid classifier combining Bayesian methods and SVMs. The three methods have been evaluated on data sets recorded in an augmented home environment. The data sets were based on events created by a video tracking system and contained five different role labels: "walking", "standing", "sitting", "interaction with table", and "lying". SVMs outperformed the Bayesian classifier when recognizing these role labels. In order to measure the performance to detect unseen role classes, each role label has been excluded once from the learning data. SVMs, Bayesian classifier and hybrid classifier have been compared with regard to the identification of the data associated to the excluded role label. The hybrid classifier outperformed the Bayesian classifier and the SVMs when identifying unseen roles, showing that the proposed combination of generative and discriminative methods is beneficial.

Chapter 6 proposes a method for the unsupervised extraction of situations from multimodal observations. The method is part of the framework for the automatic acquisition and evolution of situation models. The unsupervised situation discovery has been applied to the field of automatic analysis of small group meetings. The proposed method detects changes in small group configuration and activity based on measuring the Jeffrey divergence between adjacent histograms of observations. In [18], the authors showed that different meeting activities, and especially different group configurations, have particular distributions of speech activity. This can be extended to distributions of multimodal observations coming from multi-sensory input. These distributions are represented by histograms containing the frequency of these observations. To separate distinct distributions of observations, two adjacent windows are slid from the beginning to the end of the meeting recording, while constantly calculating the Jeffrey divergence between the histograms generated from the observations within these windows. The size of the sliding adjacent windows is varied generating several Jeffrey divergence curves. The peaks of the resulting curves are detected using successive robust mean estimation. The detected peaks are merged and filtered with respect to their height and window size. The retained peaks are finally used to select the best model, i.e. the best allocation of observation distributions for the given meeting recording. The method has been tested on five short small group meeting recordings, a seminar recording and a cocktail party meeting recording. The short small group meeting recordings and the seminar recording were based on audio, while the cocktail party meeting recording included audio and video. The obtained segmentation results are very good; the audiovisual segmentation of the cocktail party meeting outperforms the pure video and pure audio segmentations.

Chapter 7 addresses the supervised learning of situation. The proposed algorithm is part of the framework for the automatic acquisition and evolution of situation models. Supervised situation learning is based on the segments of observations extracted by unsupervised situation discovery for each situation. Each segment corresponds to one situation. An expert or user provides situation labels for each of these segments. Two or more segments can have the same situation label. The notion of learner is introduced as a learning method that generates a situation representation for a given segment of observations. Examples of learner classes are expectation-maximization(EM) algorithm (with hidden Markov models as representations), or ID3 (with decision trees as representations). An instance of the learner class corresponds to a specific parameterization of the learner (e.g. the number of states for the HMMs to be learned). The proposed situation acquisition algorithm produces or learns a representation for each situation from given segments and the associated situation labels by iterating over possible learner classes and learner instances. The objective is to find the representations that are the most discriminative with regard to the given segments and associated situation labels, i.e. that maximize the ratio of between-situation distance and within-situation distance (Fisher's criterion). The proposed supervised situation learning scheme is general and can be adapted to many different learners and applications. The proposed algorithm has been applied to a video surveillance task. The CAVIAR video clips have been used for evaluation and show different situations: "walking",

"browsing", "fighting", "waiting" and "object left". Each video is associated with an XML file describing for each frame the entities and their properties. These files have been created manually. The expectation maximization algorithm has been used as learner class to create a HMM representation for each situation based on the observations in the XML files. The obtained results for situation recognition based on the learned representations are good.

Chapter 8 explores a method for evolving an initial situation model with user feedback. The proposed method is part of the framework for the automatic acquisition and evolution of situation models. The initial situation model can either be predefined by a human engineer or be constructed automatically by the methods proposed in chapter 6 and 7. The learning process must then adapt the situation model according to feedback given by the user on executed system services. The proposed algorithm focuses on the adaptation of the situations as well as the situation network and the associated system services. The input to the algorithm is an initial situation network along with feedback from prior use. The feedback concerns the correction, deletion and preservation of system services. The proposed algorithm tries first to adapt the system services directly by changing the association between situations and services. In a second step, if the feedback indicates that the concerned situation is too general, the algorithm splits the situation into sub-situations. The determination of the characteristic observations describing the sub-situations is interpreted as classification problem. The service labels of the training examples correspond to the class labels. The supervised learning scheme of chapter 7 is then used to learn these class labels (corresponding to the sub-situations) from observation sequences. The proposed algorithm has been tested in an augmented office environment. An initial situation network describing office activity has been defined by a human engineer. Two new services (turn-on and turn-off of the music player) have been correctly integrated into the situation model based on supervisor feedback. Conceptual learning algorithms Find-S and Candidate Elimination as well as decision tree algorithm ID3 have been used as learner classes (in the supervised learning scheme) in order to learn the new sub-situations.

Chapter 9 describes the implementation and evaluation of the complete framework for acquiring and evolving situation models. The proposed methods for role recognition, unsupervised situation discovery, supervised situation learning and integration of user preferences have been integrated into an augmented home environment. This system is to build up automatically and to evolve a situation model for human behavior in the scene. The perceptions of the system are based on a 3D visual tracking system as well as speech and ambient sound detection. Users wear head set microphones in the environment. Role detection is conducted on the target properties provided by the 3D tracker. Multimodal observations are generated based on role, speech and ambient sound detection. An initial situation model for the behavior in the scene is constructed using the segmentation of basic situations and the supervised learning of situation labels. The resulting initial situation model is then evolved according to user preferences using feedback. Two different evaluations have been conducted. A first evaluation concerned unsupervised situation extraction and supervised situation learning. Several small scenarios showing different

situations like “presentation” or “siesta” were recorded. The recordings have been presegmented automatically. The recognition rate of the situations in the scenarios with and without automatic presegmention has been investigated. A second evaluation concerned the combination of the three methods: unsupervised situation extraction, supervised situation learning and integration of user preferences. Therefore, 3 (longer) scenarios were recorded showing several situations like “aperitif”, “playing game” or “presentation”. The recordings have first been automatically segmented. Then, the extracted segments have been labeled and the situations have been learned. Finally, the learned situation model has been evolved with user feedback. The recognition rate of the labeled as well as of the added situation (via situation split) has been investigated. The obtained results validate the approach for a framework for automatic acquisition and evolution of situation models.

Chapter 10 summarizes and concludes this thesis. The main contributions of this thesis are described including probabilistic implementation of situation models, role learning and detection, supervised learning scheme and situation split.

Chapter 2

Definition and Analysis of the Problem

This chapter aims at defining the problem of this thesis with respect to the literature. First, the notion of augmented or "intelligent" environment is defined and illustrated by several examples. The key concepts of context-awareness and proactive system behavior are motivated and discussed. Then, the problem of learning context models is explained. Several existing learning approaches are compared with respect to their intelligibility and adaptability. A short overview of the contributions of this thesis to the problem concludes this chapter.

2.1 Augmented "Intelligent" Environments

In the 1980s and early 1990s, Xerox PARC researcher Mark Weiser developed the concept of ubiquitous computing, following the principle that:

> "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." [94]

The integration of computing devises into every-day environments has been one of the predominant trends over the last decade. Cell phones, PDAs and laptop computers as well as WLAN networks have become part of almost every household. This trend enables *computer-everywhere* environments. These environments are augmented with multiple sensors and interaction devices. Coen [34] defines the term of "intelligent" environments as "spaces in which computation is seamlessly used to enhance ordinary activity". The objective is to make computers not only user-friendly but also invisible to the user. Interaction with them should be in terms of forms that people are naturally comfortable with.

One of the very first intelligent environments, the *Intelligent Room* [33], has been realized at MIT AI Laboratory. The Intelligent Room is laid out like an ordinary conference room, with a large table surrounded by chairs (Figure 2.1). Mounted at various places in the conference area are twelve video cameras, which are used by computer vision systems. Two video projectors, several video displays as well as audio devices and wireless microphones further augment the environment. The objective of the Intelligent Room was to experiment with different forms of natural, multimodal human-computer interaction (HCI) during what is traditionally considered non-computational office activity. Numerous computer vision, speech and gesture recognition systems are used to detect what inhabitants are doing and saying.

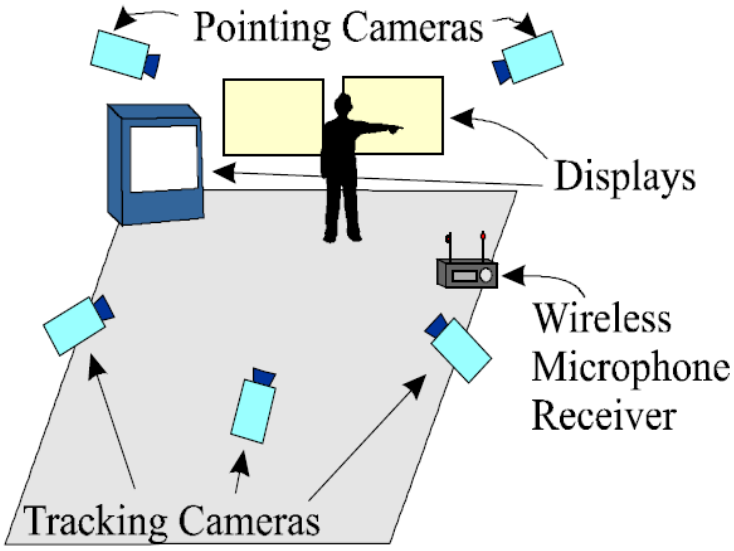


Figure 2.1: A simplified layout of the Intelligent Room at MIT AI Lab (picture from [34])

A similar office environment has also been developed at INRIA Rhône-Alpes. The SmartOffice [60] comprises a whiteboard area and a large office desk completed with a computer workstation in the center of the room. 50 sensors (cameras and microphones) and three actuators (a video projector and two speakers) are installed within the environment. The MagicBoard [53] is the main “actuator” for the SmartOffice, letting users combine digital and physical information on the whiteboard. Mobile and wide-angle cameras permit the use of computer vision recognition systems. Eight microphones distributed across the ceiling are used for speech recognition. The objective was to monitor the user in order to anticipate user intentions and to augment the environment in order to communicate useful information.

At XRCE, an intelligent workplace environment has been realized [6]. The intelligent workplace environment is laid out like a normal individual workplace, comprising a desktop computer, a PDA device and an office telephone. The environment is augmented with PC and phone usage sensors, PDA location and ambient sound sensors as well as a PDA user feedback form. The

PDA form was used by the users to give feedback on their current office activity. The objective of the intelligent workplace at XRCE was to sense individual office activity and to provide sensed information to other users (e.g. in order to derive possible availability).

Mozer [67] developed one of the first intelligent home environments at the University of Colorado. The Adaptive House has been implemented in an actual residence that was renovated in 1992, at which time the infrastructure needed for the project was incorporated into the house. The home laboratory is equipped with an array of over 75 sensors which provide information about the environmental conditions that are monitored – temperature, ambient light levels, sound, motion, door and window openings – and actuators to control the furnace, space heaters, water heater, lighting units, and ceiling fans. The objective of the Adaptive House was to make life more comfortable for inhabitants and to conserve energy at the same time. By using inferred occupancy and usage patterns in the home, the Adaptive House was to adjust automatically room heating, water heating and room illumination. Explicit Sensing and recognition of human activities in the house was not the focus of the Adaptive House Project.

The EasyLiving Project [22] at Microsoft Research was concerned with the development of an architecture and suitable technologies for intelligent home environments. The focus of EasyLiving laid on technologies for middleware (to facilitate distributed computing), geometric world knowledge and modeling (to provide location-based context), perception (to collect information about environment state) as well as service abstraction and description. Input devices can include an active badge system, cameras, wall switches, and sensitive floor tiles. Output devices can include home entertainment systems, wall-mounted displays, speakers, and lightening. Stereo computer vision tracking is used to derive the location of people in the environment as well as to maintain their identity while they are moving around. Radio-frequency (RF) wireless-LAN-enabled mobile devices are located based on the signal strength of known infrastructure access points. A geometric world model is used to derive the spatial relationship between entities in the environment. The location is used to infer a person’s intent or activity based on his or her position. The objective of EasyLiving was to enable typical PC-focused user activities to move from a fixed desktop into the environment as a whole. Several intelligent space applications like movable desktop sessions or location-based media control have been implemented.

The MavHome Project [35] developed a smart home environment at the University of Texas at Arlington. The MavHome acts as an autonomous intelligent agent that perceives its environment through the use of sensors, and can act upon the environment through the use of actuators. Perception is managed through light, humidity, temperature, smoke, gas, infrared motion, and switch sensors deployed in the environment. Main actuators are the control of lightening and blinds, water heater, different video and screen displays, sprinkler and VCR. Location-based media control and tracking is also provided. The objective was to manage the home automatically in a way that maximizes productivity and comfort of its inhabitants, minimizes the costs of operating the home, and ensures the maximum security of the home and collected/personal

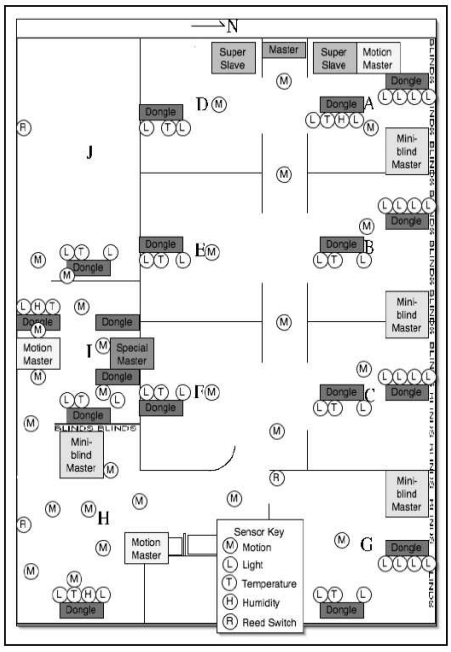


Figure 2.2: Layout of the augmented home environment *MavHome* at University of Texas at Arlington (picture from [99])

data.

At MIT Media Lab, Bobick et al. [16] constructed an early example of an intelligent entertainment environment. The KidsRoom was a perceptually-based, interactive, narrative playspace for children. The environment, which resembles a children's bedroom, uses two large back-projected video screens, four speakers, theatrical lightening, three video cameras, and a microphone array to perceive and to interact with the children. Computer-vision algorithms on the video images of the scene are used to identify the activity of several children. Constant tracking of the positions of up to four children and a strong story context are used to limit the number possible children's activities. Images, music, narration, light, and sound effects generated by the system guide the children through the story. The strong story context defines the possible children's activities at the actual state of the play and the appropriate reactions to be taken by the system. The objective of the KidsRoom was to explore the design of interactive spaces and to develop suitable computer vision techniques.

The eClass Project [2] (formerly known as Classroom 2000 Project) concerned the development of an intelligent education environment at Georgia Tech. The project constructed a prototype classroom environment and the necessary software infrastructure to seamlessly capture much of the rich interaction that occurs in a typical university lecture. The classroom is augmented with single audio-video stream recording facilities, electronic whiteboards, and personal pen-based

interfaces. Further, software and WWW access facilitate automatic capture and content-based access of multimedia information in the educational setting. The objective of the eClass Project was to automate the capture of individual and group activity in the classroom and to provide an easily accessible interface that integrates this information together.

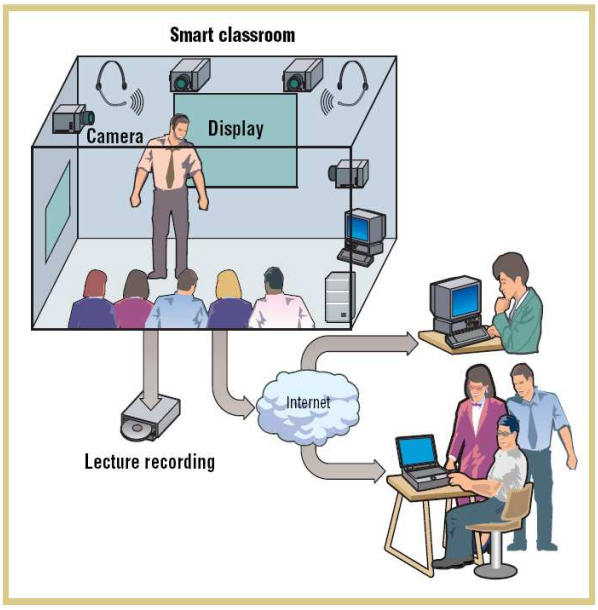


Figure 2.3: Overview of the *Smart Classroom* system at Tsinghua University (picture from [88])

The Smart Classroom Project [98] constructed an intelligent classroom environment at Tsinghua University. The augmented classroom has two wall-size projector screens, one on the front wall and the other on a side wall, and several cameras that are deployed in the environment (Figure 2.3). Additional cameras are installed on the computers of remote students. The teacher wears a wireless headset microphone to capture his or her speech. A touchsensitive board further enhances the room. Voice-recognition, computer vision techniques and activity recognition are used to permit the simultaneous instruction of local and remote students. The objective of the Smart Classroom Project was to seamlessly integrate tele-education and traditional classroom activities. The system turns a physical classroom into a user interface for tele-education.

This section has presented different examples of intelligent environments in the domains of workplace, housing and education. These augmented environments involve various research disciplines, ranging from computer science, over social science to psychology. In computer science, ubiquitous or pervasive computing [83] integrates computing into these environments, nomadic computing [58] mobilizes computing devices, and ambient intelligence [86] helps making these environments smart(er). In the field of ambient intelligence, we consider sensing and responding to human and environmental context to be a key feature for achieving augmented

intelligent environments. The following section defines and discusses the terms *context* and *context-awareness* as well as *proactive* services that context-aware systems supply.

2.2 Context-Aware Services: towards unobtrusive, proactive system behavior

In 1996, Weiser and Brown [95] introduced the notion of *calm technology*, described as:

> "If computers are everywhere they better stay out of the way, and that means designing them so that the people being shared by the computers remain serene and in control."

Computing systems should "stay out of the way", while providing useful and enriching services. In the context of smart environments and smart artifacts, Streitz et al. [89] distinguish two types of service behavior: system-oriented, importunate smartness and people-oriented, empowering smartness. System-oriented, importunate smartness enables the environment to take certain self-directed actions, while people-oriented, empowering smartness focuses on empowering the users to make decisions and take responsible actions.

System-oriented, yet unobtrusive smartness constitutes a major challenge as it addresses two important issues:

1. sensing, and recognizing user behavior, needs and intents,
2. while keeping the user informed and in control.

The system services are to be supplied without interrupting the user's current task and activity. In addition, they should be predictable for the user (principle of least surprise [7]). These services will not replace human-computer interaction itself because depending on the complexity of the current task of the user, deriving user behavior, intent, or needs may be too difficult. The main purpose is to reduce the communication workload of the user when working on his tasks. Obviously necessary actions may be automated and so the user can concentrate on essential work and human-computer interaction tasks.

The automatic supply of system services is addressed by the term *proactive system behavior*. Salovaara and Oulasvirta [82] outline that:

> "... the concept proactive refers to two critical features of a system: 1) that the system is working on behalf of (or pro) the user, and 2) is taking initiative autonomously, without user's explicit command."

Proactive systems are thus acting on their own initiative on behalf of the user. Schmidt [86] extends this notion to implicit human computer interaction (iHCI). iHCI is the interaction of a human with the environment and with artifacts which is aimed to accomplish a goal. Within this process the system acquires implicit input from the user and may present implicit output to the user. Implicit input are actions and behavior of humans, which are done to achieve a goal and are not primarily regarded as interaction with a computer, but captured, recognized and interpreted by a computer system as input. Implicit output is not directly related to an explicit input and is seamlessly integrated with the environment and the task of the user.

Actions and behavior of humans, captured, recognized and interpreted by a computer system are the input and basis for iHCI. The computer systems must hence be aware of what the humans are doing in the environment and determine the context in which human actions take place. This issue is normally addressed by the term context-aware computing.

The word context is composed of "con" (with) and "text" and refers thus to the meaning that must be inferred from adjacent text. Winograd [97] refers to context as a shared reference frame of ideas and objects that are suggested by a text. Context is a consensual space, called "common ground" [31], that establishes a framework for communication based on shared experience. Such a shared framework provides a collection of roles and relations with which to organize meaning for a phrase.

Schilit and Theimer [84] first defined the term context-awareness by:

> "location information [that] enables software to adapt according to its location of use, the collection of nearby people and objects, as well as the changes to those objects over time"

This definition is particularly useful for mobile computing applications. An example is the context-aware tourist guide system proposed by Cheverst et al. [27]. The system combines mobile computing technologies with a wireless infrastructure to present visitors to the city of Lancaster (UK) with information tailored to both their personal and environmental contexts.

However, more complex context-aware computing applications need to be built on notions of context that encompass more than only location information [85, 87]. Pascoe [72] defines context as a subset of physical and conceptual states of interest to a particular entity. This definition has sufficient generality to apply to a system that recognizes human actions and behavior. Dey [41] reviews definitions of context and provides a definition of context as any information that

characterizes a situation related to the interaction between humans, application and the surrounding environment. Situation refers here to the current state of the environment. Context specifies the elements that must be observed to model a situation. An entity refers to a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [3, 42].

Recent definitions of context-awareness go even further by defining context as part of a never ending evolution process of interaction in an augmented environment. Coutaz et al. [38] observe hence that:

> "Context is not simply the state of a predefined environment with a fixed set of interaction resources. It's part of a process of interacting with an ever-changing environment composed of reconfigurable, migratory, distributed, and multiscale resources."

Social and interactional aspects of context must also not be neglected. Individual behavior may not be the correct unit of analysis. Evidence shows that about 40 % of the variance in human behavior may be attributable to non-linguistic social context [73]. Dourish [44] further highlights the interactional nature of context. Contextuality is a relational property that holds between objects and activities. Context is particular to each occasion of activity or action; the scope of contextual features is redefined dynamically. Context is not just a fixed part of the environment, but context arises from human activity.

Some scientists claim, however, that context-awareness in real-world applications is simply impossible. An exhaustive enumeration of the set of existing contextual states of the environment seems difficult [52]. Lueg [63] even states that "context-aware artifacts are far from being able to recognize situation". Further, we cannot always know which information determines a specific contextual state. As consequence, determining which appropriate action should be taken by the system autonomously seems impossible [52]. Erickson [47] summarizes that "computers are good at gathering information, humans are good at recognizing context and determining what is appropriate". Human should hence be kept in the control loop, and context-aware computing should rather do visualization of contextual information than recognition and reasoning on human intentions.

One strategy to respond to these critics is to provide feedback to the user about the reasoning of the system. Cheverst et al. [29] propose the term comprehensibility to suggest that the user "can look through the outer covering (e.g. glass box) to examine the inner workings of the device". The motivation is that users fear the lack of knowledge of what some computing system is doing, or that something is being done 'behind their backs' [1]. Bellotti and Edwards [12] go further by defining the term intelligibility by:

> "Context-aware systems that seek to act upon what they infer about context must be able to represent to their users what they know, how they know it, and what they are doing about it."

The term scrutability further refers to the ability of a user to interrogate her user model in order to understand the system's behavior. Kay et al. [57] describe this process as:

> " ...when the user wants to know why systems are performing as they are or what the user model believes about them, they should be able to scrutinize the model and the associated personalization processes."

The issue of comprehensibility and scrutability is closely related to the issue of control over the system. Kay et al. [57] see scrutability as a foundation for user control over personalization. The user needs to be able to understand what the system is doing, and also be able to overrule system decisions and processes if necessary. Cheverst et al. [28] mention the obvious motivation: people often want to perform a non-standard action in a given context. Figure 2.4 relates scrutability and system/user control in a two-dimensional design space. Different augmented environments approaches, presented in section 2.1, are plotted into this design space.

Barkhuus and Dey [10] report on findings from a study in the context of a mobile scenario, where they investigated the relationship between user control and service automation. In their particular setting, mobile phone users were willing to give up control in exchange for services such as tracking the location of friends or recommendation of nearby restaurants at lunch time. Although their study relied on the participants to imagine their usage patterns if such a service was available, one of their main findings was that users were willing to give up control if the benefits (i.e. the convenience or added value) of doing so was high.

If users are willing to give up control for a number of system services, this implies that they trust the automation process of the system. Muir [69] shows that the usage of an automated and context-aware system will be optimal if the user's trust corresponds to the objective trustworthiness of the system. Trustworthiness refers here to the system reliability. This process is called *calibration of trust* [59]. Figure 2.5 illustrates the problem. Human trust and trustworthiness of the system should ideally cover the same range of services. When trust exceeds system capabilities, this leads to misuse of the system (overtrust). When human trust is lower than system capabilities, this leads to disuse of the system (distrust). Recent user studies indicate further that perceived system usability has a significant effect on user trust in a system [61].

One might also want to consider the cost and consequences of automated action execution. In particular, if we trust in a system, and the system is said to be reliable, on which basis system decisions for actions executions need to be done ? This depends, of course, on the action to

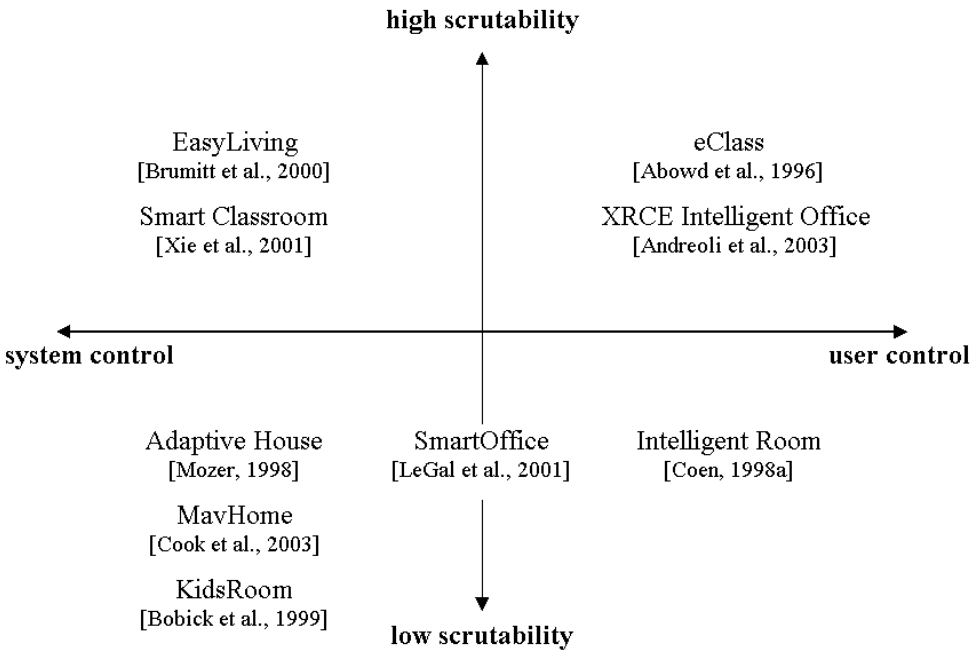


Figure 2.4: Two-dimensional design space spanning control and scrutability dimensions (adapted from [29])

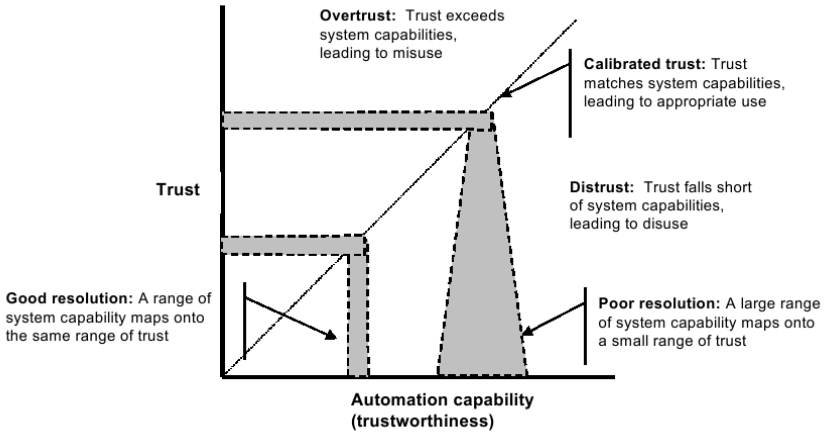


Figure 2.5: The relationship among calibration, resolution, and automation capability in defining appropriate trust in automation. Overtrust may lead to misuse and distrust may lead to disuse (taken from [59])

be automated and the criticity of the environment. For a hospital environment, Bardram et al. [9] summarize that “the triggering of a context-awareness action depends upon the accuracy of the sensed context information, the degree to which you know what action to take in a certain situation, and the consequence of performing this action.”

The automating companion agent is still not an everyday life experience. Sensing, recognizing human behavior, and automating services are currently research issues. This thesis will focus on the modeling and sensing aspects of context-awareness: how to model, and in particular how to acquire and adapt context models, which will constitute the backbone of (future) automated service supply.

2.3 Learning Context Models

Human behavior evolves over time. New activities and scenarios emerge in an intelligent environment, others disappear. New services need to be integrated into the environment, while obsolete services need to be deleted. To cope with these changes, a context-aware system needs to evolve by adapting its contextual representation of users and environment, i.e. its context model. These adaptations can be done by experts knowing changing user needs and sensor perceptions of the system. If you were very rich, you might consider hiring a full-time team of engineers to customize your intelligent environment and to keep it up to date. Mozer [67] state, however, that an intelligent environment must itself be adaptive. The environment’s intelligence lies rather in its ability to adapt its operation to accommodate the users. The research challenge is to develop machine learning methods for this process, making it possible to automatically acquire and evolve context models reflecting user behavior and needs in an intelligent environment.

We can distinguish two different motivations for learning context models:

1. Knowledge engineering (acquisition of a context model)
2. User preferences (adaptation of a context model)

Knowledge engineering refers to building up a context model and context-aware system from collected data. The acquisition process is data-driven, i.e. based on recorded observations of the environment; adapting or evolving the model is not foreseen. The aim is to discharge a human engineer when setting up and customizing a context model for new users and environment. User preferences refer to the updating process, integrating changing user preferences into an existing context model and system using user computer interaction. A user preferences approach can include a knowledge engineering step to set up an initial context model.

Why is the automatic acquisition and adaptation of high-level context models difficult? From the machine learning point of view, the acquisition and adaptation of high-level models reflecting human behavior is already non-trivial. However, considering the adaptation of context in the framework of a whole system (including different users), the problem becomes very challenging. Learning approaches normally assume that human behavior is coherent and stable over time. Further, the variation of behavior should not be too large between users (allowing to integrate this in user preferences). In practical applications, we saw, however, that humans tend to appropriate new technical devices and environments [43] either by misusing them (e.g. by using coffee cups as entrance badges [86]) or by adapting their own behavior to the behavior of the system. When the system now constantly adapts to human behavior and needs, will user and system be capable of reaching a fix point (stable state problem [28])? And how can the user trust such a system that constantly adapts, making it impossible to foresee its actions (trust in automation problem [59])?

Unfortunately, there is no general solution to these difficult questions concerning the usability of adaptive systems. Our focus here is to propose and illustrate different methods for acquiring and evolving context models and their application.

Several approaches have been proposed concerning the prediction of user behavior from sensor perceptions in the environment. Human behavior is recorded over a longer period of time using different sensors deployed in the augmented environment. Machine learning methods are applied to these sensors recordings in order to build up (and update) a user model. The learned user model can then be used to automate system services. We distinguish approaches that construct a user model from data (pure knowledge engineering) and those that construct *and* evolve such a model (user preferences integration).

Mozer [67] uses different sensor recordings (motion, room temperature, water temperature, illumination) of an intelligent home environment to construct and forecast the states of a user model. Various predictors attempt to derive the current state of the environment, based on sensor recordings, and forecast future states of human behavior in the environment. These predictors have been implemented using neural networks. Based on current and predicted states, the system automates air heating, lighting, ventilation, and water heating from the learned user models. The user models are intended to be constantly adapted to changing user behavior and preferences.

Youngblood et al. [99] propose a layered approach based on prediction, data mining, and decision making components. Based on recordings of different sensors (light, humidity, temperature, smoke, gas, infrared motion, and switch sensors) in an augmented home environment, an Active-LeZi algorithm based component predicts inhabitant actions and behavior. A data mining component then abstracts inhabitant activity to episodes that represent the current user task. A hidden Markov model based decision making component decides which action can be automated based on extracted abstract episodes. Possible automations are the control of lightening and blinds, water heater, different video and screen displays, sprinkler and VCR. The system is

to adapt to changing user preferences and behavior by adapting its learned user models.

Mayrhofer [64] proposes an architecture for context prediction. The aim is to construct clusters from recorded low-level contextual feature data (like bluetooth, microphone etc.) using various clustering methods. These cluster models are then used to predict user behavior, i.e. trajectories within the learned cluster networks. Different methods for clustering and trajectory prediction are evaluated. The proposed approach is intended to be adaptive, i.e. changing user behavior will lead to an evolution of the learned models.

Clarkson [32] describes a wearable system with a video camera and a microphone, capable of distinguishing coarse locations and user situations. Hidden Markov models provide an unsupervised clustering of video and audio data recordings. Different user locations and situations like "home", "at work", or "restaurant" are isolated and then recognized based on this clustering. The approach aims at constructing human behavior models from data; no evolution of these models is foreseen.

Mayrhofer [64] and Clarkson [32] focus on the clustering of human activity observations and the prediction of sequences of human behavior (based on these clusters). These approaches are without any input of supervisor knowledge or user feedback. Thus no human activity recognition and interpretation is done. A comparison with human activity perception is mainly done a posteriori, showing that a correlation between isolated clusters and situation perceived by humans is possible. Mozer [67] and Youngblood et al. [99] focus rather on the association of system actions to sensor perceptions. The aim is the discrimination of different system actions or services with regard to the sensing and prediction of human activity patterns. Human behavior is neither modeled nor recognized. All these approaches neglect the issue of intelligibility because they are completely constructed from sensor data without any human knowledge input (e.g. activity labels). Thus no understandable context model is generated. A user can not understand the model and reasoning process of these systems.

Many approaches for the explicit recognition of human activity have been proposed in recent years. The idea is to learn and recognize predefined human activity labels from recorded sensor data. Most of this work is based on visual information [71, 81, 100] or audio information [18] using statistical models for learning and recognition (in particular hidden Markov models). However, most work does not attempt to acquire a high-level context model of human behavior, but again tries to associate sensor values to (predefined) activity classes. The main focus is laid on classification of basic human activities or scenarios without considering a richer contextual description.

Some approaches attempt though to provide a more detailed analysis and representation of human activities. Bayesian methods, in particular Bayesian networks, seek to model the relationship between pertinent observation variables. In some cases, this can lead to a more understandable representation of human activity.

Muehlenbrock et al. [68] propose a system for office activity learning and recognition based on Bayesian classifiers. The user gives feedback about his current activities and the corresponding availabilities. A naive Bayesian classifier is used to learn user activity and availability from sensor data (keyboard and phone usage, PDA location and PDA ambient sound sensors, and co-location sensors) according to the given feedback. The recognition results can be provided to the users, augmented with location and reasoning details based on the probabilities of the classifier. Although the recognition is mainly based on collected sensor data, the user model can constantly be updated with new sensor data as the users work with the system.

The Lumiere Project [55] at Microsoft Research was initiated with the goal of developing methods and an architecture for reasoning about the goals and needs of software users as they work with software. The realized systems (e.g. the Microsoft Office Assistant) aim at providing assistance to the user according to made observations. At the heart of Lumiere are Bayesian models that capture the uncertain relationships between the goals and needs of a user and observations about program state, sequences of user actions over time, and words in a user's query. The Bayesian models used for modeling and reasoning about user behavior are based on a large amount of recorded example data as well as feedback from experts. The constructed models identify a number of (understandable) pertinent variables (like task history or assistance history), their reasoning is, however, not obvious to the user. The models are automatically adapted to behavior and preferences of the users as they work with the system.

Eagle and Pentland [45] propose a prototype system for sensing complex social systems. The system is based on data recordings from mobile phones. The information collected from the mobile phones includes call logs, bluetooth devices in proximity, cell tower IDs, application usage and phone status (such as charging and idle). A hidden Markov model permits to recognize three different situations (home, work, and other) based on the collected information and time (hour, day of the week). Further, a Gaussian mixture model detects proximity patterns of users and correlates them with the type of relationship. The labels for the model come from user surveys. The results are an estimation of social relationships and networks.

Zhang et al. [102] propose a two-layered framework for modeling and recognizing individual and group actions in meetings. A first layer detects individual actions like "writing" or "speaking" from individual audio and video recordings using hidden Markov models. The second group layer fuses the individual output of the first layer as well as group audio and video features (coming from projector screen and white-board). The output of the second layer are group actions like "discussion", "monologue", "note-taking", or "presentation". The second layer is also based on HMMs and detects the group actions in a supervised manner. An unsupervised segmentation of group actions in the second layer has also been proposed [101]. The objective of approach is the offline analysis of multimodal meeting recordings and the construction of the corresponding models (knowledge engineering). The acquired models are evaluated, but not evolved according to changing user preferences and behavior.

The approaches proposed by Muehlenbrock et al. [68], Horvitz et al. [55] as well as Eagle and Pentland [45] are based on a more elaborated representation of human activity. The relationship between (partially) understandable observation variables is modeled and explicit human activity labels are learned. However, the employed representations of human activity and context are not based on a human understandable model and framework. Learning and reasoning process stay further "in the black box". Zhang et al. [102] propose a framework that acquires two layers of individual and group activity labels, but no adaptation of the model to changing human behavior is foreseen.

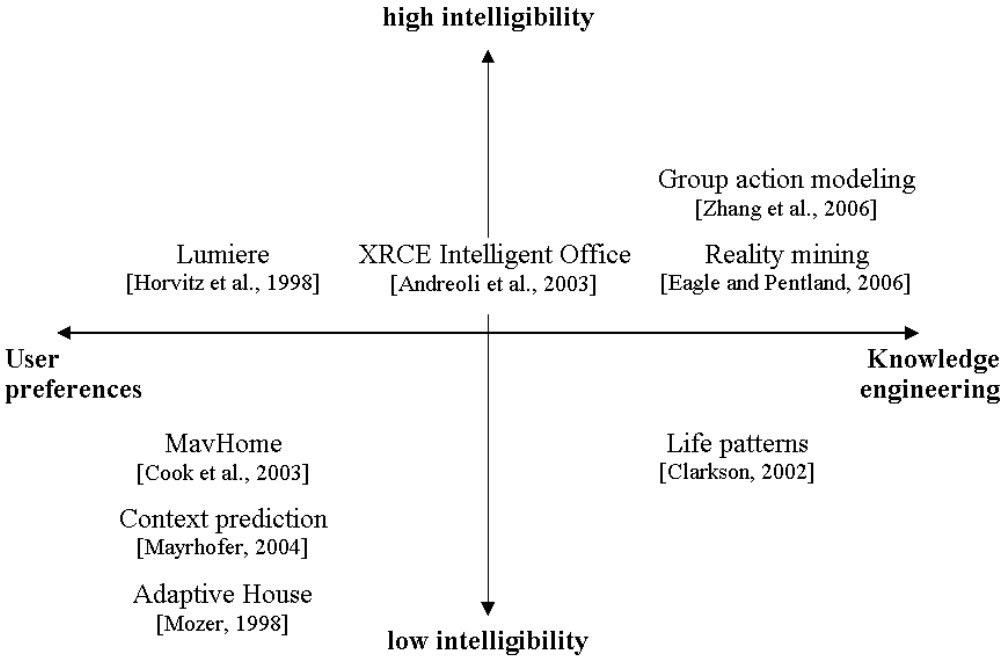


Figure 2.6: Two-dimensional design space spanning adaptivity and intelligibility dimensions

Figure 2.6 summarizes the approaches presented in this section with regard to their intelligibility and adaptivity. Intelligibility refers to the comprehensibility of representation and learning process as well as whether a possible model adaptation step may be understood by the user. Adaptivity refers to our distinction of the motivations of the adaptation process: knowledge engineering refers to the construction of a model from recorded data, while user preferences address the integration of changes into an existing model. Note that a user preferences approach can include a knowledge engineering step (in order to set up an initial model to be evolved).

2.4 Contributions of this thesis

A key requirement for context-aware intelligent environments is to be able to build up and evolve an intuitive, comprehensible context model. This context model needs first to be constructed as automatically as possible using machine learning methods, while staying transparent for the user. The user then should be able to integrate his preferences into this model, and adapt it constantly according to the evolution of his behavior and needs. A context-aware intelligent environment with these capacities would be situated in the upper left corner of the graph in Figure 2.6.

This thesis addresses the issue by providing an intelligible framework for acquiring and evolving a context model, called *situation model*. The methods proposed as part of this framework acquire different layers of the situation model, with different levels of supervision. The situation model serves as frame and support for the different learning methods, permitting to stay in an intuitive declarative framework.

The situation model and the underlying concepts are motivated by the human perception of behavior in an augmented environment. Human behavior is described by a finite number of states, called situations. These situations are characterized by entities playing particular roles and being in relation within the environment.

Situations and the underlying concept of roles are acquired from collected data. Different methods are proposed for learning roles. Further, a method for unsupervised situation discovery is described. A supervised situation learning scheme completes the knowledge engineering part of this thesis.

A method for integrating user preferences into a situation model is also proposed. This method is based on the split of a situation into sub-situations according to feedback from the user, permitting to personalize a constructed situation model.

Chapter 3

Modeling and Perceiving Context - the Situation Model

The aim of this chapter is to propose an abstract model enabling modeling and perception of context in augmented environments. The proposed situation model is based and motivated by the perception of human activity. In the following, we will first discuss the relationship between context and human activity. Then, based on the derived conclusions, we will define our situation model and the underlying concepts. The interface with perceptual components and the necessary recognition processes are further described and illustrated by an example implementation in a lecture room.

3.1 Context and Human Activity

Given its centrality to context-aware computing, the notion of context has been much debated. Some scientists claim indeed that real context-awareness is an intractable problem. Greenberg argues for example that it is difficult or even impossible to enumerate the set of all contextual states that exist [52]. Further, he says that we cannot know which contextual information determines the actual state within this set. Given this lack of precise information, a system cannot state which possible appropriate action to be taken. In addition, Lueg claims that today's context-aware artifacts are far from being able to recognize the situation of a user [63]. Intelligibility and accountability can help alleviating this problem; they become, however, intractable when applied to real-world problems (could we interpret sensor readings and interpretations of an intelligent mobile robot that has not our perception of the world?). Erickson finally argues that computers are good at gathering information, humans are good at recognizing context and determining what is appropriate, so he concludes to keep humans in the control loop and to let

context-aware computing rather do visualization than artificial intelligence [47].

Given these critics, one might think that context-awareness and context-aware applications are seemingly impossible to realize. However, many approaches have been implemented successfully in recent years [1, 19, 27] . What have these successful approaches in common? First of all, none of these approaches attempts to model the whole world with all possible contextual states. Most approaches focus on a specific application in a specific domain, making thus a closed-world assumption. In this closed world, they are able to model all necessary contextual states to provide services to the user. Depending on the application and domain, recognizing these contextual states is reduced to a technical issue like for example object or person recognition. Further, it is much easier to give explanation for these states to the user when we are in a closed world where the number of contextual states is limited.

One important issue is, however, in what detail these context-aware applications can capture the complexity of social context. The majority of documented context-aware applications only use identity and location in their attempts to capture user environment changes [1, 27]. An example is the intelligent tourist guide system by [27] providing visitors with context-aware information and guidance. Even though these approaches produced good results, it has been argued that more complex social situations would require context-aware applications that are built on notions of context relying on more than only location [85, 87]. Schilit et al. [85] observed hence that:

> "Context encompasses more than just the user's location, because other things of interest are also mobile and changing. Context includes lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation; e.g., whether you are with your manager or with a co-worker."

The limitation of contextual cues when modeling human interactions and behavior risks reducing the usefulness of context-aware applications. Of course, again many problems can be considered as technical issues like signal processing problems or person/object recognition. However, the omission of social aspects in context modeling is a considerable problem. Context-aware computing research typically assumes context to be a form of information that is delineable, stable, and separable from activity. Addressing the problem of social aspects in context modeling, Dourish [44] proposes an interactional view of context where context is understood as something relational, dynamic, occasioned, and arisen from human activity. Thus context is not something that describes a setting or configuration, but it is something that people do.

3.2 Defining Concepts: Role, Relation, Situation and Situation Network

Following the conclusions about context and human (section 3.1), a context model should:

- describe a closed-world motivated by the application domain and the needs of the user,
- be suitable and intuitive to provide explanations to the user,
- focus on human relations and activities when describing context rather than environment settings.

How can context be modeled in context-aware approaches?
Dey [41] defines context as "any information that can be used to characterize the situation of an entity". An entity can be a person, place or object considered relevant to user and application. Loke [62] states that situation and activity are, however, not interchangeable, and activity can be considered as a type of contextual information which can be used to characterize a situation. Dey defines situation further as "description of the states of relevant entities". Situation is thus a temporal state within context. Allen's temporal operators [4, 5] can be used to describe relationships between situations. Crowley et al. [39] introduce then the concepts of role and relation in order to characterize a situation. Roles involve only one entity, describing its activity. An entity is observed to "play" a role. Relations are defined as predicate functions on several entities, describing the relationship or interaction between entities playing roles. Acceptance tests determine whether a particular entity plays a role or whether several entities are in relation. These acceptance tests associates roles and relations with relevant entities. In the following, we detail the definitions of role, relation, situation and situation network.

Situations are a form of state defined over observations. A situation is defined using a predicate expression. The logical functions that make up this expression are defined in terms of a set of roles and relations. Situations in the context model are connected by arcs that represent events. Events correspond to changes in the assignment of entities to roles, or changes in the relation between entities.

A context [37, 39] is a composition of situations that share the same set of roles and relations. A context can be seen as a network of situations defined in a common state space. A change in the relation between entities, or a change in the assignment of entities to roles is represented as a change in situation. Such changes in situation constitute an important class of events that we call Situation-Events. Situation-Events are data driven. The system is able to interpret and respond to them using the context model.

The concept of role is a subtle (but important) tool for simplifying the network of situations. A role is an abstract entity that is able to perform certain actions. Roles are “played” by entities within a situation. Assignment of an entity to a role requires that the entity passes an acceptance test. In our framework, the relations that define a situation are defined with respect to roles, and applied to entities that pass the test for the relevant roles.

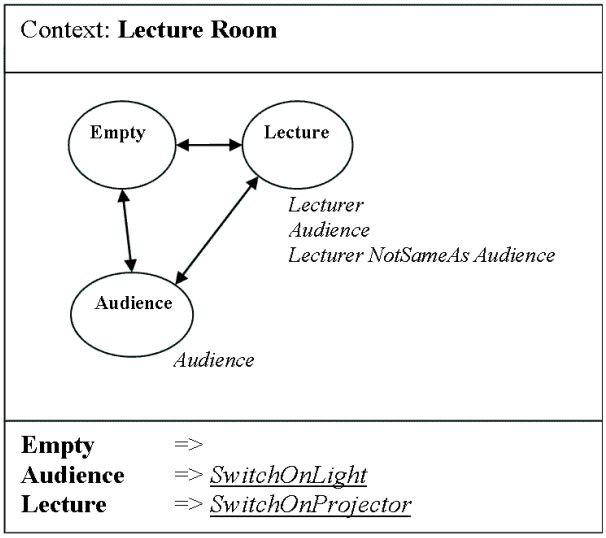


Figure 3.1: Example of a simple context model for a lecture room. **Empty**, **Audience** and **Lecture** are the available situations. *Lecturer*, *Audience* are the available roles and *NotSameAs* the available relation. SwitchOnLight, SwitchOnProjector are system services

For example, in a lecture situation, at any instant, one person plays the role of the “lecturer” while the other persons play the role of “audience” (Figure 3.1). “Lecturer” and “audience” share the “notSameAs” relation, i.e. the entities playing the corresponding roles are different.

The mapping between entities and roles is not bijective. One or more entities may play a role. An entity may play several roles. The assignment of entities to roles may (and often will) change dynamically. Such changes provide the basis for an important class of events: role-events. Role events signal a change in assignment of an entity to a role, rather than a change in situation.

Human behavior within the environment can be described by a script. A script corresponds to a sequence of situations in the situation network reflecting human behavior in the scene. However, scripts in a situation network are not necessarily linear.

3.3 Interface with Perceptual Components

The situation model receives both events and streams from perceptual components (Figure 3.2). A stream is an ordered sequence of entities with properties. To interpret the stream, the situation modeling system must apply a test to the properties to determine which relations and roles apply.

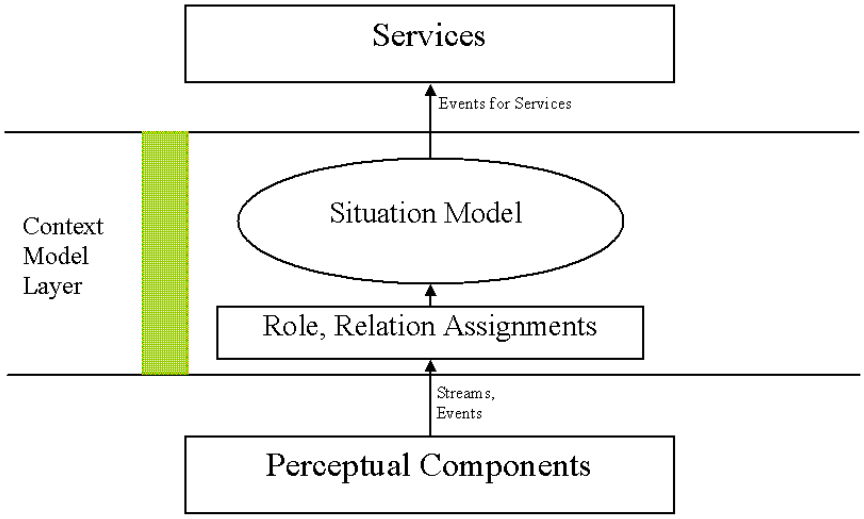


Figure 3.2: Situation Model and its interface with perceptual components and events

Events from perceptual components denote a change in state of an entity. In this case, a logical test for a role or relation has been migrated to, and implemented by, the perceptual component. The situation model needs only note and react to the event.

3.4 Recognition Processes: Acceptance Tests for Roles and Relations

The recognition processes for roles and relations are interpreted as acceptance tests applied to the properties of relevant entities. Acceptance tests for roles are applied to the properties of all available entities in order to determine their role assignment. Acceptance tests for relations are applied to the properties of entities already assigned to roles (“playing a role”). Thus the search space for relation acceptance test is limited by the available entities playing roles (Figure 3.3).

In the first section of this chapter, we describe the general design of recognition processes as acceptance tests based on events. The second section presents an example implementation of

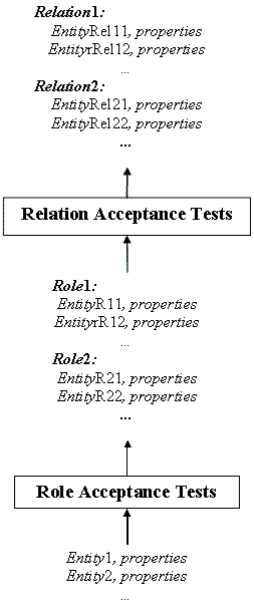


Figure 3.3: Role acceptance tests are applied to all available entities, while relation acceptance tests are applied to entities playing roles

acceptance tests for a lecture scenario.

3.4.1 Acceptance Tests based on Events

An acceptance test is based on events sent by perceptual components. Events concern entities and their properties. Acceptance tests take these events as input and generate role and/or relation assignments of entities as output. Thus acceptance tests are the connectors between the context model layer and the perceptual components.

An acceptance test can be divided into two different phases: 1) filtering and 2) role and relation assignment (Figure 3.4). The filtering phase concerns the raw entity event stream generated by perceptual components. A filter is applied to this stream in order to extract more abstract events that reduce the dimension of the data. The second phase concerning role and relation assignment takes abstract events as input and generates the corresponding roles and relations. The filtering phase can be integrated into perceptual components, while the role, relation assignment is part of the context model layer.

For example, a perceptual component tracking objects in video images may send events of the form (Position *Entity X Y*), where *Entity* denotes the entity identifier and *X*, *Y* the entity's position. To apply an acceptance test, the constant stream of entity events is first parsed for

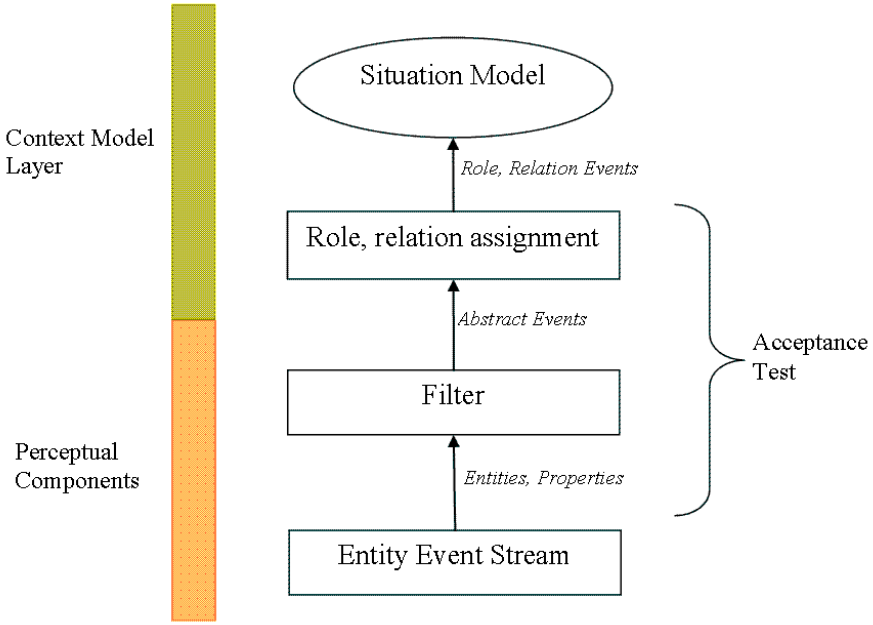


Figure 3.4: Acceptance test as connector between perceptual components and context model

particular entity property values *X*, *Y* (filtering). Abstract events like (Enters *RegionOfInterest Entity*) indicating particular interest zones of entities are sent to role, relation assignment. Roles and relations can then be determined for the available entities using the abstract events from one or several filters.

3.4.2 Example Implementation of Acceptance Tests for the Lecture Scenario

In the Lecture room context example, we are concerned with person-entities and their location in regions of interest. We will assume that a perceptual component tracks moving objects in the video images of the scene (for example by using image background subtraction). The resulting event stream contains the entity identifier as well as the entity position in the video images (Position *Entity X Y*). We assume that this component can determine whether the tracked object is a person or not (for example by using skin color detection or other features). The result is an abstract event indicating whether an entity identifier has been detected as person or not (Person *Entity*). As tracking and detection has to be in real-time, the filtering for regions of interest as well as person detection is integrated into this component (Figure 3.6).

Figure 3.5 shows the filtered interest zones (in violet): one interest zone next to the board

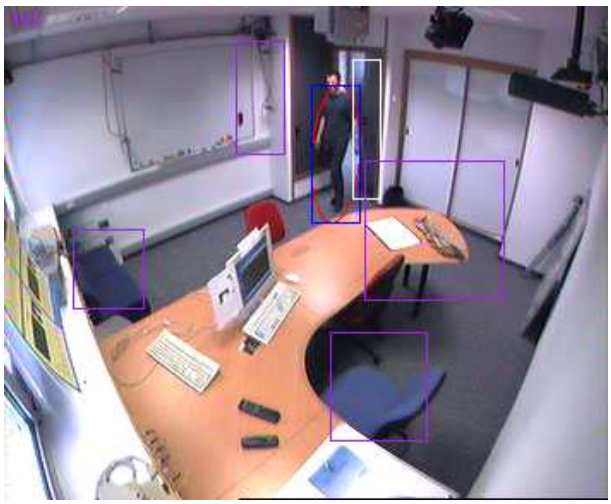


Figure 3.5: Interest zones (violet) for the lecture scenario

("LectureArea"), three zones around the tables and chairs ("AudienceArea"). The perceptual component tracks the entities in the scene. If the entity enters or leaves a region of interest, an abstract event of the form (enters <Region-ID> <Entity>) or (exits <Region-ID> <Entity>) is sent respectively. When the entity leaves the scene a new event is sent to the situation model (ExitScene <Entity>). If the perceptual component looses the entity, an abstract event will be sent of the form: (Lost <Entity>).

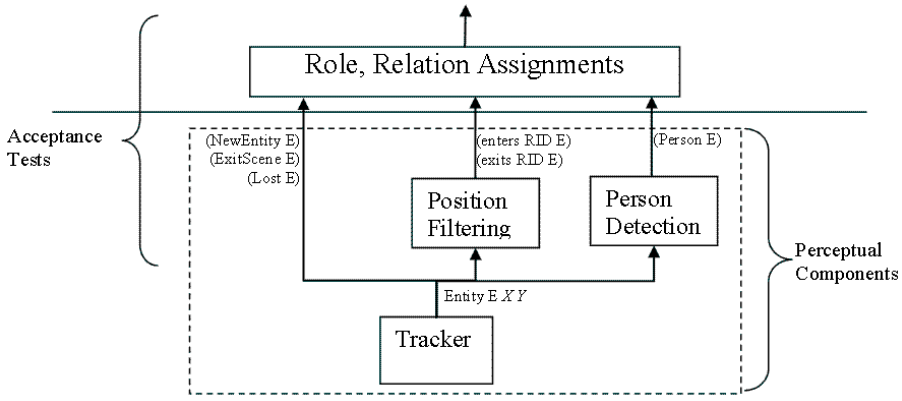


Figure 3.6: Perceptual component for the lecture scenario

Figure 3.1 indicates the necessary roles and relations for the lecture scenario. We can identify the roles *lecturer*, *audience* and the relation *NotSameAs*.

The abstract events that will be sent to the role, relation assignment are:

(NewEntity <Entity>)

(ExitScene <Entity>)

(Lost <Entity>)

(Person <Entity>)

(enters LectureArea <Entity>)

(exits LectureArea <Entity>)

(enters AudienceArea <Entity>)

(exits AudienceArea <Entity>)

Using these abstract events, we can define the following conditions and actions for role, relation assignment:

Role: *Lecturer*

Conditions	Action
{(Person <Entity>), (enters LectureArea <Entity>)}	*Lecturer(<Entity>))*
{(exits LectureArea <Entity>)}, {(ExitScene <Entity>)}, {(Lost <Entity>)}	¬*Lecturer(<Entity>)*

Role: *Audience*

Conditions	Action
{(Person <Entity>), ,(enters AudienceArea <Entity>)}	*Audience(<Entity>))*
{(exits AudienceArea <Entity>)}, {(ExitScene <Entity>)}, {(Lost <Entity>)}	¬*Audience(<Entity>))*

Relation: *NotSameAs*

Conditions	Action
{<Entity>≠<Entity2>}	*NotSameAs(<Entity>,<Entity2>)*
{<Entity>=<Entity2>}	¬*NotSameAs(<Entity>, <Entity2>)*

Note that the condition of the relation *NotSameAs* is only applied to entities that have been selected by the role acceptance tests, i.e. that are “playing roles”. The test for (in)equality between two entities used as the condition for *NotSameAs* can be done by comparing the entity identifiers.

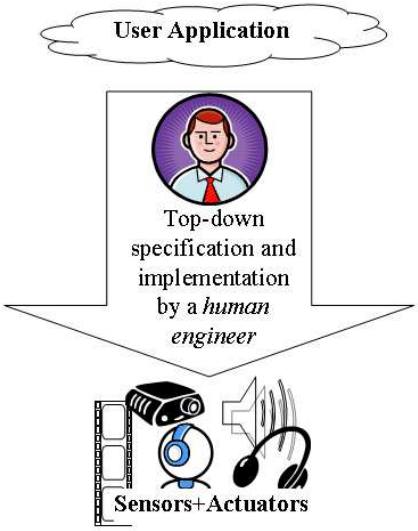
Further note that a role or relation is only activated/deactivated by the respective conditions. If no conditions hold, the last state (“true” or “false”) of the role/relation assignment is maintained. This is important to provide stability to the system.

3.5 Conclusions

In this chapter, we provided an ontology and architecture for modeling context. A finite state machine, the situation model, is constructed according to the needs of contextual and perceptual information of the services. Events and entities are created by perceptual components. Recognition processes are used to assign entities to roles and relations. A recognition process is interpreted as an acceptance tests applied on entities. This acceptance test is based on abstract entity events filtered from entity events streams. Roles and relations can be combined to situations. These situations are connected to form a graph. The fact that the model is a finite state machine provides stability and reduces fluctuations. Services can provide their functionality according to the situation state of the model. A top-down configuration of the perceptual components according to the needs of the services is possible. This top-down filtering adjusts the parameters of the perceptual components using state information from the model as well as information from the services.

Chapter 4

Implementation and Evaluation of Situation Models



This chapter addresses the top-down specification and implementation of situation models. Based on user needs and envisaged application, a human engineer specifies and implements the situation model and associates sensor perceptions and system services to the situations. In the following, a deterministic and a probabilistic implementation of situation models are presented.

Situations and the underlying abstract concepts can be interpreted as finite-state machines. The finite-state machine implementation influences the control flow and how perceptions, coded as

events, are finally used and interpreted to activate situations. We present a deterministic and a probabilistic implementation of situation models. The deterministic implementation is based on Petri Nets, while the probabilistic implementation is based on hidden Markov models. The choice of the implementation depends on the application that is envisaged. An example is given for each implementation in the following sections.

4.1 Deterministic Implementation: Petri Nets

We begin this section with an informal review of Petri nets. We then describe how Petri nets are used to implement a situation network representing a context model and how to evaluate a script.

4.1.1 Petri Nets

A Petri net is a graphical mathematical tool used to model dynamical systems with discrete inputs, outputs and states. Such models have first been defined by Petri [74]. A Petri net is an oriented graph, composed of arcs and two categories of nodes (places and transitions). Places are the state variables of the system containing zero or a positive number of marks. Transitions model the system evolution and are connected *from* places *to* places. A transition is valid if all the "from" places have at least one mark. When a valid transition is fired: one mark is removed from every "from" place and one mark is added to every "to" place. Only one transition can be fired at a time. A more formal definition of Petri nets is given by Murata [70].

Finite-state machines like situation models can be equivalently represented by a subclass of Petri nets [70]. Several extensions of Petri nets have been proposed. One of them is the *synchronized Petri net*. A synchronized Petri net is a Petri net with *events* associated to each transition. A transition can now be fired if it is valid and the corresponding event has been triggered (Figure 4.1).

4.1.2 Implementation and Script Evaluation using Petri Nets

A context model is defined by a network of situations. The connections between the situations are temporal constraints based on Allen's temporal operators [4]. *Events* indicate state changes of the concepts (activity, roles, or relations) describing situations. A situation change is triggered by these events. For instance, consider two situations S1 and S2 such that S2 meets S1 (Figure 4.2). To trigger a change from the current situation S1 to situation S2, we need to observe at least two events:

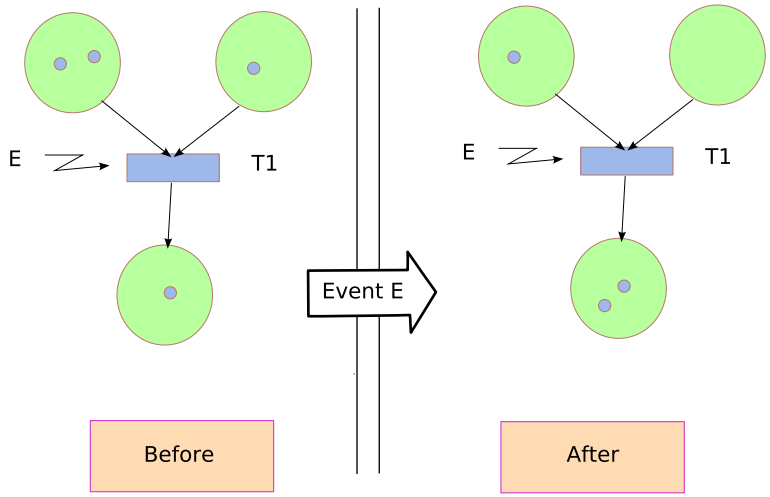


Figure 4.1: Synchronized Petri net with places S1, S2, S3 and transition T1 triggered by event E

1. a first event invalidating the situation S1 ($\neg ValidS1$), followed by
2. a second event validating situation S2 ($ValidS2$) .

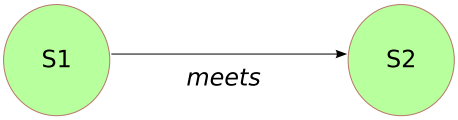


Figure 4.2: A small context : S2 meets S1

This example can be directly represented as a synchronized Petri net (Figure 4.3). We associate a situation to every place. The situation is active if there is at least one mark in the corresponding place: we are currently in situation S1.

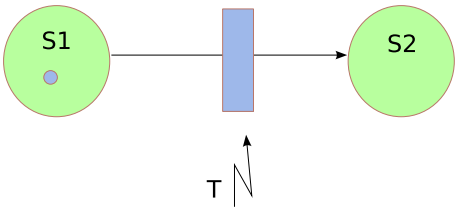


Figure 4.3: Synchronized Petri net implementation of the context

In a “standard” Petri net, the transition between S1 and S2 is automatically fired as soon as S1 is active. We need to control this transition based on the perceptual events coming from the

environment. This transition control is managed using the transition event of the synchronized Petri net. We generate this transition event T only when Situation S1 is no more valid and Situation S2 becomes valid (as seen in subsection 4.1.1):

$$T = \neg\text{ValidS1} \wedge \text{ValidS2}$$

More generally, the event function T of a transition is the conjunction of a function associated to the place before the transition and a function associated to the place after the transition. We will call those two functions respectively $PreT$ and $PostT$. We have:

$$T = PreT \wedge PostT$$

$PreT$ corresponds to what we are *currently observing* in the environment. $PostT$ indicates what the system *should expect* to see next based on the context model.

This short example showed how to transform the "meet" operator into a corresponding synchronized Petri net. This transformation can be done for all the other Allen operators as summarized in Table 4.1 (see [80] for details).

Note that Petri nets are very well adapted for implementing situation models containing parallelism. As shown in Table 4.1, all Allen temporal operators can be implemented by Petri nets. However, Petri nets are not suitable for applications with erroneous perceptions or uncertain perception expectations.

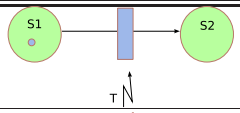
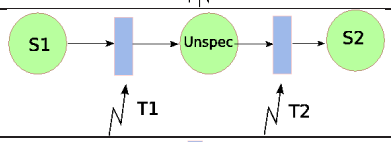
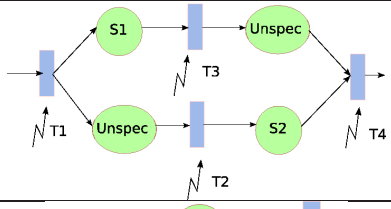
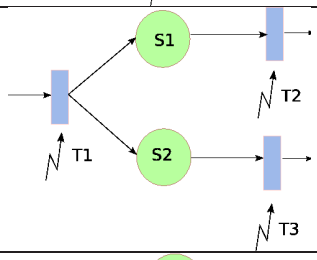
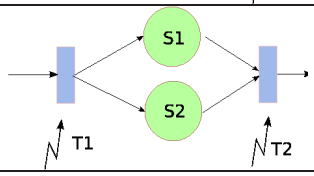
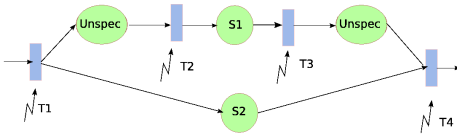
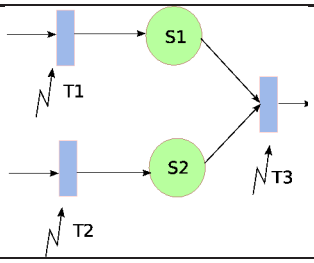
Allen operator	Petri net	Transition function
meets		$\{\ T = \neg ValidS1 \wedge ValidS2$
before		$\begin{cases} T1 = \neg ValidS1 \\ T2 = ValidS2 \end{cases}$
overlaps		$\begin{cases} PostT1 = ValidS1 \\ T2 = ValidS2 \\ T3 = \neg ValidS1 \\ PreT4 = \neg ValidS2 \end{cases}$
starts		$\begin{cases} PostT1 = ValidS1 \wedge ValidS2 \\ PreT2 = \neg ValidS1 \\ PreT3 = \neg ValidS2 \end{cases}$
equals		$\begin{cases} PostT1 = ValidS1 \wedge ValidS2 \\ PreT2 = \neg ValidS1 \wedge \neg ValidS2 \end{cases}$
during		$\begin{cases} PostT1 = ValidS2 \\ T2 = ValidS1 \\ T3 = \neg ValidS1 \\ PreT4 = \neg ValidS2 \end{cases}$
finishes		$\begin{cases} PostT1 = ValidS1 \\ PostT2 = ValidS2 \\ PreT3 = \neg ValidS1 \wedge \neg ValidS2 \end{cases}$

Table 4.1: Synchronized Petri nets for the Allen operators : the “Unspec” place stands for an unspecified situation, which means that we do not model what might happen

4.1.3 Example : Automatic Cameraman

The automatic cameraman [48] is an audio-video recording system that automatically records a lecture. The lecture room is equipped with multiple cameras and microphones. The system is context-aware selecting at every time, based on the current situation, the appropriate camera to provide images (Figure 4.4).

Figure 4.4: Different camera images recorded by the automatic cameraman system

The available actions are *Start recording*, *Filming the whole room*, *Filming the lecturer*, *Filming the audience* and *Filming the slides*. The situation list is :

- Init → start recording and filming the whole room.
- The lecturer speaks → filming the lecturer.
- Someone in the audience asks a question → filming the audience.
- There is a new slide → filming the new slide.
- Someone enters the room → filming the whole room.

The lecture is an alternation of “lecturer speaking” and “audience asking a question”. “New slide” and “someone entering the room” can happen in parallel. The situation network is given in Figure 4.5, the corresponding Petri net is given in Figure 4.6.

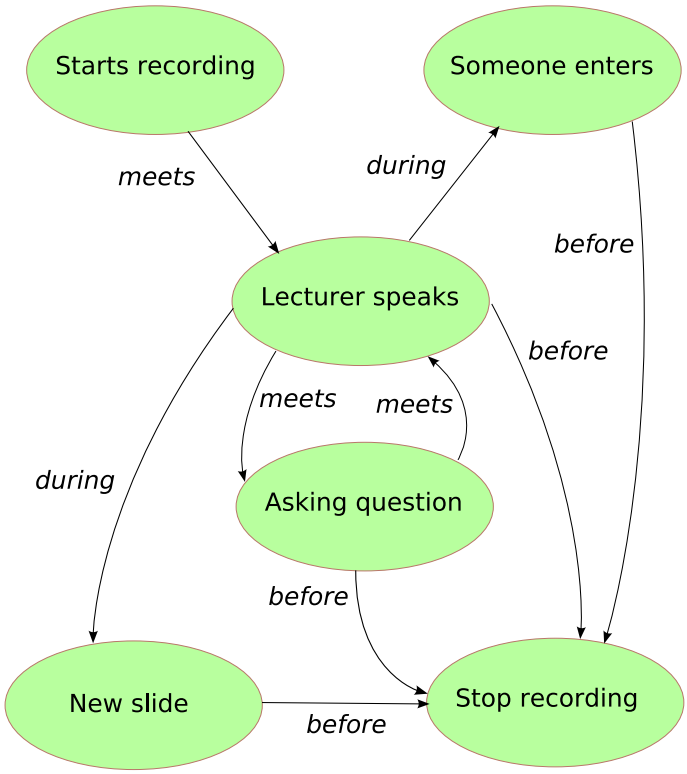


Figure 4.5: Situation network of the automatic cameraman system

Code generation is done by automatically transforming the synchronized Petri net into a corresponding program in Jess (see [80] for details). Jess [56] is an expert system programming environment (facts database with forward chaining rules). The input of the generated program are facts based on events describing state changes of the concepts (roles and relations here)[1]. The output are the current situation and the associated action(s).

This approach recorded a four day seminar on “Language Technology” and “Language, Cognition, and Evolution” [40], which was held on the premises of the FORUM2004 [51] in Barcelona. The automatic cameraman has also been used to record and broadcast real lectures in the amphitheater at INRIA Rhône-Alpes (example images can be seen in Figure 4.4).

[1] Activity, roles, and relations are reconstructed from data provided by perceptual components (video tracker, speech activity detectors, etc.)

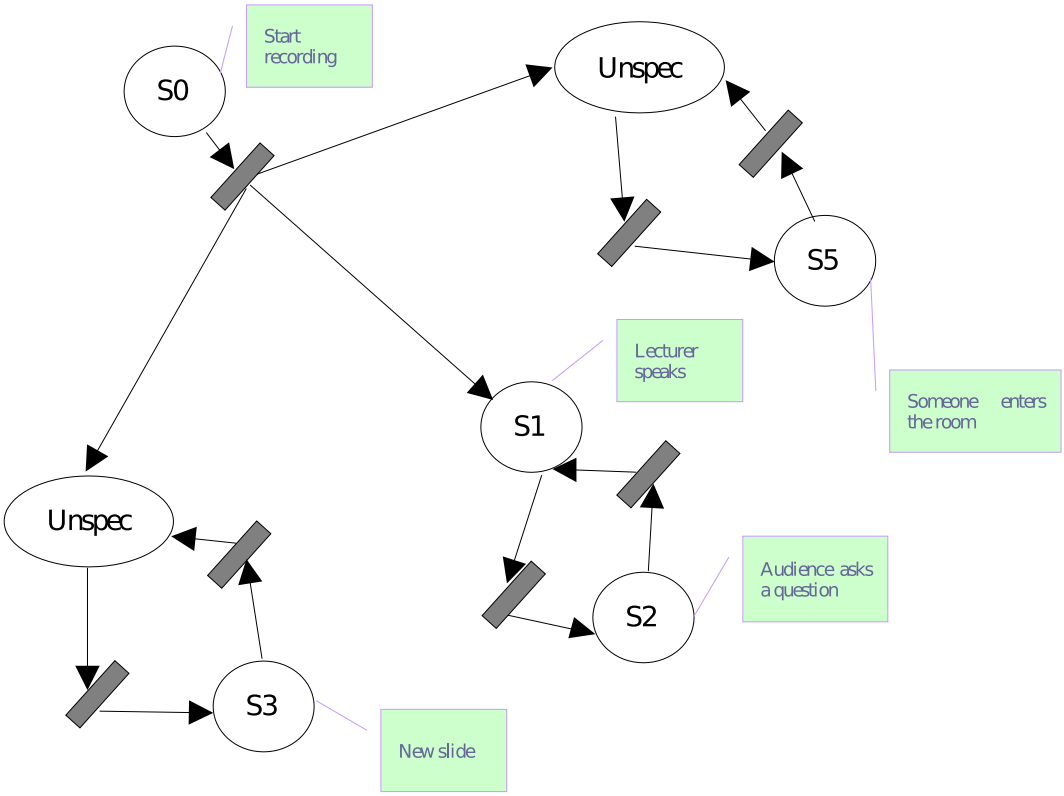


Figure 4.6: Petri net of the automatic cameraman system

4.1.4 Experimental results

Table 4.2 shows the confusion matrix for two hours of recording. A ground truth dataset has been produced by manually annotating the situations on the video. The three situations are :

- *Slides :* a new slide is projected on the screen
- *Speaker :* the speaker is talking or answering a question
- *Audience :* someone in the audience is asking a question

The lines of the matrix contain the detection results, while the columns contain the expected response.

	Slides	Speaker	Audience
Slides	0.96	0.04	0.0
Speaker	0.0	0.89	0.11
Audience	0.37	0.05	0.58

Table 4.2: Confusion matrix

We have obtained a recognition rate of 93.7%. As the context determination is deterministic, this confusion matrix mainly shows the robustness of the underlying perception algorithms (the speech activity and the “new slide” detector). In the last line of the matrix, we can see that there is a big confusion between “audience asking a question” and “new slide”. Indeed, when someone starts answering a question, at the same time, he is often seeking for slides in his presentation. This example illustrates that in some case, deterministic approach can be advantageously replaced by more fuzzy ones.

We have also made an informal survey after broadcasting the lectures in the amphitheater at INRIA. Remote spectators could access two streams : a first stream provided by a fix camera and a stream provided by our automatic cameraman. They all preferred the automatic cameraman stream, allowing them to better understand the presentation.

4.2 Probabilistic Implementation: Hidden Markov Models

A probabilistic implementation of the situation model integrates uncertainty values into the model. These uncertainty values can both refer to confidence values for events and to a less rigid representation of situation and situation transitions.

The situation model is a finite-state machine. The natural choice for a probabilistic implementation is then a probabilistic finite-state machine [93]. A probabilistic finite-state machine is a probabilistic automaton (PFA) defined over a finite alphabet $\sum$. A language is a subset of $\sum^*$. A PFA defines a stochastic language, which is a probability distribution over $\sum^*$. The distribution must verify:

$$\sum_{x \in \sum^*} Probability(x) = 1$$

A formal definition of PFA can be found in [93].

4.2.1 Hidden Markov Models

A hidden Markov model (HMM) [79] is a stochastic process where the evolution is managed by states. The series of states constitute a Markov chain which is not directly observable. Such a chain is said to be "hidden". Each state of the model generates an observation. Only the observations are visible. A more formal definition of an HMM is given by Rabiner [79]. The two following propositions hold [92]:

Proposition 1: Given a PFA A with m transitions and $Probability(\varepsilon) = 0$ (ε-transitions are not allowed), there exists an HMM M with at most m states such that the stochastic language D_M of M is equal to the stochastic language D_A of A.

Proposition 2: Given an HMM M with n states, there exists a PFA A with at most n states such that the stochastic language D_A of A is equal to the stochastic language D_M of M.

As we are only interested in PFA without ε-transitions, i.e. PFA the transitions of which are triggered by events, language-equivalent HMMs can be used to implement PFA.

4.2.2 Implementation and Script Evaluation using Hidden Markov Models

The situations of a context model can be implemented by the states of a HMM. *Events* indicating state changes of the concepts (activity, roles, or relations) generate the observations for the HMM. A state (situation) is characterized by a particular probability distribution of these observations. The activation of a new situation (state) is thus determined by the transition probability from the current state to this new state as well as by the probability of the given observations in this new state. The connections in the situation network are represented by non-zero transition probability values. The observation probability distributions for the situations as well as

the transition probabilities between the situations need to be specified (or learned) when implementing a situation model. We are interested in three basic problems:

1. Given a sequence of observations (based on events) and a situation model implemented by a HMM, how to choose the corresponding state sequence (situation sequence)? This includes the determination of the (most likely) current situation and the determination of likely following situations.

2. Given a sequence of observations (based on events) and a situation model implemented by a HMM, how to compute the probability of the observation sequence, given the model? This corresponds to the likelihood of the situation model (based on the given events).

3. How to adjust the HMM model parameters? This corresponds to adjusting probability distributions of situations based on given event data.

Rabiner [79] gives several solutions to these problems. The Viterbi algorithm is used to determine the most probable state sequence, given a HMM and an observation sequence (Problem 1). The probability of a HMM, given an observation sequence, can be computed using the Forward-Backward algorithm (Problem 2). The expectation-maximization (EM) Baum-Welch algorithm adjusts the HMM model parameters, given observation sequences (Problem 3).

Note that the HMM implementation of a situation model is particularly suitable for applications that deal with erroneous perceptions as well as situations that are characterized by a particular frequency of events. A HMM implementation is, however, less suitable for representing parallelism (not all Allen operators can thus be represented by a classical HMM).

4.2.3 Example : Detection of Interaction Groups

This example addresses the problem of detecting changing interaction group configurations in a smart environment. During a meeting, participants can form one big group working on the same task, or they can split into subgroups doing independent tasks in parallel. Our objective is to determine the current small group configuration, i.e. who is interacting with whom and thus which interaction groups are formed. As we focus on verbal interaction, one group has a minimum size of two individuals (assuming that isolated individuals do not speak). The speech of each meeting participant is recorded using a lapel microphone. An automatic speech detector [48] parses this multi-channel audio input and detects which participant stops and starts speaking. We admit the use of lapel microphones in order to minimize correlation errors of speech activity of different participants, i.e. speech of participant A is detected as speech of participant B.

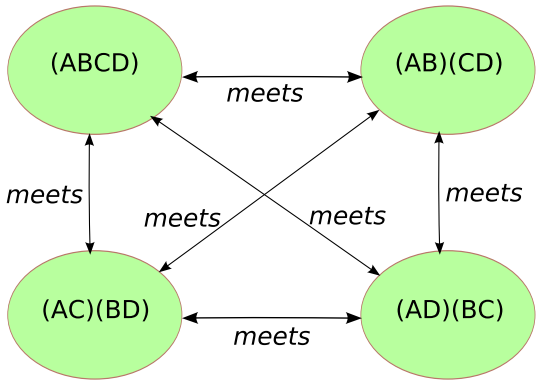


Figure 4.7: Situation model for a meeting of 4 participants A, B, C, D

The proposed approach is based on a HMM implementation of the context model. The observations of the HMM are a discretization of speech activity events sent by the automatic speech detector. Figure 4.7 shows the situation network for a meeting with 4 participants. Each possible interaction group configuration is represented by one situation. These situations are transformed to the states of a HMM (Figure 4.8).

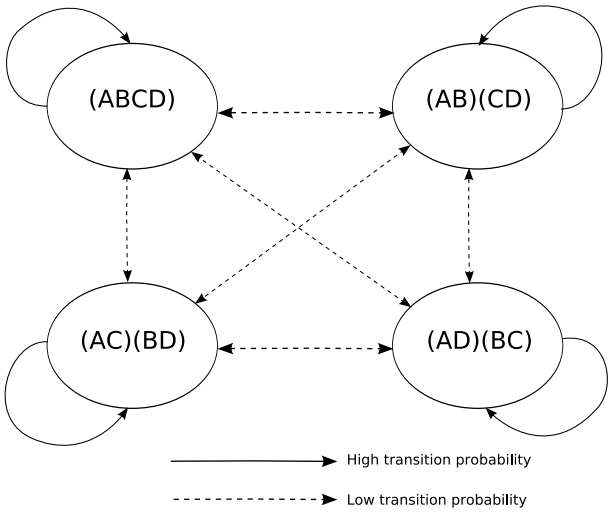


Figure 4.8: States of the HMM implementation of the situation model for a meeting of 4 participants A, B, C, D

The probability distributions of the different states are specified based on conversational hypotheses. These conversational hypotheses assume that speech within an interaction group is more regulated than speech between distinct interaction groups. The transition probabilities between the states are set to a very low level in order to stabilize the detection of state changes assuming hence that group changes occur in reasonable delays. To detect different group con-

figurations, we apply the Viterbi algorithm (solution to Problem 1 in section 4.2.2) to the flow of arriving observations.

Figure 4.9: Example of a configuration of 2 groups of 2 participants

4.2.4 Experimental results

To evaluate, we recorded three small group meetings with 4 participants (Figure 4.9). Using the HMM detector, we obtained a total recognition rate for the small group configurations of 84.8 % [18]. Table 4.3 shows the confusion matrix for the 3 experiments. This matrix indicates for each group configuration the correct and wrong detections. The lines of the matrix contain the detection results, while the columns contain the expected response.

	(ABCD)	(AB)(CD)	(AC)(BD)	(AD)(CB)
(ABCD)	0.88	0.03	0.06	0.03
(AB)(CD)	0.08	0.87	0.05	0.00
(AC)(BD)	0.22	0.01	0.77	0.00
(AD)(CB)	0.04	0.04	0.08	0.84

Table 4.3: Confusion matrix

Figure 4.10 gives an overview of the detection of group configurations over time. The lines of

the chart correspond to different group configurations. The continuous line indicates the correct group configuration expected as detection result.

The results are encouraging and tend to validate the conversational hypotheses to distinguish interaction groups. The Viterbi algorithm executed on long observation sequences is quite robust to wrong detections of the speech activity detector. However, a minimum number of correct speech activity detections is necessary, as the method relies on the information of who speaks at which moment. The use of lapel microphones made it possible to limit wrong detections as these microphones are attached to a particular person (and thus should only detect his/her speech).

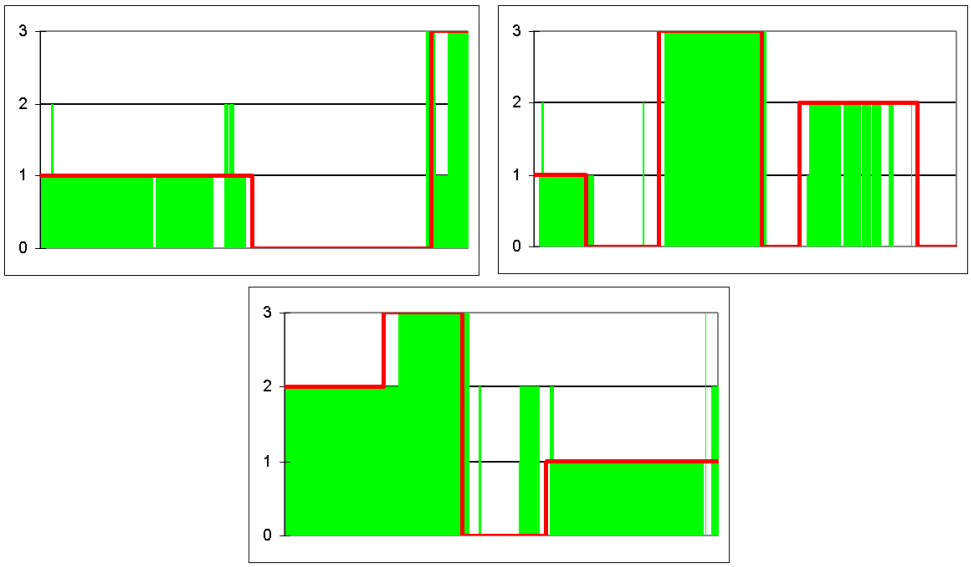


Figure 4.10: Ground truth (red) and detection (green) of group configurations over time for Experiment 1 (upper left, duration: 9 min. 22 sec.), Experiment 2 (upper right, duration: 15 min. 16 sec.) and Experiment 3 (bottom, 16min. 19sec.). On the y-axis, 0 corresponds to the group (ABCD), 1 to (AB)(CD), 2 to (AC)(BC), and 3 to (AD)(BC)

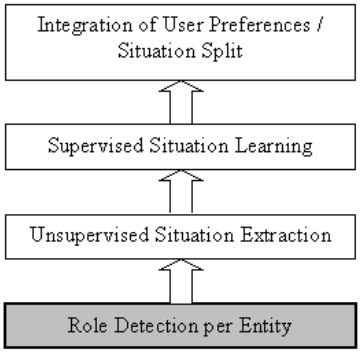
4.3 Conclusions

This chapter described two different implementations for the situation model representing context: a deterministic one based on Petri nets and a probabilistic one based on hidden Markov models. Both approaches have been applied to real world problems with success: an automatic cameraman system (Petri nets) and an interaction group detector (HMMs) have been implemented. Each implementation is well adapted for particular applications. Petri nets can imple-

ment all Allen temporal operators, in particular those describing parallelism. However, Petri nets can not model erroneous perceptions and uncertain situations (uncertain expectations of perceptions for a situation and uncertain situation transitions). Hidden Markov models are less rich in modeling temporal constraints (in particular parallelism), but HMMs permit to model erroneous input and uncertain situations. Thus the choice of the implementation depends on the application that is envisaged.

Chapter 5

Learning Role Acceptance Tests



This chapter addresses the learning of role acceptance tests. The proposed methods constitute the first layer of the framework for acquiring and evolving situation models. As we saw in section 3.4.1, a role acceptance test is divided into a filtering phase and role and relation assignment. The filtering phase is considered to be the key process, isolating abstract events necessary to activate a role. These abstract events fuse and filter event streams coming from perceptual components. The event streams contain the available entities as well as their properties. In the following, we will first present three methods for learning different role labels from entity event streams. Role labels refer to the abstract events necessary for role assignment. The proposed methods are based on a Bayesian classifier, support vector machines (SVMs) and a hybrid classifier combining Bayesian methods and SVMs. The three proposed methods have been evaluated on data sets based on event streams coming from a video tracking system in an augmented home environment (see also lecture scenario in subsection 3.4.2). Augmented home environment, video tracking system, employed role labels and recorded data sets will also be presented.

5.1 Method

By using machine learning methods, our system is to find a connection between the sensed information (entity properties per observation frame) and the roles as perceived and labeled by the supervisor. We are focusing particularly on Bayesian methods, because they are well adapted to deal with erroneous sensor data and they have proven to be useful in many application domains, in particular computer vision [71, 81, 100]. In the following, we will present three methods: a Bayesian classifier, support vector machines and a novel hybrid classifier for identifying unseen roles.

5.1.1 Bayesian Classifier

We first use a generative learning method to model and learn our data. On the basis of the sensor data and the associated role labels, we seek to learn a probabilistic classifier for relevant roles. The proposed Bayesian classifier is similar to classifiers proposed by Ribeiro and Santos-Victor [81] and Muehlenbrock et al. [68]. The classification is done framewise, i.e. the classifier takes the entity properties of one observation frame as input and generates the role prediction for the frame as output.

We seek to determine the role r_{MAP} with the maximum a posteriori (MAP) probability, given the entity property set T (equation 5.1).

$$r_{MAP} = \underset{r}{\operatorname{argmax}}\, P(r|T) \tag{5.1}$$

$$P(r|T) = \frac{P(T|r)P(r)}{P(T)} \tag{5.2}$$

We apply Bayesian theorem (equation 5.2) and we further assume that the prior probabilities $P(r)$ for the roles are equal for each frame. As the constant denominator can be eliminated because of the argmax, we get (equation 5.3).

$$r_{MAP} = \underset{r}{\operatorname{argmax}}\, P(T|r) \tag{5.3}$$

We model $P(T|r)$ for each role as multidimensional Gaussian mixture distribution estimated by running EM algorithm [14] on the learning data. The initial number of Gaussians in the mixture is set to a high value (in the evaluation: 128); Gaussians with too weak contribution to the mixture are successively eliminated.

5.1.2 Support Vector Machines

In order to further improve recognition results, we use a discriminative learning method to classify our data. Support vector machines (SVMs) [17, 36] are well-known to be a powerful discriminative learning method. As for the Bayesian classifier, the classification is done frame-wise, i.e. the SVMs take the entity properties of one observation frame as input and generate the role prediction for the frame as output. SVMs classify data through determination of a set of support vectors, through minimization of the average error. The support vectors are members of the set of training inputs that outline a hyperplane in feature space. This l-dimensional hyperplane, where l is the number of features of the input vectors, defines the boundary between the different classes. The classification task is simply to determine on which side of the hyperplane the testing vectors reside.

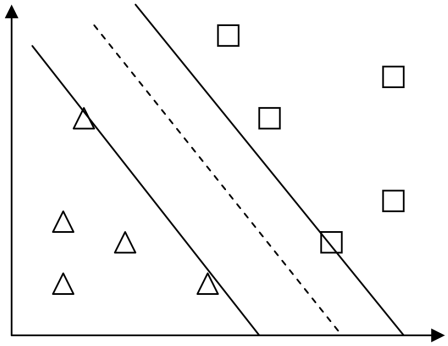


Figure 5.1: SVM classifier hyperplane and margins for a training set of two classes (△ and □)

Given a training set of instance-label pairs $(x_i, y_i), i = 1..l$ where $x_i \in \Re^n$ and $y_i \in \{1, -1\}$ (two class problem), the support vector machines require the solution of the following optimization problem:

$$\min_{\omega,b,\xi} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}\xi_i \quad subject \quad to \quad y_i(\omega^T\phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{5.4}$$

Here training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function ϕ. Then the SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space (Figure 5.1). $C > 0$ is the penalty parameter of the error term. $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is called the kernel function. Though new kernels are being proposed by researchers, there are four basic kernels:

- Linear: $K(x_i, x_j) = x_i^T x_j$

- Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \quad \gamma > 0$
- Radial basis function (RBF): $K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|), \gamma > 0$
- Sigmoid: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$

γ, r and d are kernel parameters. For multi-class classification, a "one-against-one" classification for each of the k classes can be done. $\frac{K(K-1)}{2}$ classifiers are then generated to train the data, where each training vector is compared against two different classes and the error (between the separating hyperplane margin) is minimized. The classification of the testing data is accomplished by a voting strategy, where the winner of each binary comparison increments a counter. The class with the highest counter value after all classes have been compared is selected.

5.1.3 Hybrid Classifier for Identifying Unseen Roles

As we will see in section 5.2, SVMs are a discriminative classification method that outperforms the generative Bayesian classifier for particular data sets. However, SVMs do not provide reliable information about whether or not a new data item may be coherent with the training data sets. Although there are probabilistic SVMs [75], the generated probabilities only refer to the distribution within the trained classes. Unseen data such as wrong target detections or new role classes cannot be identified. These data will be attributed to one of the existing classes. The Bayesian classifier is a generative classification method that generates a model for the training data, providing a possible probability output for each new data item. A hybrid classifier combines the strong points of each method: the probabilistic output of the Bayesian classifier and the discriminative power of the SVMs. First approaches for such a classifier have been applied to text-independent speaker identification [49]. The focus, however, was on classification of trained speakers; unseen classes/data have not been considered.

In the following, we propose a hybrid classifier combining Bayesian methods for identifying unseen data and SVMs for classifying seen data. We will compare the method with an extended Bayesian classifier and classical SVMs. The architecture of the classifiers can be seen in Figure 5.2.

In subsection 5.1.1, we used equation 5.3 to determine the class of a new data item. We modeled $P(T|r)$ for each role as multidimensional Gaussian mixture distribution estimated by EM. We have extended this by modeling additionally $P(T)$ as multidimensional Gaussian mixture distribution estimated by EM. $P(T)$ makes it possible to estimate the probability for a new data item to be generated from the training data set model. By using a threshold on this probability value, we can determine whether the new data item is part of the learned classes or whether it is unseen data (e.g. wrong detections or new class). The threshold can be automatically estimated

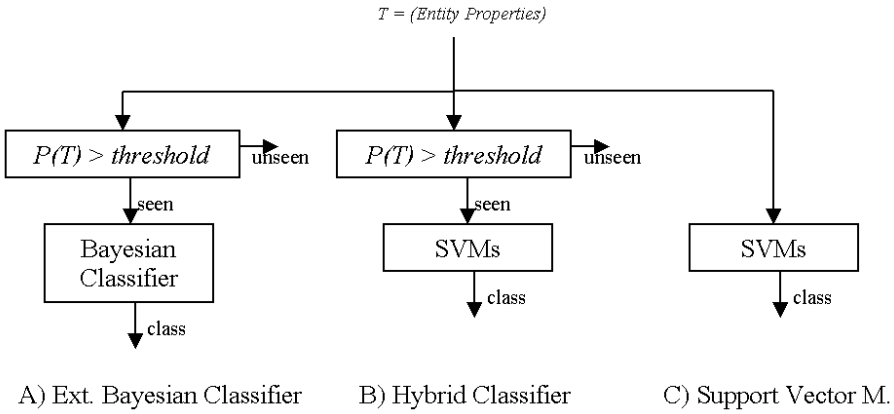


Figure 5.2: Extended Bayesian classifier, hybrid classifier and support vector machines

from the training data sets (based on minimal probability of data items of the classes). The hybrid classifier (Figure 5.2 B) combines the estimation of $P(T)$ (generative model) with SVMs trained on the classes. If a data item is determined to be seen data, the SVMs determine the class of this item. For evaluation, we compare the hybrid classifier with an extended Bayesian classifier (Figure 5.2 A) and classical SVMs (Figure 5.2 C). The extended Bayesian classifier combines the estimation of $P(T)$ with a classical Bayesian classifier. We want to show that the hybrid classifier outperforms both a purely Bayesian classifier and a purely SVM approach.

5.2 Evaluation and Results

In this section, we will evaluate the proposed classifiers on several data sets. The data sets are the recorded properties of targets (entities) detected by a video tracking system in a smart environment In the following, we will first present the smart home environment where the recordings took place as well as the video tracking system. Then, the recorded data sets and role values will be described. Finally, the results of the different classifiers will be depicted.

5.2.1 Smart Home Environment

The experiments described in this section are performed in a laboratory mockup of a living room environment in a smart home. The environment contains a small table surrounded by three armchairs and one couch (Figure 5.3 left). Microphone arrays and video cameras are mounted on all walls in the environment. In this paper we concentrate on the use of a single wide-angle video camera mounted in a corner of the smart room (Figure 5.3 right) opposite the couch.

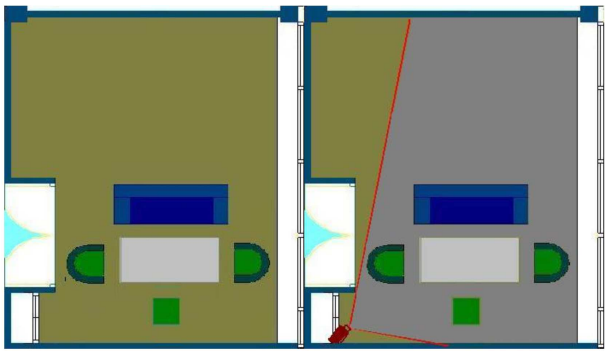


Figure 5.3: Map of our Smart Room (left), wide-angle camera view shown in gray (right)

The wide-angle camera observes the environment (Figure 5.4) with a frame rate between 15 and 20 images per second. A real-time robust tracking system detects and tracks targets in the video images [24, 103].

Figure 5.4: The Smart Room environment as seen by the wide angle camera

5.2.2 Video Tracking System

The video tracking system employs a supervisory controller to dynamically control the selection of processing modules and the parameters used for processing (Figure 5.5). This system employs a library of pixel level detection operations to detect and track blobs at video rate. In our smart environment, adaptive background subtraction is used to detect and track moving users in the environment. A central supervisor is used to adapt processing parameters so as to

maintain reliable real-time tracking. Within a detection level targets can be detected by energy measurements based on background subtraction or intensity normalized color histograms.

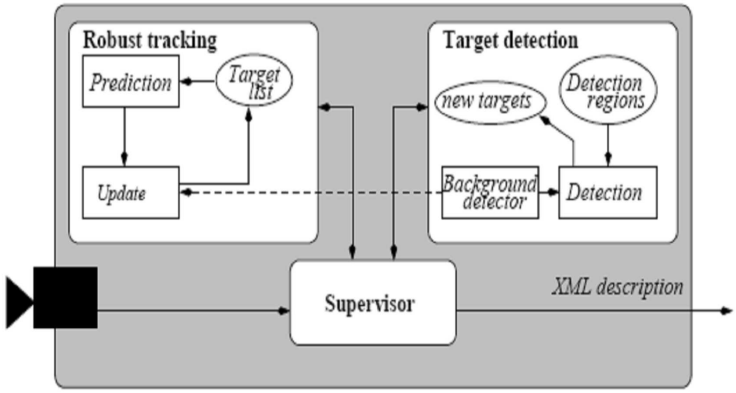


Figure 5.5: Architecture of the robust tracking system

The robust tracking module is a form of Kalman filter [96] operating on the list of current targets. For each target a search region and a Gaussian mask centered on the most likely position is determined using a linear prediction from the previous image. A pixel level detection algorithm is executed with the search region, and the detected pixels are then multiplied by the Gaussian mask. The first moment of this product provides a new estimate for the target position, while the eigenvectors of the second moment provide position orientation and width, and height.

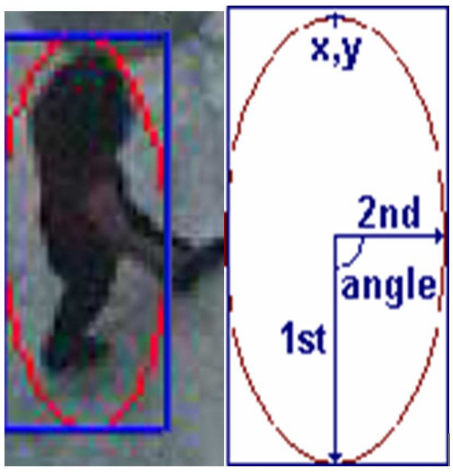


Figure 5.6: Target properties returned by the system

The video tracking system returns a vector of properties for each video frame. Each vector contains the position, size and orientation of one target detected and tracked by the system. The returned properties for each target are top position (x, y) of the bounding ellipse, the radius of

the first and second axis of the ellipse and the angle describing the orientation of the ellipse (Figure 5.6). Additional features including velocity, speed or energy can also be determined from the target tracking process.

5.2.3 Individual Role Values

The five elementary roles that we want to recognize in this evaluation are: “walking”, “standing”, “sitting”, “interaction with table” and “lying” (Figure 5.7).

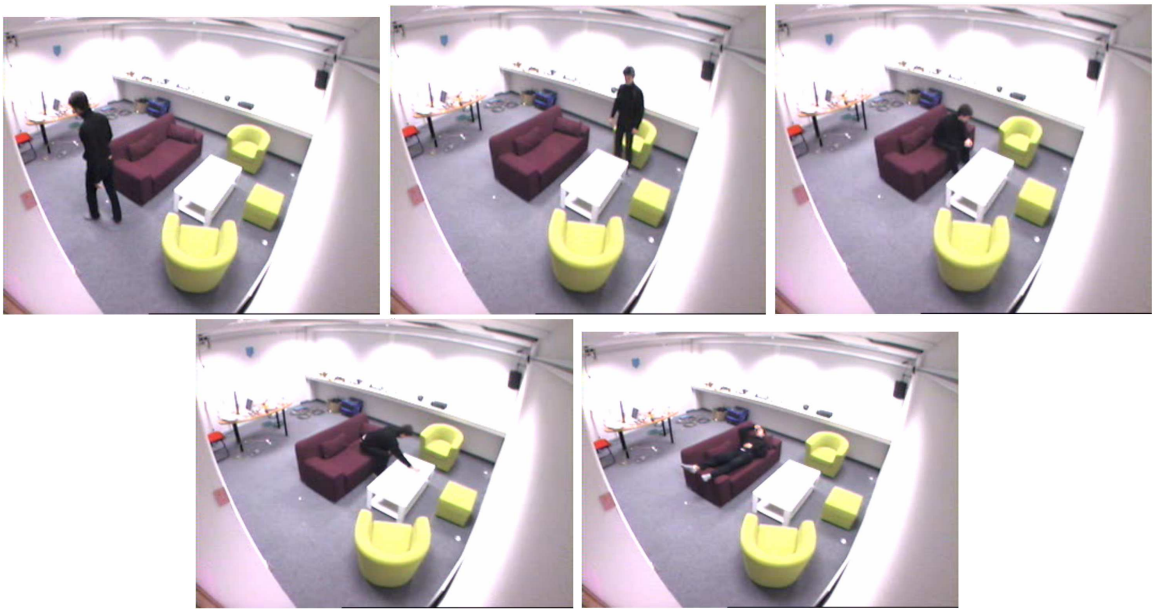


Figure 5.7: Individual roles (from left to right): “walking”, “standing”, “sitting”, “interaction with table” and “lying”

5.2.4 Data Sets

In order to develop and evaluate the recognition process, we recorded 8 short video sequences in the environment. During these sequences, one or several individuals played different elementary roles in the smart room. The number of frames and the number of different roles played during the sequences are indicated in Table 5.1. The overall distribution of the different roles in the data sets is depicted in Table 5.2.

The roles played by the individuals in the video sequences have been hand labeled for use in learning and evaluation. The labeling process assigns a role label to each target detected by the

Sequence	No. frames	No. Role labels
1	1352	4
2	6186	5
3	4446	5
4	4684	5
5	4027	5
6	4477	5
7	3067	5
8	3147	5
Total	30885	5

Table 5.1: Video sequence recordings

robust tracking system for each frame. The labeler had the possibility of assigning a “no role” label if a detected target did not appear to play any of the five elementary roles. Thus, each of the 8 data sets contains a list of target properties $(x, y, first\,radius, second\,radius, angle)$ and the associated role label.

Video Sequence	Walking (%)	Standing (%)	Sitting (%)	Inter. table (%)	Lying (%)
1	0.79	0.03	0.11	0.00	0.08
2	0.14	0.06	0.48	0.24	0.08
3	0.14	0.09	0.40	0.24	0.12
4	0.14	0.08	0.50	0.18	0.10
5	0.17	0.09	0.46	0.18	0.11
6	0.19	0.12	0.42	0.16	0.10
7	0.13	0.12	0.50	0.15	0.11
8	0.15	0.09	0.48	0.19	0.09
Total	0.18	0.09	0.45	0.19	0.10

Table 5.2: Distribution of the different role labels in the data sets

5.2.5 Results

Bayesian classifier, support vector machines and hybrid classifier have been evaluated on the data sets described in subsection 5.2.4. First, we will present the results of the Bayesian classifier and the support vector machines. We evaluate the recognition of all role values using cross-validation. Then, the results of the hybrid classifier will be described. Here, we evaluate the recognition/identification of new role values (classes) excluded from learning. We compare the

performance of an extended Bayesian classifier, the hybrid classifier and classical SVMs for this task of identifying unseen role values.

Bayesian Classifier

We evaluated the Bayesian classifier on the video sequence recordings (Table 5.1) using 8-fold cross-validation. Each sequence has been used for testing once, while learning the model with the 7 remaining sequences. The average classification results can be seen in form of confusion matrices in Table 5.3, Table 5.4 and Table 5.5.

	Walking	Standing	Sitting	Inter. table	Lying
Walking	0.8015	0.1536	0.033	0.0089	0.003
Standing	0.4809	0.4611	0.0349	0.0217	0.0013
Sitting	0.0704	0.0221	0.7385	0.0833	0.0856
Inter. table	0.0324	0.0257	0.1163	0.8207	0.0049
Lying	0.0006	0.0013	0.1405	0.0005	0.8571

Class	TP rate	FP rate	Precision	Recall	F-measure
Walking	0.8015	0.0967	0.6889	0.8015	0.7359
Standing	0.4611	0.0476	0.4732	0.4611	0.4474
Sitting	0.7385	0.082	0.8403	0.7385	0.7845
Inter. table	0.8208	0.0605	0.7636	0.8208	0.7827
Lying	0.8571	0.0378	0.7113	0.8571	0.764
Total	0.7358	0.0649	0.6955	0.7358	0.7029

Table 5.3: Confusion matrix and information retrieval statistics for Bayesian classifier with $T = (X, Y)$

We evaluated three different target (entity) property sets T. The first set was the position X, Y in the image. The results are good (Table 5.3) showing that the position in the environment is discriminating for individual roles. Position is, however, very dependent on environment configuration, e.g. couch and chair localization. Therefore, the second target set was $(1st, 2nd, angle)$, which only contains information on the form of the ellipse and not its position. The results (Table 5.4) are quite similar to those obtained for the position.

The combination of the first and second target property sets $(X, Y, angle, 1st, 2nd)$ gives the best results (Table 5.5). In general, ambiguous roles like “sitting” and “interacting with table” or in particular “walking” and “standing” are difficult to distinguish for each frame (even for a human supervisor!), which leads to numerous wrong classifications.

The overall results of the Bayesian classifier can be seen in the left column of Table 5.9.

	Walking	Standing	Sitting	Inter. table	Lying
Walking	0.8077	0.1552	0.0172	0.0093	0.0105
Standing	0.5403	0.3967	0.0376	0.019	0.0064
Sitting	0.0678	0.024	0.7668	0.1133	0.0281
Inter. table	0.0455	0.0257	0.1329	0.7668	0.0291
Lying	0.0176	0.0086	0.0442	0.0216	0.908

Class	TP rate	FP rate	Precision	Recall	F-measure
Walking	0.8077	0.1066	0.6781	0.8077	0.7311
Standing	0.3967	0.051	0.4195	0.3967	0.39
Sitting	0.7668	0.0655	0.8603	0.7668	0.8086
Inter. table	0.7667	0.0789	0.7154	0.7667	0.7341
Lying	0.9079	0.0247	0.8202	0.9079	0.8545
Total	0.7292	0.0653	0.6987	0.7292	0.7037

Table 5.4: Confusion matrix and information retrieval statistics for Bayesian classifier with $T = (1st, 2nd, angle)$

	Walking	Standing	Sitting	Inter. table	Lying
Walking	0.9335	0.0499	0.0154	0.0008	0.0004
Standing	0.6759	0.2738	0.0393	0.0106	0.0004
Sitting	0.0882	0.0169	0.7757	0.1046	0.0145
Inter. table	0.014	0.0248	0.154	0.8073	0
Lying	0.0142	0.0051	0.0202	0.0053	0.9553

Class	TP rate	FP rate	Precision	Recall	F-measure
Walking	0.9336	0.127	0.7019	0.9336	0.7966
Standing	0.2739	0.0231	0.3959	0.2739	0.3156
Sitting	0.7757	0.0679	0.8543	0.7757	0.8109
Inter. table	0.8074	0.0701	0.7304	0.8074	0.7606
Lying	0.9553	0.0044	0.9582	0.9553	0.9561
Total	0.7492	0.0585	0.7281	0.7492	0.7280

Table 5.5: Confusion matrix and information retrieval statistics for Bayesian classifier with $T = (X, Y, 1st, 2nd, angle)$

Support Vector Machines

Like the Bayesian classifier, the SVMs have been evaluated on the video sequence recordings (Table 5.1) using 8-fold cross-validation. A radial basis function kernel (see subsection 5.1.2) with $C = 11.0$ and $\gamma = 11.0$ showed good results for our training data. The LIBSVM library [26] has been used for implementation and evaluation.

	Walking	Standing	Sitting	Inter. table	Lying
Walking	0.8198	0.0824	0.0732	0.0206	0.004
Standing	0.6035	0.2884	0.0793	0.0288	0
Sitting	0.0293	0.0005	0.8471	0.0593	0.0638
Inter. table	0.0014	0.0014	0.1661	0.8275	0.0036
Lying	0	0	0.1422	0	0.8578

Class	TP rate	FP rate	Precision	Recall	F-measure
Walking	0.82	0.0813	0.7001	0.82	0.7489
Standing	0.2882	0.025	0.5922	0.2882	0.3333
Sitting	0.8472	0.1198	0.8046	0.8472	0.8183
Inter. table	0.8276	0.0495	0.8101	0.8276	0.811
Lying	0.8579	0.0296	0.7959	0.8579	0.815
Total	0.7282	0.0610	0.7406	0.7282	0.7053

Table 5.6: Confusion matrix and information retrieval statistics for SVMs with $T = (X, Y)$

Again we evaluated the three different target (entity) property sets $T = (X, Y)$, $T = (1st, 2nd, angle)$ and $T = (X, Y, 1st, 2nd, angle)$. As for the Bayesian classifier, the results for the position (Table 5.6) and the form of the ellipse (Table 5.7) are quite similar. The combination of target property sets (X, Y) and $(1st, 2nd, angle)$ produced best results (Table 5.8). As for Bayesian classifier, the ambiguity of roles, in particular between “walking” and “standing”, persists, resulting in a reduced precision for “walking” and a poor recall for “standing”.

The overall classification results of the 8-fold cross-validation for the SVMs are depicted in the right column of Table 5.9. Both SVM and the Bayesian Classification are applied framewise. That is, the target properties for each frame are used to produce a role label, independent of values in other frames. Because the SVM is a discriminative method, it optimizes classification between the given/trained classes, outperforming the Bayesian classifier. However, SVM does not learn the structure for a given data set, but only borders and margins between classes. As a result, with the SVM it is difficult or impossible to reject unseen test data (“garbage”) or to discover new classes of roles.

	Walking	Standing	Sitting	Inter. table	Lying
Walking	0.8522	0.0923	0.0342	0.0145	0.0068
Standing	0.7701	0.0923	0.1091	0.0237	0.0048
Sitting	0.0514	0.0042	0.8536	0.0756	0.0153
Inter. table	0.0083	0.0053	0.2044	0.757	0.025
Lying	0.0071	0.0004	0.0917	0.0181	0.8827

Class	TP rate	FP rate	Precision	Recall	F-measure
Walking	0.8523	0.1147	0.6732	0.8523	0.7444
Standing	0.0923	0.0271	0.353	0.0923	0.1423
Sitting	0.8536	0.1155	0.8091	0.8536	0.8282
Inter. table	0.7569	0.0555	0.7829	0.7569	0.7624
Lying	0.8827	0.0155	0.8869	0.8827	0.8758
Total	0.6876	0.0657	0.7010	0.6876	0.6706

Table 5.7: Confusion matrix and information retrieval statistics for SVMs with $T = (1st, 2nd, angle)$

	Walking	Standing	Sitting	Inter. table	Lying
Walking	0.8999	0.0849	0.0131	0	0.0022
Standing	0.5438	0.4066	0.0418	0.0069	0.001
Sitting	0.0425	0.008	0.8851	0.0396	0.0248
Inter. table	0.0066	0.0049	0.1346	0.8539	0
Lying	0.0049	0.0005	0.0269	0	0.9676

Class	TP rate	FP rate	Precision	Recall	F-measure
Walking	0.8999	0.0834	0.7398	0.8999	0.8079
Standing	0.4067	0.0264	0.6086	0.4067	0.4529
Sitting	0.885	0.0628	0.8765	0.885	0.8802
Inter. table	0.854	0.0271	0.8782	0.854	0.8632
Lying	0.9676	0.0054	0.9538	0.9676	0.9583
Total	0.8026	0.0410	0.8114	0.8026	0.7925

Table 5.8: Confusion matrix and information retrieval statistics for SVMs with $T = (X, Y, 1st, 2nd, angle)$

		Bayesian Classifier	SVMs
X,Y	Mean	76.959	78.5537
	Std. dev.	4.689	3.9751
1st,2nd,angle	Mean	76.9107	78.1124
	Std. dev.	3.9349	4.6881
X,Y,1st,2nd,angle	Mean	81.5024	86.1033
	Std. dev.	1.4583	2.7573

Table 5.9: Overall recognition rates (in percent) for Bayesian classifier and SVMs

Hybrid Classifier

We evaluated the three different classifiers proposed in subsection 5.1.3 on the video sequence recordings (Table 5.1) using 8-fold cross-validation. In order to test the classifiers on unseen data, we excluded each class once from the training data sets. This resulted in $5 * 8 = 40$ test runs. The obtained overall results for the classifiers are depicted in Table 5.10. The hybrid classifier outperforms the extended Bayesian classifier and the SVMs for the complete data sets.

	Ext. Bayesian Classifier	Hybrid Classifier	SVMs
Mean	75.2157	77.8555	71.0088
Std. dev.	5.4840	6.3881	8.3958

Table 5.10: Overall recognition rates (in percent) for extended Bayesian classifier, hybrid classifier and SVMs with $T = (X, Y, 1st, 2nd, angle)$

Table 5.11 shows the information retrieval statistics of the role classes that have been excluded from training for the hybrid classifier. These results are identical for the extended Bayesian classifier because the detection of the unseen classes by the probability values of $P(T)$ is common for both classifiers. As the classical SVMs are not trained to detect the unseen classes, the TP rate, FP rate, precision, recall and F-measure are zero for SVMs. The detection results for the unseen activities “standing” and “interacting with table” are mediocre. From an activity point of view, both classes overlap with more frequent classes (“walking” and “sitting” respectively), which explains detection errors. A distinct role class like “lying” is, however, very well recognized as unseen. The overall rates indicate that the approach can be used to identify unseen role classes.

Class	TP rate	FP rate	Precision	Recall	F-measure
Walking	0.7374	0.1356	0.6481	0.7374	0.6763
Standing	0.0108	0.001	0.3938	0.0108	0.0208
Sitting	0.7467	0.2677	0.6576	0.7467	0.6713
Inter. table	0.5336	0.1217	0.6845	0.5336	0.5867
Lying	0.8476	0.0631	0.6557	0.8476	0.723
Total	0.5752	0.1178	0.6079	0.5752	0.5356

Table 5.11: Information retrieval statistics of the unseen roles for the hybrid classifier with $T = (X, Y, 1st, 2nd, angle)$

5.3 Conclusions

In this chapter, we presented an approach for learning and recognizing individual roles. The approach is part of a framework for acquiring a high-level context model for human behavior in augmented environments. Role recognition is the backbone of this framework as roles are necessary for determining relations between entities, current situation and scenario. The proposed methods for role learning and recognition are a Bayesian classifier and support vector machines. In order to detect unseen role classes, a hybrid classifier has then been proposed combining generative Bayesian methods and discriminative SVMs.

The proposed methods have been tested and evaluated in a smart home environment. A robust tracking system was used to create and track targets (entities) in wide-angle camera images of the scene. Bayesian classifier and support vector machines have been applied to the extracted target properties $(X, Y, 1st\ radius, 2nd\ radius, angle)$ in order to learn and detect individual target role classes “walking”, “standing”, “sitting”, “interacting with table”, “lying”. The evaluation of both classifiers on recorded data sets showed good results. Support vector machines outperformed the Bayesian classifier. In order to detect unseen role classes, the hybrid classifier has been applied to the recorded data sets. The obtained overall detection results for unseen classes in the recorded data sets are good. The hybrid classifier outperformed the Bayesian classifier and the SVMs when identifying unseen roles, showing that the proposed combination of generative and discriminative methods is beneficial.

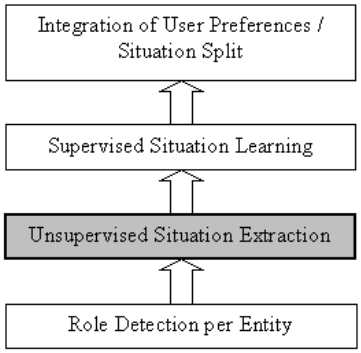
Future work may concern an improvement of the recognition rate for roles. A first step is to integrate additional features into the entity property sets. Taking the example of a target tracking system in a smart home environment, position of face and hands, derived by skin color detector, velocity, speed and background subtraction energy may be prospective candidates. Velocity, speed and energy estimation need also to be added. We would like to mention that, in order to optimize role recognition, it is sometimes useful to reduce the number of classes to be recognized and to treat some classes “manually”. This is especially the case when a class

label is badly recognized in an automatic manner (perhaps due to insufficient learning data), but some entity property values can easily be specified by an engineer in order to detect the concerned class. For example, an engineer may specify a minimal and a maximal distance from the center of the table in order to recognize “interacting with table”. Entity properties like speed may also be specified manually in order to recognize “walking”. The advantage is a reduction of the amount of learning data that is necessary for a learning algorithm like SVMs in order to learn correctly all class labels. As the amount of available learning data is normally limited, a reduction of the number of classes and manual specification of some “difficult” classes can largely improve the overall recognition rates. Subsection 9.2.2 will give an example of such a manual specification of some class labels.

In the following chapters, we will pass to the next step: learning and recognizing situations.

Chapter 6

Unsupervised Situation Discovery



This chapter addresses the unsupervised discovery of situations from multimodal observation. The proposed method constitutes the second layer of the framework for acquiring and evolving situation models. Multimodal observation generation is based on the entity end role detection processes (chapter 5). The properties associated to each entity and the roles played by these entities are used to generate multimodal observations for human activity in the scene. We assume here a constant sampling rate for these observations. The objective of situation discovery is the offline segmentation of the incoming multimodal observation stream. Each segment corresponds then to a temporal interval containing the multimodal observations of one situation. These segments can later serve as input for supervised situation learning.

The unsupervised method proposed in this chapter has been designed and tested in the field of automatic analysis of small group meetings. Automatic analysis of small group meetings is an emerging field of research for speech, video and multimodal technologies. In general, the group and its members are defined in advance. The objective is then to recognize particular key actions executed by group members [65] or to analyze the type of meeting in a global manner [23]. However, the detection of dependencies between individuals and their membership in

one or several groups is not considered. Analyzing large amounts of data from recordings of interactions enables the reconstruction of social networks for a number of individuals [30]. The detection and analysis of conversations is then necessary. The automatic detection of conversations using mutual information [11], in order to determine who speaks and when, requires a significantly long duration for each conversation. Little work has been done on the analysis of changing small group configuration and activity. Aoki et al. [8] have presented a user study of a system determining and supporting interaction groups in an audiospace. The system uses a naive Bayesian classifier to determine the interaction group configuration. However, the focus of the paper is laid on the user study, no detection results are presented. In [18] (and briefly in section 4.2), a real-time detector for small group configurations of 4 participants has been proposed. The detector takes speech activity events of meeting participants as input. The meeting situations describing the possible group configurations have been defined in advance (Figure 6.1).

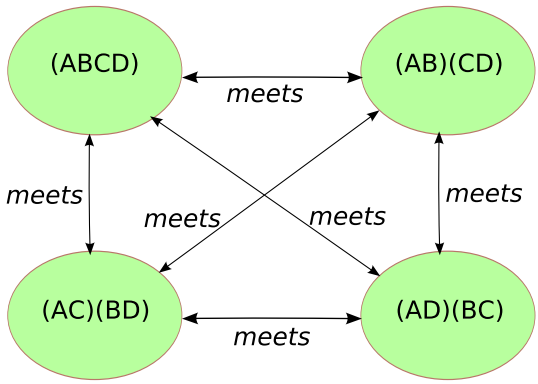


Figure 6.1: Situation model describing possible group configurations for a meeting of 4 individuals A, B, C, D

However, a predefinition of meeting situations (as in [18, 65]) is not always possible in advance, especially when dealing with an increasing number of participants and informal meetings. Thus we propose an unsupervised method for detecting small group meeting configurations and activities from a stream of multimodal observations. This first segmentation can be used as input for classification and detection of activities. The proposed method detects changes in small group configuration and activity based on measuring the Jeffrey divergence between adjacent histograms of observations. In [18], the authors showed that different meeting activities, and especially different group configurations, have particular distributions of speech activity. This can be extended to distributions of multimodal observations coming from multi-sensory input. These distributions are represented by histograms containing the frequency of these observations. To separate distinct distributions of observations, two adjacent windows are slid from the beginning to the end of the meeting recording, while constantly calculating the Jeffrey divergence between the histograms generated from the observations within these windows. The size of the sliding adjacent windows is varied generating several Jeffrey divergence curves. The

peaks of the resulting curves are detected using successive robust mean estimation. The detected peaks are merged and filtered with respect to their height and window size. The retained peaks are finally used to select the best model, i.e. the best allocation of observation distributions for the given meeting recording.

The method has been tested on observation recordings of 7 meetings. Five speech activity recordings of short small group meetings with 4 participants, one speech activity recording of a seminar with 5 participants and a audiovisual observation recording of a cocktail party meeting with 5 participants. The approach showed promising results for all meeting recordings.

6.1 Method

A novel method based on the calculation of the Jeffrey divergence between histograms of observations is presented. These observations are a discretization of events coming from multisensory input. The observations are generated with a constant sampling rate depending on the sampling rates of the sensors.

6.1.1 Observation Distributions

In [18], the authors stated that the distribution of the different speech activity observations is discriminating for group configurations in small group meetings. It is further assumed that in small group meetings distinct group configurations and activities have distinct distributions of multimodal observations. The objective of the proposed approach is hence to separate these distinct distributions, in order to identify distinct small meeting configurations and activities. Because the observations are discrete and unordered (e.g. a 1-dimensional discrete code) and there is no a priori observation distribution, histograms are used to represent observation distributions. A histogram is calculated for an observation window (i.e. the observations between two distinct time points in the meeting recording) and contains the frequency of each observation code within this window.

$$J_{p,q} = \sum_{x \in X} p(x) \cdot log \frac{p(x)}{\frac{p(x)+q(x)}{2}} + q(x) \cdot log \frac{q(x)}{\frac{p(x)+q(x)}{2}} \tag{6.1}$$

The Jeffrey divergence [76] is a numerically stable and symmetric form of the Kullback-Leibler divergence between histograms. Equation 6.1 indicates the formula to calculate the Jeffrey divergence between two histograms p and q. The set X contains the bins of the histograms. The value $p(x)$ refers to the empirical probability of the observation associated to bin x.

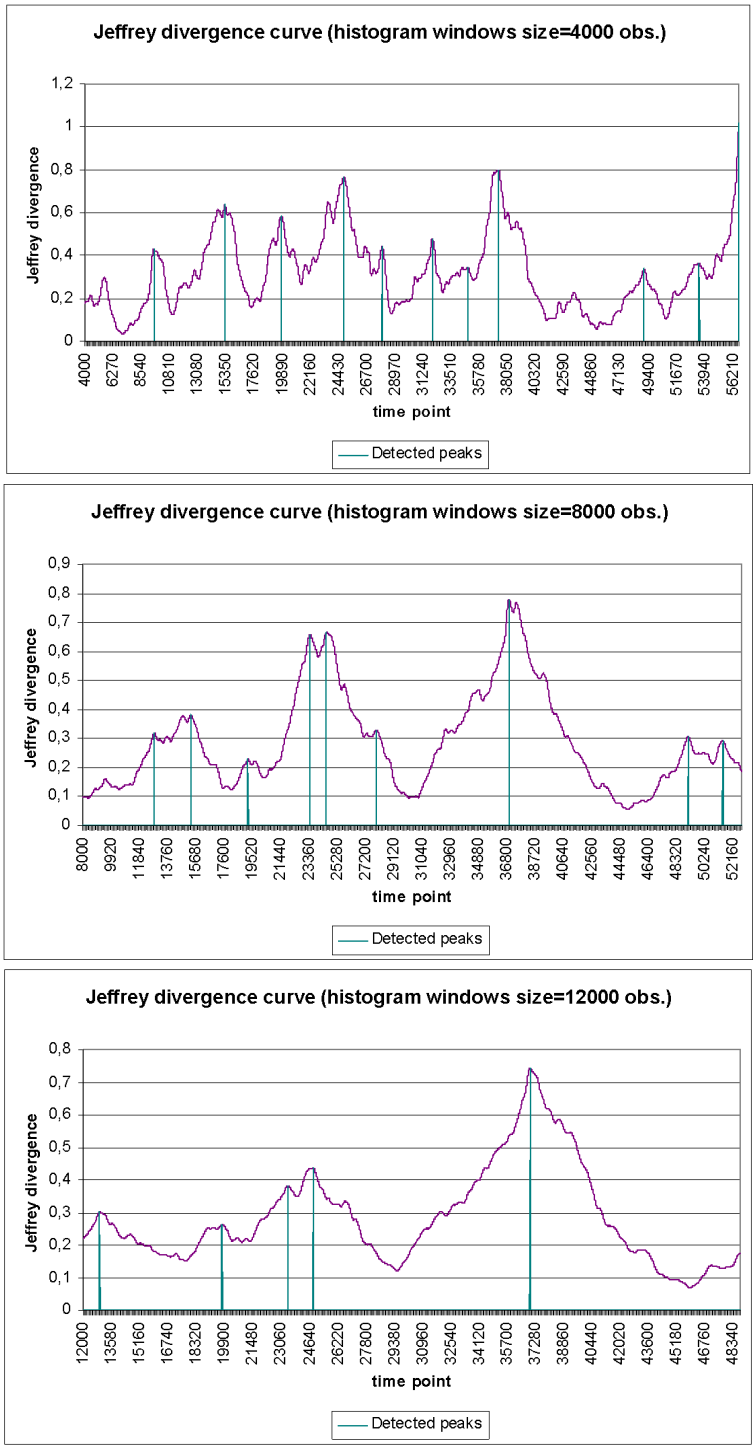


Figure 6.2: Small Group Meeting 5: Jeffrey divergence between histograms of sliding adjacent windows of 4000, 8000 and 12000 speech activity observations (64sec, 2min 8sec and 3min 12sec)

The Jeffrey divergence may be used to separate different observation distributions by calculating the divergence between the histograms of two adjacent observation windows. With this approach, two adjacent observation windows are slid from the beginning to the end of the recorded meetings, and the Jeffrey divergence is computed for each position. The result is a divergence curve of adjacent histograms (Figure 6.2).

The peaks of the Jeffrey divergence curve can be used to detect changes in the observation distribution of a meeting recording. The peaks of the curves indicate high divergence values, i.e. a big difference between the adjacent histograms at that time point. The size of the adjacent windows determines the exactitude of the divergence measurement. The larger the window size, the less peaks has the curve. However, peaks of larger window sizes are less precise than those of smaller window sizes.

As the observations are generated with a fixed sampling rate, an observation window size used for the calculation of a histogram corresponds to a temporal interval. Different window sizes cover thus the detection of activities with different durations. As we do not want to have a strong a priori concerning the duration of activities and group configurations, we apply our method to several different window sizes. The choice of these window sizes is fixed by the minimal duration of the activities that we expect. For small group meetings, we fixed a minimal duration between 64sec and 4min 16sec, which corresponds to a window size of between 4000 and 16000 audio observations.

6.1.2 Peak Detection

To detect the peaks of the Jeffrey divergence curve, successive robust mean estimation is used. Robust mean estimation detects the dominant peak of the Jeffrey divergence curve. Successive robust mean estimation applies the robust mean estimation process several times to the curve in order to isolate all peaks. In the following, we will first detail the robust mean estimation process and the associated equations. Then, successive robust mean estimation will be described.

Robust mean estimation has first been used by Qian et al. [77] to locate the center position of a dominant face in skin color filtered images. The idea is to calculate iteratively a trimmed mean for the filtered pixels of the image. The trimmed mean converges towards the dominant skin color blob in the image.

Step 1. Compute mean μ and standard deviation σ based on all the points of the Jeffrey curve.

Step 2. Let $\mu(0)=\mu$ and $\delta(0)=max(\sigma, mindev)$.

Step 3. Compute trimmed mean $\mu(k+1)$ and deviation $\delta(k+1)$ based on points within the interval $[\mu(k)-\delta(k), \mu(k)+\delta(k)]$.

Step 4. Repeat Step 3 until $|\mu(k+1)-\mu(k)| < \varepsilon$. Denote the converged mean as μ^* and the converged deviation δ^*.

Step 5. Set the dominant peak position p^* to the position of the maximum within the interval $[\mu^*-\delta^*, \mu^*+\delta^*]$.

Figure 6.3: Robust mean estimation process detecting a dominant peak of the Jeffrey divergence curve

Figure 6.3 describes the robust mean estimation process to detect the dominant peak of the Jeffrey divergence curve. The first step of robust mean estimation is to calculate global mean μ and standard deviation σ for the Jeffrey divergence curve using equations 6.2 and 6.3.

$$\mu = \frac{1}{\widehat{J}} \sum_{t=t_{MIN}}^{t_{MAX}} t \cdot J_{h[t-size,t],h[t,t+size]} \tag{6.2}$$

$$\sigma = \sqrt{\frac{1}{\widehat{J}} \sum_{t=t_{MIN}}^{t_{MAX}} (t-\mu)^2 \cdot J_{h[t-size,t],h[t,t+size]}} \tag{6.3}$$

$$\widehat{J} = \sum_{t=t_{MIN}}^{t_{MAX}} J_{h[t-size,t],h[t,t+size]} \tag{6.4}$$

$J_{h[t-size,t],h[t,t+size]}$ refers to the Jeffrey divergence between the adjacent histograms of size $size$ at time point t. Both equations are normalized by the sum of all Jeffrey divergence values (equation 6.4). In the second and third step, a new trimmed mean $\mu(k+1)$ and deviation $\delta(k+1)$ are calculated based on the Jeffrey curve values within the (standard) deviation around the previous (global) mean. This process is repeated until the trimmed mean converges (Step 4). The maximum within the last interval is set to be the dominant peak of the Jeffrey divergence curve.

To detect all peaks of the Jeffrey divergence curve, the robust mean estimation process is successively applied (Figure 6.4). After each robust mean estimation, the found dominant peak

Step 1. Detect dominant peak p^* using robust mean estimation.

Step 2. Erase points within peak window $[\mu^*-\delta^*, \mu^*+\delta^*]$ from Jeffrey divergence curve.

Step 3. Repeat Steps 1 and 2 until $J_{h[p^*-size,\,p^*],\,h[p^*,\,p^*+size]} \leq \bar{J}$ with

$$\bar{J} = \frac{1}{t_{MAX}-t_{MIN}+1} \sum_{t=t_{MIN}}^{t_{MAX}} J_{h[t-size,t],\,h[t,t+size]} \,.$$

Figure 6.4: Successive robust mean estimation process detecting the peaks of the Jeffrey divergence curve

is erased (Step 2). This process is repeated while the heights of isolated peaks are above the average height of the curve (Step 3).

6.1.3 Merging and Filtering Peaks from Different Window Sizes

Peak detection is conducted for a fixed histogram window size, i.e. the size of the adjacent observation windows used for calculating the histograms needs to be specified for the successive robust mean estimation process (subsection 6.1.2).

Peak detection using successive robust mean estimation (subsection 6.1.2) is conducted for Jeffrey curves with different histogram window sizes. The window size refers to the observation window used for calculating the histograms. Figure 6.2 shows example Jeffrey curves for three different observation window sizes. We see that some peaks are detected for several curves, while others are specific for one particular window size. In order to determine which peaks to choose for segmenting the multimodal observation recording, we need first to merge peaks appearing at several window sizes and to filter these peaks locally with respect to their window size and peak height.

Merging peaks

To merge peaks appearing at several histogram window sizes, we need to calculate the distance between these peaks. Figure 6.5 proposes a normalized distance measure between peaks of different window sizes. The (temporal) distance between two peaks is normalized by the minimum of the involved histogram window sizes. The resulting normalized distance measures the degree of overlap between the histogram windows.

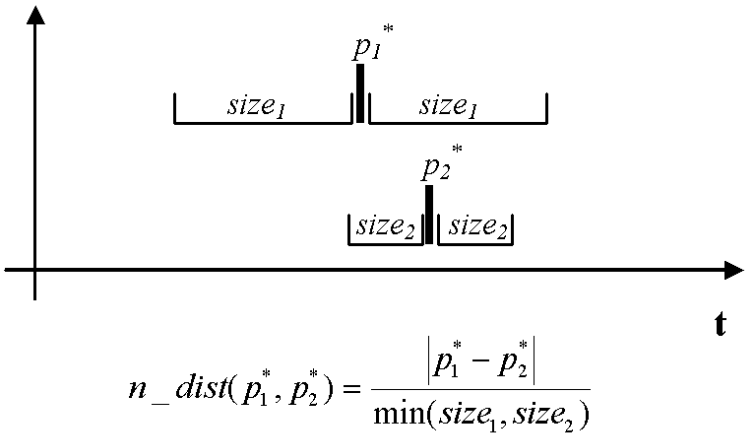


Figure 6.5: Normalized distance n_dist between two peaks p_1^*, p_2^* of Jeffrey curves with different window sizes $size_1$, $size_2$

To merge two peaks, the histogram windows on both sides of the peaks must overlap, i.e. the normalized distance must be less than 1.0. The position of the resulting merged peak is determined by the position of the highest peak that has been merged.

Filtering peaks

The resulting peaks are filtered by measuring peak quality. Relative peak height and number of votes are introduced as quality measures. The relative peak height is the Jeffrey curve value of the peak point normalized by the maximum value of the Jeffrey curve (with the same window size). A peak needs to have a relative peak height between 0.5 and 0.6 to be retained. The number of votes of a peak is the number of peaks that have been merged to form this peak. A number of 2 votes are necessary for a peak to be retained.

Merging and filtering operate on the positions and features of the detected peaks, i.e. in a local context. In order to determine the best allocation of observation distributions for a given recording, we need to search for the best combination of the peaks retained by the merging and filtering process. This global search process is called *model selection*.

6.1.4 Model Selection

Model selection is a global search process that aims at determining the best allocation of observation distributions for a given recording. The input is the list of peaks retained by the merging

and filtering process. The output is the combination that maximizes the divergence between the distinct observation distributions of the recording. We assume that the best allocation of observation distributions corresponds to maximizing the average divergence between the observation distributions.

To search for the best model for a given recording, all possible peak combinations are examined, i.e. each peak of the final peak list is both included and excluded to the (final) model. For each such peak combination, the average Jeffrey divergence of the histograms between the peaks is calculated. As the goal is to separate best the distinct observation distributions of a recording, the peak combination that maximizes the average divergence between the peak histograms is accepted as the best model for the given recording.

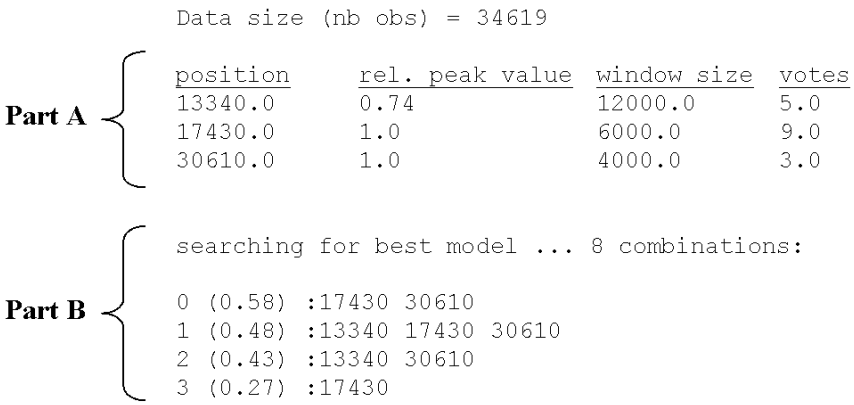


Figure 6.6: Small Group Meeting 1: Output of the algorithm

Figure 6.6 shows an example output of the model selection algorithm. Part A of the figure indicates the resulting peaks of the merging and filtering process. Four peaks have been retained, which means that $4 * 4 = 16$ possible peak combinations must be examined. Part B lists the eight best peak combinations (sorted by descending average Jeffrey divergence) that have been found by the model selection process. Model 0 would have been selected, corresponding to a segmentation of the recording at positions $17430, 30610$ and an average Jeffrey divergence between the three segments of 0.58.

6.2 Evaluation and Results

To evaluate our approach, 5 short small group meetings (subsection 6.2.2), one seminar (subsection 6.2.3) and a cocktail party meeting (subsection 6.2.4) have been recorded. The group

configurations and activities of these meetings have been hand labeled. The result of the proposed approach is the peak combination separating best the activity distributions for each meeting recording. The intervals between the peaks are interpreted as segments of distinct group configuration and activity. The asp, aap and Q measures (described in subsection 6.2.1) are used for the evaluation of these segments with regard to the labeled group configurations and activities.

6.2.1 Segment Quality Measure

The timestamps and durations of the (correct) group configurations and activities have been hand labeled. As the proposed method is unsupervised, the direct correspondence between detected segments and hand labeled activities cannot be measured (e.g. by using confusion matrices) because the unsupervised segmentation process does not assign any labels to the found segments.

$$asp = \frac{1}{N}\sum_{i=1}^{N_s} p_{i\bullet} \times n_{i\bullet} \quad , \qquad aap = \frac{1}{N}\sum_{j=1}^{N_a} p_{\bullet j} \times n_{\bullet j} \quad ,$$

$$Q = \sqrt{asp \times aap} \quad .$$

with

n_{ij} = *total number of observations in segment i by activity j*

$n_{i\bullet}$ = *total number of observations in segment i*

$n_{\bullet j}$ = *total number of observations of activity j*

N_a = *total number of activities*

N_s = *total number of segments*

N = *total number of observations*

$$p_{i\bullet} = \sum_{j=1}^{N_a} \frac{n_{ij}^2}{n_{i\bullet}^2}$$

$$p_{\bullet j} = \sum_{i=1}^{N_s} \frac{n_{ij}^2}{n_{\bullet j}^2}$$

Figure 6.7: Average segment purity (asp), average activity purity (aap) and the overall criterion Q

In order to measure segment quality, we use the three measures proposed by Zhang et al. [101]: average segment purity (asp), average activity purity (aap) and the overall criterion Q (Figure 6.7). asp, aap and Q measure the quality of the segmentation based on purity of the found segments and labeled segments. The asp measures the purity of one segment with regard to the labeled activities, i.e. the asp indicates how well one segment is limited to only one activity. The aap measures the purity of one activity with regard to the detected segments, i.e. the aap indicates to which extent one labeled activity corresponds to only one detected segment. The Q criterion is an overall evaluation criterion combining asp and aap.

asp, aap and Q values are comprised between 0 and 1, where larger values indicate better quality. In the ideal case (one segment for each labeled activity), $asp = aap = 1$ and $Q = 1$.

6.2.2 Short Small Group Meetings

Figure 6.8: Interaction group configurations for the small group meetings of 4 individuals

Five short meetings (duration: between 9 min. 14 sec. and 16 min. 12 sec.) with 4 participants have been recorded. The speech of each individual was recorded using a lapel microphone. The use of lapel microphones has been admitted in order to minimize correlation errors of speech

activity of different individuals, i.e. speech of individual A is detected as speech of individual B. A real-time speech activity detector [20, 91] generated binary observation values (speaking, not speaking) for each individual that is recorded. These binary observations were combined to a 1-dimensional discrete observation code. The generated code comprises 2^n different values, where n is the number of recorded individuals. For the five short meetings with 4 individuals, the resulting observation code has $2^4 = 16$ values comprised between 0 and 15. The automatic speech detector has a sampling rate of 62.5 Hz, which corresponds to the generation of one observation every 16 milliseconds.

The individuals formed different interaction groups during the meetings (Figure 6.8). The number and order of group configurations, i.e. who will speak with whom, was fixed in advance for the experiments. The timestamps and durations of the group configurations were, however, not predefined and changed spontaneously. The individuals were free to move and to discuss any topic.

Figure 6.9 shows the labeled group configurations for each small group meeting as well as the segments detected by the proposed approach. Table 6.1 indicates the asp, aap and Q values for each meeting as well as the average of these values for all meetings. Unlike meeting recordings 1, 4 and 5, recordings 2 and 3 contain numerous wrong speech activity detections caused by correlation errors and microphone malfunctions, which explains lower asp and Q values. However, the overall results of the proposed approach are very good; the average Q value is 0.85.

	Duration	**asp**	**aap**	**Q**
Meeting 1	9m 14s	0.94	0.93	0.93
Meeting 2	10m 14s	0.68	0.99	0.82
Meeting 3	16m 11s	0.66	0.86	0.75
Meeting 4	14m 47s	0.78	0.91	0.85
Meeting 5	16m 12s	0.93	0.92	0.92
Average		0.80	0.92	0.85

Table 6.1: asp, aap and Q values for the small group meetings

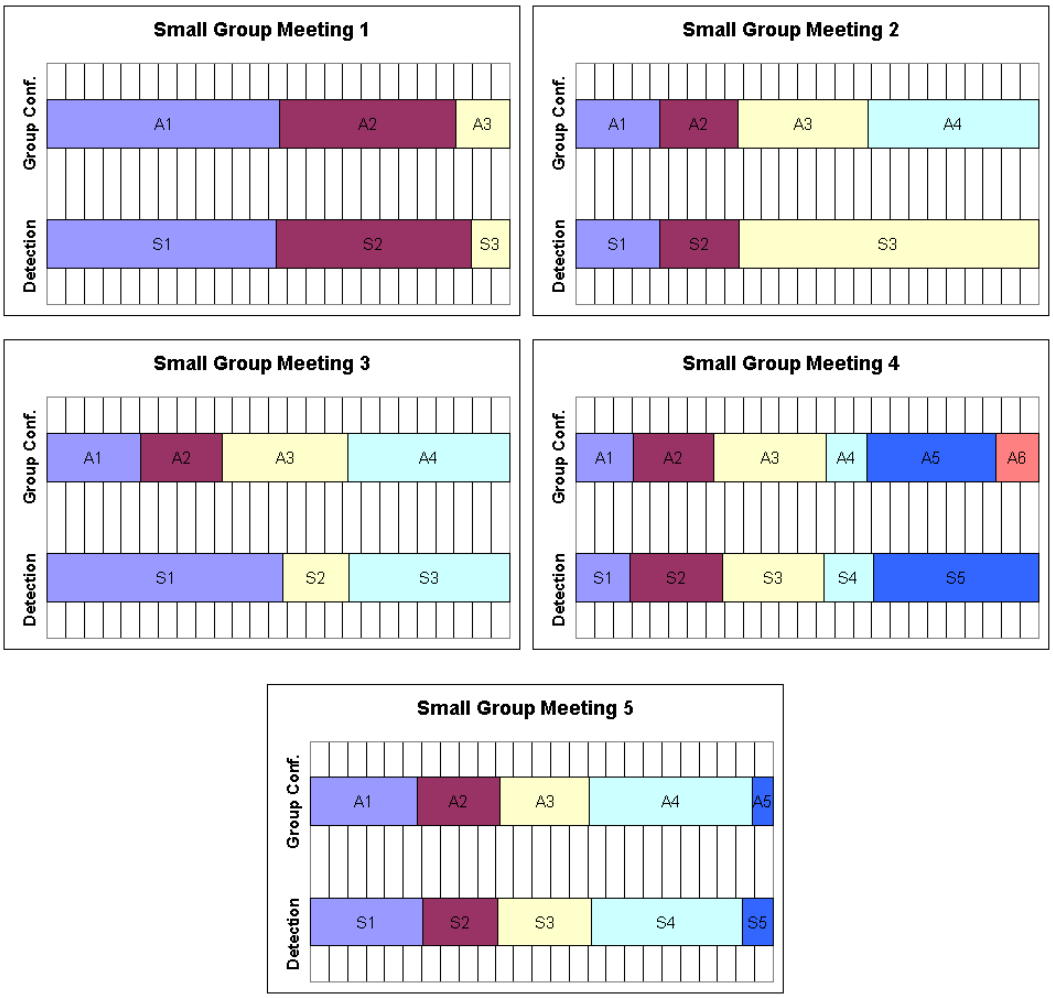


Figure 6.9: Meeting 1-5: group configurations and their detection

6.2.3 Seminar

A seminar (duration: 25 min. 2 sec.) with 5 participants has been recorded. As for the small group meetings, the speech of the participants was recorded using lapel microphones. The automatic speech detector provided the speech activity observations (speaking, not speaking) for each individual. These observations were combined to an observation code with 2^5 different values comprised between 0 and 31. The activities during the seminar were "discussion in small groups" (D1), "presentation" (P), "questions" (Q) and "discussion in small groups" (D2). Figure 6.10 shows the labeled activities for the seminar as well as the segments detected by the proposed approach. Table 6.2 indicates the asp, aap and Q value. The results of the automatic segmentation are very good; the obtained Q value is 0.90.

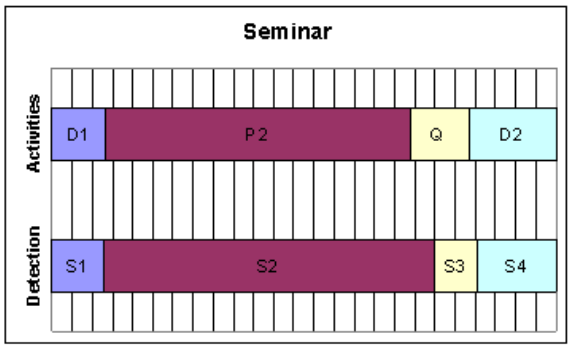


Figure 6.10: Seminar: activities and their detection

	Duration	asp	aap	Q
Seminar	25m 2s	0.88	0.91	0.90

Table 6.2: asp, aap and Q values for the seminar

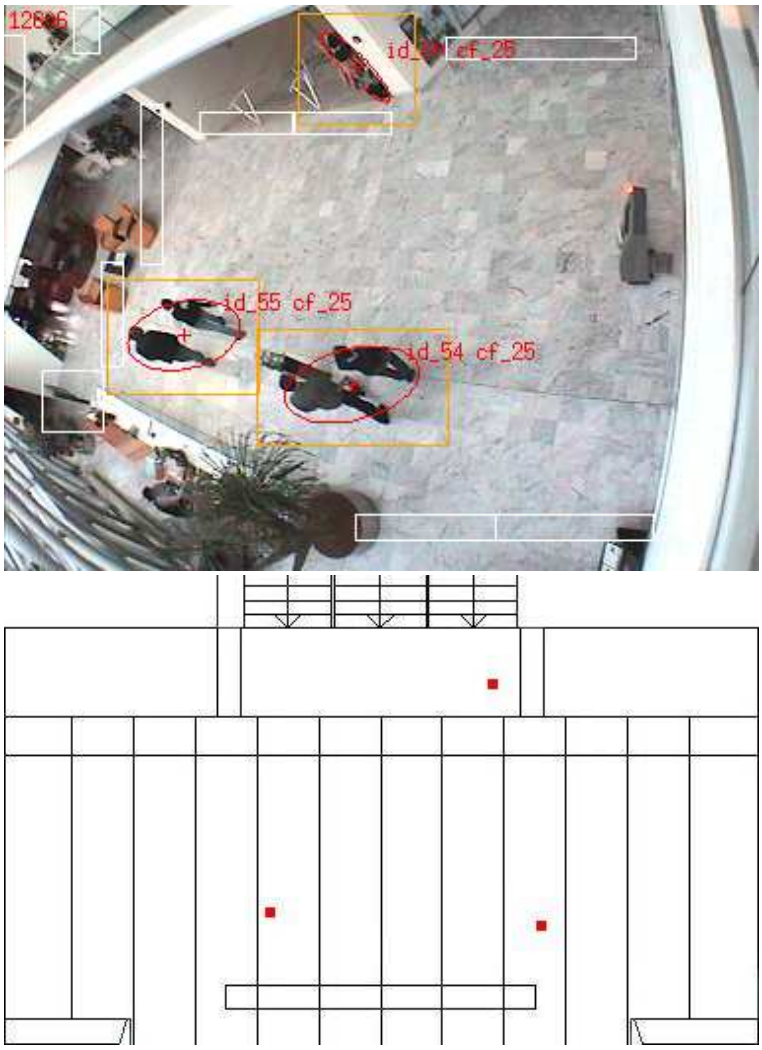


Figure 6.11: Wide-angle camera image of INRIA Rhône-Alpes entrance hall with three targets being tracked (above) and the corresponding target positions on the hall map after applying a homography (below). White rectangles in the camera image (above) indicate the detection zones used by the visual tracker for creating new targets

6.2.4 Cocktail Party Meeting

A cocktail party meeting (duration: 30 min. 26 sec.) with 5 participants has been recorded in the entrance hall of INRIA Rhône-Alpes. The recording was multimodal, including audio and video information. The speech of the participants was recorded using headset microphones. A wide-angle camera filmed the scene and a visual tracking system provided targets corresponding to individuals or small groups (Figure 6.11 above). Our method has been applied to the audio, video and audiovisual information of the recording.

The audio of each individual has been recorded using lapel microphones. As for the small group meetings and the seminar, the audio channels of the different lapel microphones have been analyzed by a speech activity detector providing binary speech activity observations (speaking, not speaking) for each individual. These binary values are combined to an audio observation code ($2^5 = 32$ values between 0 and 31) generated every 16 milliseconds.

The visual tracking system [24] is based on background subtraction and creates and tracks targets based on the video images of the wide-angle camera. The detected targets may correspond to individuals or small groups. The split and merge of these targets made it difficult to track small interaction groups directly, in particular when interaction groups are near to each other. In order to generate visual observation codes, the positions of the targets need to be discretized. First, the targets tracked by the visual tracking system have been mapped on the hall map using a homography. A homography defines a relation between two figures, such that to any point in one figure corresponds one and only one point in the other, and vice versa. Figure 6.11 below shows the three points on the hall map corresponding to the three targets currently tracked by the visual tracking system (Figure 6.11 above). Then, a multidimensional EM clustering algorithm [14] has been applied to all target positions on the hall map as well as the angle and the ratio of first and second axis of the bounding ellipses of all targets. The EM algorithm was initially run with a high number of possible clusters, while constantly eliminating those with too weak contribution to the whole model. 27 clusters were identified for the cocktail party recording. Figure 6.12 indicates the positions of all targets (red dots on the hall map) as well as the clusters learned by EM (small blue ellipses on the hall map). Finally, the visual observations are generated based on the dominant position clusters in the current video frame. The dominant position clusters are the clusters of the EM model with the highest probability of having generated the targets in the current video frame. The number of visual observations is limited to the number of clusters (here: 27). The appearance of a dominant cluster in a video frame is counted as one observation, thus augmenting the frequency of this cluster in the histograms. The tracking system has a frame rate of 16 frames per second, which corresponds to the generation of visual observation codes every 62.5 milliseconds.

The histograms of the proposed approach are calculated for the audio observations coming from the speech activity detector as well as for the visual observations coming from the visual tracker.

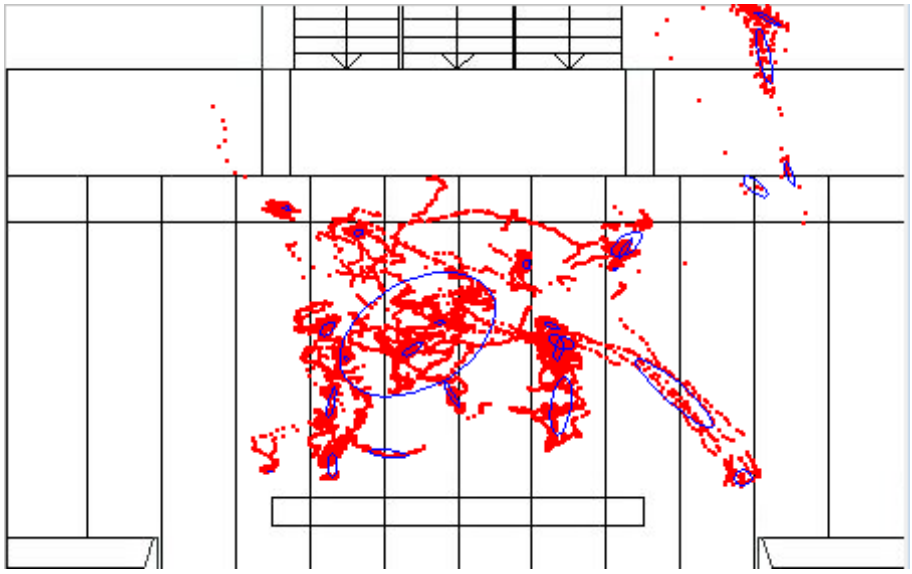


Figure 6.12: Cocktail party: positions of all targets on the hall map (red dots) and 27 position clusters isolated by EM algorithm (small blue ellipses)

The fusion is done by simply summing the Jeffrey divergence values of the audio observation histograms and the visual observation histograms. Summing the Jeffrey divergence values of the histograms from different modalities is an easy and efficient way to fuse multimodal information because no data conversions or additional fusion calculations are necessary.

The participants formed different interaction groups during the cocktail party meeting. The interaction group configurations were labeled. Figure 6.13 shows the labeled group configurations as well as the segments detected by the proposed approach. The approach has been applied to the speech detector observations (Figure 6.13 top left), the visual model observations (Figure 6.13 top right), and both the speech detector and the visual model observations (Figure 6.13 bottom). Table 6.3 indicates the corresponding asp, aap and Q values. The results of the audio segmentation were very good in the beginning of the cocktail party, but degraded afterwards due to less regulation in speech contributions of the participants and correlation errors of the microphones.

	Duration	**asp**	**aap**	**Q**
Audio	30m 26s	0.57	0.83	0.70
Video	30m 26s	0.83	0.92	0.87
Audio+Video	30m 26s	0.94	0.94	0.94

Table 6.3: asp, aap and Q values for the cocktail party

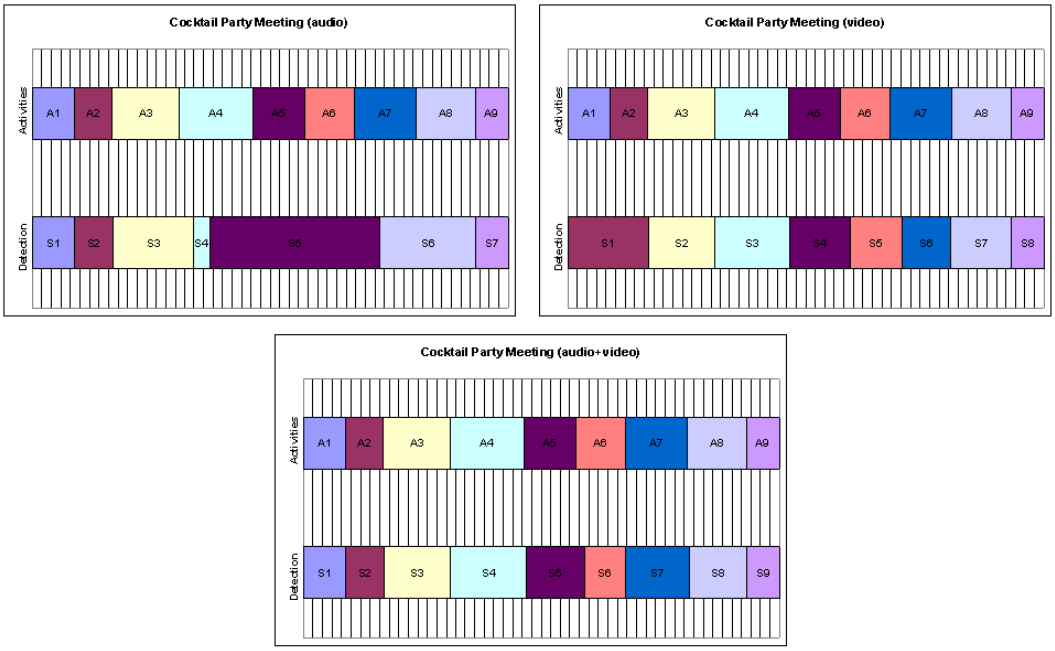


Figure 6.13: Cocktail party: group configurations and their detection based on audio, video and audiovisual data

The results of the visual segmentation are very good because of the fact that participants forming an interaction group tend to separate from other interaction groups in the environment. However, distinct interaction groups do not always separate in the environment, which leads to detection errors in the beginning of the meeting. The results of the segmentation of both audio and video are very good, outperforming the separate segmentations. The Q value of the video and audio segmentation is 0.94.

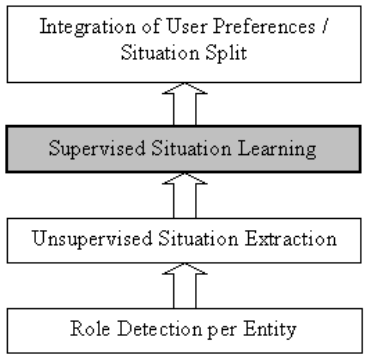
6.3 Conclusions

This chapter proposed an approach for detecting small group configurations and activities from multimodal observations. The approach is based on an unsupervised method for segmenting meeting observations coming from multiple sensors. The Jeffrey divergence between histograms of meeting activity observations is calculated. The peaks of the Jeffrey divergence curve are used to separate distinct distributions of meeting activity observations. These distinct distributions can be interpreted as distinct segments of group configuration and activity. The correspondence between the detected segments and labeled group configurations and activities has been measured for 7 small group recordings. The obtained results are promising, in particular as the method is completely unsupervised.

The fact that the proposed method is unsupervised is especially advantageous when analyzing meetings with an increasing number of participants (and thus possible group configurations) and a priori unknown activities. The method then provides a first segmentation of a meeting, separating distinct group configurations and activities. These detected segments can be used as input for learning and recognizing meeting situations and to build up a context model for a meeting. Additional meeting information will then be necessary to disambiguate all situations. Head orientation, pointing gestures or interpersonal distances seem to be good indicators. As described for the cocktail party meeting, the proposed approach can easily be extended to integrate further meeting information coming from different sensors.

Chapter 7

Supervised Learning of Situations: injection of expert knowledge



This chapter addresses the supervised learning of situations. The proposed method constitutes the third layer of the framework for acquiring and evolving situation models. Supervised Situation learning is based on the segments extracted by unsupervised situation discovery (chapter 6) for each situation. An expert provides situation labels for each of these segments. The objective is then to learn a representation for each situation from given segments of observations and the associated situation labels. The learned situation representations and constructed situation model can later be adapted according to user feedback.

A situation is a temporal state describing activities and relations of detected entities (persons) in an augmented environment. Perceptual information from the different sensors in the environment is associated to the situations. The different situations are connected within a network. A path in this network describes behavior in the scene. System services to provide can be associated to the different situations in the network.

To the best of our knowledge, the research work on situation learning and recognition has just begun. Most of the existing applications construct the situations manually and apply them to a specific application in an *ad hoc* manner. There is no systematic methodology. We aim at solving two fundamental issues: learning and recognizing a situation representation, and learning the relationship between situations:

Representation of Situation

The input to the situation learning is n sequences of perceptions P_i associated to m situation labels ($m \leq n$). Each sequence corresponds to one situation. Two or more sequences can have the same situation label. The situation acquisition algorithm produces or learns the situation representations from the given perception sequences and associated labels. The output will be the representation of each situation to be used for situation recognition. A supervised learning methodology will be adopted. The graphical representation illustrating the abstract idea of this process is shown in Figure 7.1. We aim at finding a robust and discriminative representation for each situation. The difficulty in this process is to define the criteria to measure the robustness and discriminative power of the representation. One possible way is to employ Fisher's criterion [50].

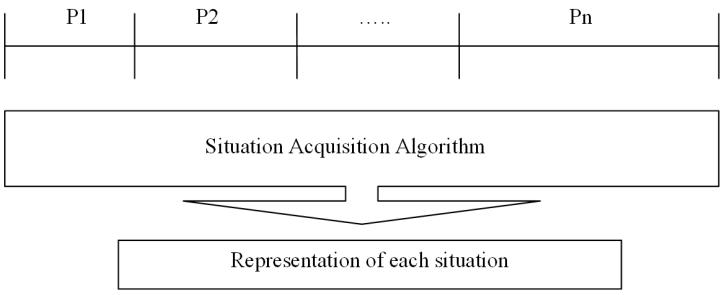


Figure 7.1: Learning of situations

Recognition of Situation

Once the situation representation algorithm is developed, recognition of situation can be proceeded. The testing perception sequences will go through situation recognition. The situation in the testing sequence will be recognized by comparing the situation representation for the testing sequence and the reference situations.

Context Understanding: Situation Relationship

Context is represented by a set of relationships between situations. The system needs to be trained with different categories of context (e.g. based on spatial or temporal relationships [54]) in order to find the most appropriate situation relationships. Once the situations in the testing sequences have been recognized, the relationships between these situations, and thus context, can be found.

In the following, we propose a Supervised Situation Acquisition Algorithm taking perception sequences as input and generating situation representations as output. The proposed generic algorithm can be used with different learning methods and be applied to many different applications. An example application to a video surveillance task is presented and evaluated.

7.1 Method

7.1.1 Supervised Situation Acquisition Algorithm

The objective of the situation acquisition algorithm is to find a representation for each situation such that this representation is the most discriminative with regard to the given perception sequences (entity/role/relation values). Based on the concept of Fisher's Criterion [50], we would thus like to determine a representation for each situation such that the ratio of between-situation distance and within-situation distance is maximized. We have:

- m sequences associated to n situation labels $(m \leq n)$
- Each sequence corresponds to one situation
- Two or more sequences can have the same situation label
- Each sequence contains entity/role/relation values, called perceptions, $\{P_1, P_2, ..., P_k\}$

To convert a sequence of perceptions to a situation representation, we introduce the notion of learner L: $\{P_1, P_2, ..., P_k | k > 0\} \mapsto$ *situation representation* S. The learner incorporates the learning method taking the perception sequences as input and generating a learned situation representation as output. As the learning method and the generated situation representation are

interchangeable, we can distinguish different learner classes corresponding to different learning methods (e.g. hidden Markov model learners, decision tree learners etc.). An instance of each learner class, i.e. a learner, corresponds to a particular parameterization of the learning method (e.g. EM learning a HMM with 5 states). The ratio of between-situation and within-situation distance is defined for each learner class. A general ratio needs to be defined when comparing different learner classes (step B of algorithm in Figure 7.2). An example of a general ratio is to calculate for each learner the ratio of the percentage of perceptions correctly classified as not corresponding to the situation and the percentage of perceptions incorrectly classified as corresponding to the situation.

```
A. For each learner class do:
    d. {optimization step}
       For each situation label do:
          • Select learner/set of learners
          • Apply learner to given perceptions
    e. {validation step}
       Calculate ratio of between-situation and within-
       situation distance
    f. Repeat a.-b. until optimal ratio is obtained

B. Choose learner class with best ratio of between-
   situation and within-situation distance
```

Figure 7.2: Generic iterative situation acquisition algorithm

The generic iterative algorithm for acquiring situation representations is designed and shown in Figure 7.2. It consists of an iteration over the available learner classes, which corresponds to the use and evaluation of different learning methods. The core algorithm contains an optimization step and a validation step. The optimization step first chooses a learner or a set of successive learners for each situation label. This choice can be done using exhaustive search over all learners of the class or following some criteria like gradient descent-based algorithm or information gain [66] . The chosen learner is applied to the perceptions of the situation label, generating a situation representation. Once a situation representation is generated for each label, the validation step calculates the ratio of between-situation and within-situation distance with respect to the given perceptions and generated situation representations. Optimization and validation step are repeated until optimal ratio is obtained. Finally, when applying several learner classes, the learner class with the best ratio is chosen. It is important to point out that the proposed situation acquisition algorithm is general (not like the current *ad hoc* and *domain specific* approaches). We believe that it can be applied to most, if not all applications.

7.2 Evaluation and Results

In this section, we want to provide an example application of the generic situation acquisition algorithm and its evaluation. The application addresses learning and recognizing situations for video surveillance and is mainly based on computer vision input, i.e. data coming from a real-time tracking system and extracted features. Learning and recognition are conducted per video sequence. As hidden Markov models are well adapted for learning and representing temporal observation sequences, we used hidden Markov models for learning and representing the situations.

7.2.1 Learning Situations for Video Surveillance

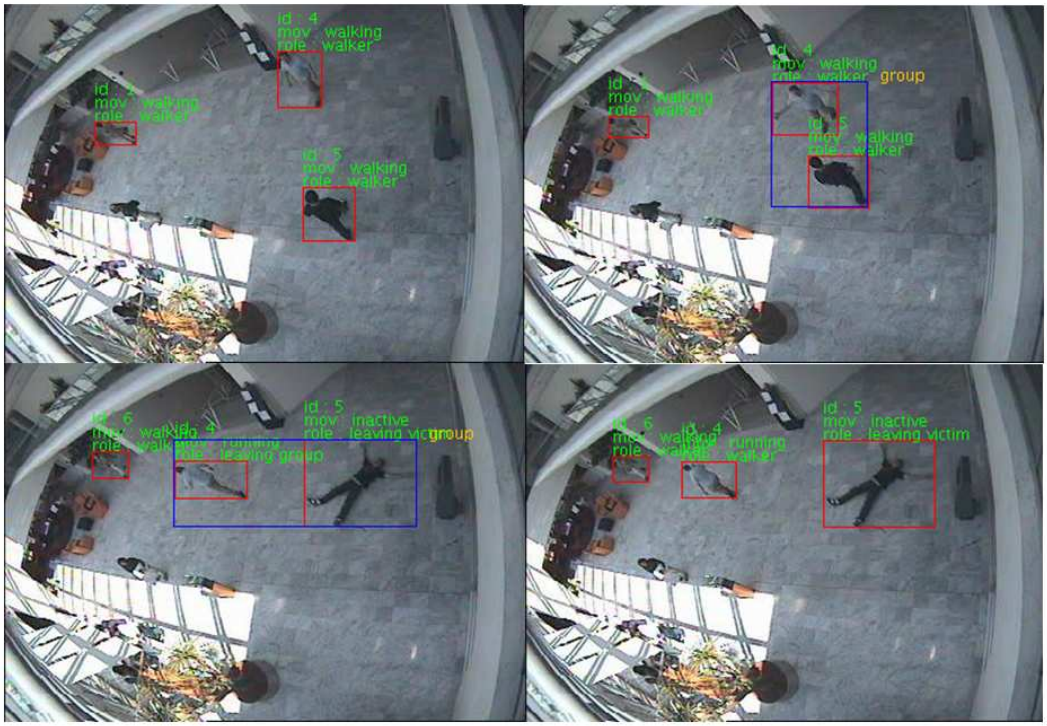


Figure 7.3: Four frames from a CAVIAR video clip

This example shows the application of the situation acquisition to video surveillance. The CAVIAR video clips [25] are used for evaluation and show different situations (“walking”, “browsing”, “fighting”, “waiting”, “object left”). Each video is associated with an XML file describing for each frame the entities with their position, movement, role (“walker”, “browser”, “none”, “fighter”, “leaving group”, “leaving victim”, “leaving object”) and group information

(binary value indicating whether an entity is in group relation with other entities). These files were created manually and used as the *ground truth* of the videos. Figure 7.3 shows four frames from one of the CAVIAR video clips where two people are walking, forming a group, fighting and running/leaving. Their roles are therefore “walker”, “fighter”, “leaving group”, and “leaving victim”. The “fighting” situation would for example involve these roles consecutively played by entities as well as the “group” relation.

Entities and roles provided by the *ground truth* can be acquired using the methods proposed in chapter 5. In the following, we will focus on the acquisition of the different situations applying the situation acquisition algorithm scheme. The perceptions consist of the entity, role and group values provided by the ground truth for the different situation labels.

A. For EM learner class do:
   a. *{optimization step}*
      For each situation label do:
      - Select a number of states for EM training
      - Learn HMM parameters with EM
   b. *{validation step}*
      Calculate ratio of perception sequence probability within and outside correct situation
   c. Repeat a.-b. until optimal ratio is obtained

Figure 7.4: Iterative situation acquisition algorithm using EM learner class

We choose hidden Markov models to represent the situations. The only learner class is thus the expectation maximization algorithm (EM) for learning hidden Markov model parameters. The instances of the class are the possible number of states of the HMM to learn. We use the ratio of perception sequence probability within the correct situation (represented by HMM) and perception probability outside the correct situation (represented by HMMs of other situations). The iterative situation acquisition algorithm using EM as learner class is shown in Figure 7.4.

To evaluate, we use 114 perception sequences extracted from the CAVIAR XML files. We did a 3-fold cross-validation, using one third of the sequences for training and two third of the sequences for testing. The confusion matrix and information retrieval statistics as well as the number of states of the HMMs learned for the different situation labels are indicated in Table 7.1. The obtained average error is 6.22 %, with a standard deviation of 2.07 % [21].

The obtained results for situation recognition are better than those for role recognition (see chapter 5). This can be explained by the fact that we assume low-level information (like roles and relations) to be provided (e.g. by lower-level recognition processes or given ground truth),

	Walking	Browsing	Fighting	Waiting	Object left
Walking (3 states)	0.8889	0	0.0635	0.0476	0
Browsing (3 states)	0.1944	0.8056	0	0	0
Fighting (7 states)	0	0	0.9167	0.0833	0
Waiting (7 states)	0.0476	0	0.1429	0.8095	0
Object left (5 states)	0	0	0	0	1

Class	TP rate	FP rate	Precision	Recall	F-measure
Walking (3 states)	0.8889	0.0692	0.9446	0.8889	0.9147
Browsing (3 states)	0.8056	0	1	0.8056	0.8857
Fighting (7 states)	0.9167	0.0695	0.6127	0.9167	0.7313
Waiting (7 states)	0.8095	0.0444	0.8519	0.8095	0.8194
Object left (5 states)	1	0	1	1	1
Total	0.8841	0.0366	0.8818	0.8841	0.8702

Table 7.1: Confusion matrix and information retrieval statistics of the CAVIAR data sets, with the number of states of the learned HMMs

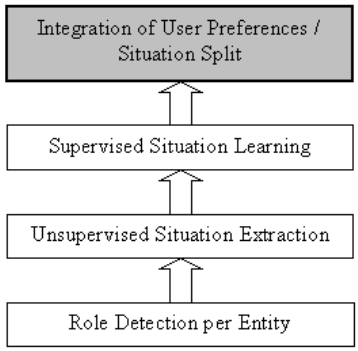
which improves situation recognition results considerably. Further, role recognition refers to the detection of activities of individuals for each frame, while situation recognition refers to detecting activities of one or more individuals for frame sequences. Of course, recognition per frame sequence is more discriminative than framewise recognition. However, both recognition processes are connected and can refer to similar activities on different levels of elaborateness ("walking" refers to a role (chapter 5) as well as "walking" to a situation(chapter 7)).

7.3 Conclusions

In this chapter, we proposed a generic Situation Acquisition Algorithm for supervised learning of situations. The algorithm takes perception sequences (e.g. entity/role/relation values) as well as the associated situation labels as input and generates a situation representation for each label. As the proposed algorithm is a generic schema, the employed learning methods as well as the discrimination criteria are interchangeable. The proposed algorithm has been applied with success to a video surveillance task.

Chapter 8

Adapting to User Preferences



This chapter addresses the adaptation of an initial situation model to user preferences driven by feedback on executed system services. The proposed method constitutes the last layer of the framework for acquiring and evolving situation models. The initial situation model can either be predefined by a human engineer or be constructed automatically by the methods proposed in chapter 6 and chapter 7. These methods are intended to be used for:

1. segmenting a multimodal observation stream
2. supervised learning of the situations based on extracted observation segments and expert feedback

Once an initial situation model has been defined (or learned), the learning process must adapt the situation model according to given feedback on the system services. As in the chapters before, we will focus on the adaptation of the situations as well as the situation network and the associated system services.

The input to the algorithm is a predefined situation network along with feedback from prior use. We want to minimize the frequency with which the system offers inappropriate services, while minimizing disruption. This means that the feedback given to the system is to be minimal to achieve the wanted changes of system services. We assume that a person, denoted supervisor in the following, is capable of specifying system services to be executed by the system and that his feedback is always consistent. The user himself or another person can act as this supervisor. We distinguish three forms of supervisor feedback:

1. (Re)action correction: the service or (re)action executed by the system is wrong and a different service must be executed instead. The supervisor gives the different system (re)action as feedback to the system. This includes the case where the supervisor wants the system to execute a (re)action while the system does not execute anything.

2. (Re)action deletion: the service executed by the system is wrong and no system service must be executed instead. The supervisor gives a particular (re)action, the "erase" (re)action, as feedback to the system.

3. (Re)action preservation: the service executed by the system is correct. The supervisor does not give any information to the system. As we assume that the supervisor is always consistent, we can interpret the absence of his corrections or deletions as positive feedback for the currently executed system (re)actions.

8.1 Method

Figure 8.1 shows an overview of the proposed algorithm. The input of the algorithm is a predefined situation network and feedback given by the supervisor. The supervisor corrects, deletes or preserves the service executed by the system while acting or observing a user in the environment. Each correction, deletion, or preservation generates a training example for the learning algorithm containing current situation, roles and relations configuration, and the (correct) service. The differences between the services given in the training examples and the services provided in the predefined situation network will drive the different steps of the algorithm.

In the first step, the algorithm tries to directly modify the services associated to the situations using the existing situation network. If service A is associated to situation S, and all training examples indicate that service B must be executed instead of A, then B is associated to S and the association between A and S is deleted. If successful, the result is an adapted situation network integrating the supervisor wishes. No situation split is necessary.

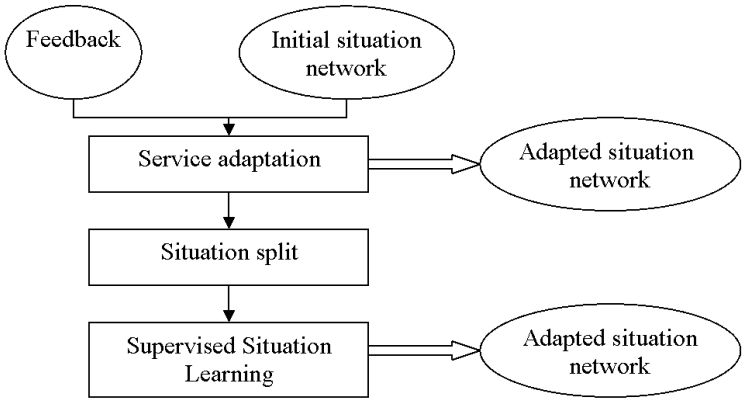


Figure 8.1: Overview of the different steps of the algorithm

In the second step, if the feedback indicates further that the concerned situation is too general, the algorithm splits the situation into sub-situations. The situation split is executed when the supervisor perceives several situations (expressed by several disjunctive services in the training examples) while the predefined situation network only perceives one situation (expressed by one service). Thus the situation perceived by the predefined situation network may be too general and the algorithm tries to split it in sub-situations. The observations (e.g. role, relations configurations) of these sub-situations needs to be determined according to the given training examples (see Splitting Situations).

Splitting Situations

When splitting situations, a number of training examples indicate different services for one situation of the predefined situation network. Several sub-situations need to be created for these services. We must determine the characteristic observations (e.g. characteristic role, relation configurations) in order to distinguish or detect these sub-situations.

The determination of the characteristic observations of the sub-situations can be seen as classification problem. The service labels of the training examples can be interpreted as class labels. For each class, we need to identify the characteristic observations (e.g. characteristic role, relation configurations) necessary to detect the corresponding sub-situation (Figure 8.2 above). We assume that only one situation can be active at a time point (no parallelism) in order to keep our learning problem manageable. The observation sequence of the initial situation is then split into sequential sub-sequences of observations corresponding to the new sub-situations (Figure 8.2 below). The start and end points of these sub-sequences are determined by the time points of the feedback on system service execution. For example, an initial situation “working in of-

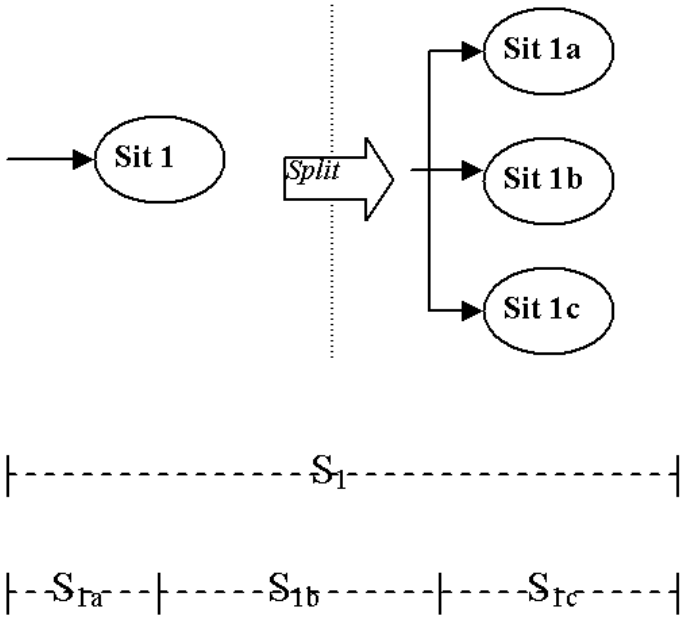


Figure 8.2: Splitting situations: Sit_1 is split into Sit_{1a},Sit_{1b} and Sit_{1c}; the observation sequence S_1 used for constructing the situation representation of Sit_1 is also split

fice" could be split into sub-situations "working on PC", "reading papers" and "being on the phone". The time points for the sub-sequences can be determined by the feedback on the services "switch on PC lamp", "switch on music for reading", and "switch on phone recorder". Then, for each new sub-situation, we can determine or construct a discriminative representation based on the given observation sequence.

```
A. For each learner class do:
    d. {optimization step}
       For each situation label do:
          • Select learner/set of learners
          • Apply learner to given perceptions
    e. {validation step}
       Calculate ratio of between-situation and within-
       situation distance
    f. Repeat a.-b. until optimal ratio is obtained

B. Choose learner class with best ratio of between-
   situation and within-situation distance
```

Figure 8.3: Generic iterative situation acquisition algorithm

The supervised learning scheme (Figure 8.3) proposed in chapter 7 can again be adopted to associate observations to the new sub-situations. The learning algorithms used as learner classes can be exchanged. Possible learner classes include decision tree learners, Bayesian classifiers, hidden Markov models and others. Note that the situation split and the associated situation acquisition scheme are a methodology for integrating user preferences. The proposed abstract methodology has been tested with implementations based on conceptual learning algorithms (Find-S and Candidate Elimination) and decision tree algorithm ID3 (see section 8.2) as well as hidden Markov models (see chapter 9).

8.2 Evaluation and Results

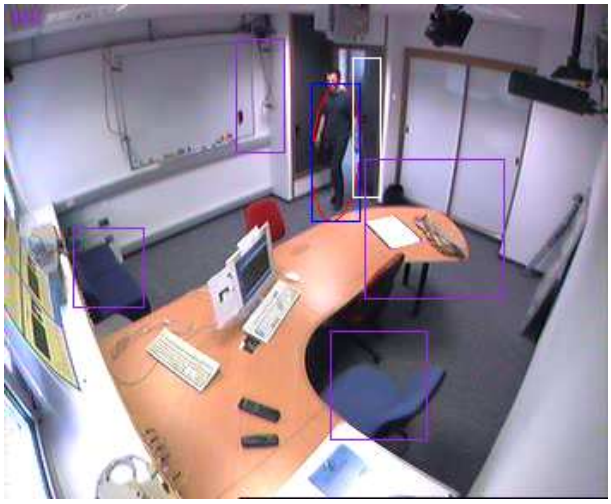


Figure 8.4: Video image of the wide-angle camera of SmartOffice. A white box next to the door is used for the creation of new targets (entities). One person is currently tracked. Four presence detection zones (chair, couch, board and table) of the tracking system are indicated

A situation model for office activity within the SmartOffice environment [60] of the PRIMA group has been manually designed and implemented (Figure 8.5). In this environment, entities are created and tracked by a robust tracking system [24]. The position of the created entities determines several roles like “comes_in” or “works_on_PC” (Figure 8.4). Additional roles are determined by the login of an entity (person) to a computer in the environment or specific appointments marked in the agenda of the logged person. The “not_same_entity_as” relation is used to distinguish entities in the environment. The services of the system are based on the control of the Linux music player and the projection of different messages or presentations on different surfaces in the environment.

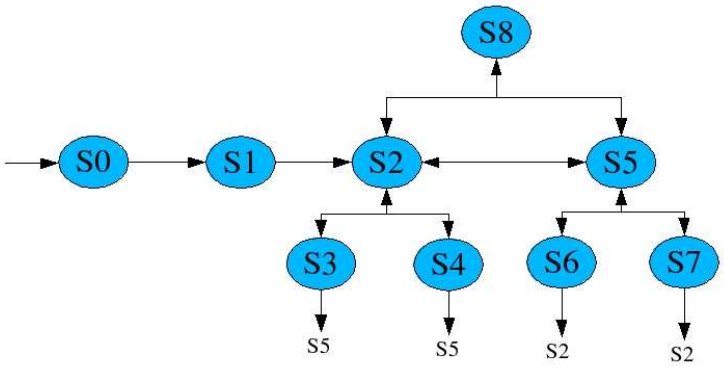


Figure 8.5: Situation model of the SmartOffice environment. Important Situations are **S0** (empty room), **S1** (newcomer enters SmartOffice), **S2** (Person connects/works on PC), **S5** (Connected Person sits on couch) and **S8** (Presentation in SmartOffice). Additional situations refer to agenda information like having a scheduled meeting on campus (while working on PC **S3**, or sitting on couch **S6**), or having a meeting in SmartOffice (while working on PC **S4**, or sitting on couch **S7**)

Because our SmartOffice situation model is defined by a finite number of available roles and relations, the situations within this model can be represented as a fixed-sized vector containing one 0/1 value for each available role and several 0/1 values for each available relation. The value 1 means that the corresponding role or relation is valid; the value 0 means that the role or relation is not valid. As a relation is applied on entities playing roles, it is represented by one 1/0 value for each different role combination it can be applied to. A characteristic role, relation configuration for one situation may contain blanks ("-") for those roles or relations that are not characteristic for this situation. A training example contains a vector with specific values reflecting the current role, relation configuration (observations) when recording the training example and the corresponding service (given by the supervisor). For example, a training example based on roles "comes_in", "works_on_PC" and relation "not_same_entity_as" would correspond to a vector of $2 + 3 = 5$ 0/1 values (two role values and $\frac{2*(2+1)}{2}$ relation values) and the associated service given as feedback.

We consider conceptual learning algorithms Find-S ([66, chapter 2.4]) and Candidate Elimination ([66, chapter 2.5]) as well as decision tree learning method ID3 [78] for learning the representations of the new sub-situations.

The conceptual learning method Find-S constructs the most specific hypothesis for each service based on the role, relation configurations (observations) in the given training examples (Figure 8.6). The idea is to start with one training example and then to generalize the values (by putting "-") of this example in order to cover all other training examples for the concerned service. The resulting hypotheses for the created sub-situations often contain, however, specific values for

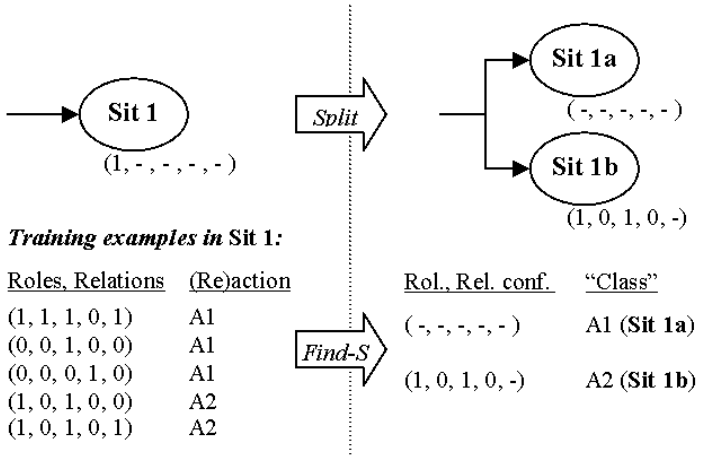


Figure 8.6: Splitting Situation using Find-S

the existence or non-existence of roles or relations that are not necessary or characteristic. As a consequence, small variations in the role, relation configuration may not be covered by the created sub-situations because their hypotheses are too specific.

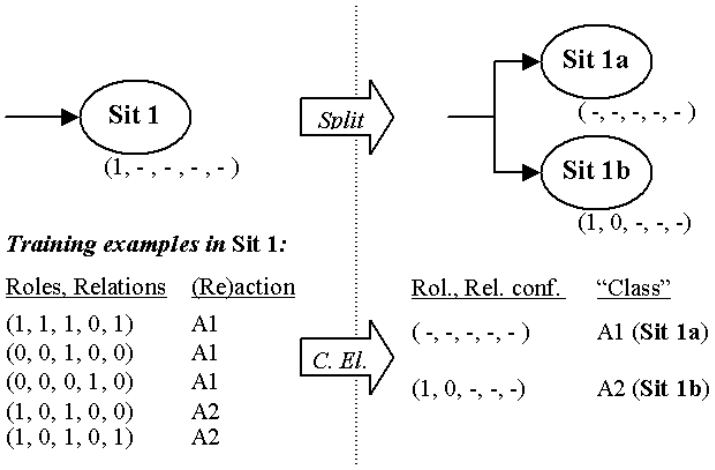


Figure 8.7: Splitting Situation using Candidate Elimination

To produce more general hypotheses for the sub-situations, we consider the conceptual learning algorithm Candidate Elimination. This algorithm constructs the most specific and the most general hypotheses for each service based on the role, relation configurations (observations) in the given training examples. The most general hypotheses for one service refer to the concepts (role, relation values) that are shared by all training examples for this service, but not by the

training examples of all other services. By combining the most general hypotheses for each service, we construct the representation, i.e. the role, relation configuration, for the corresponding sub-situations (Figure 8.7).

Both algorithms Find-S and Candidate Elimination have, however, the restriction that they can only find one conjunctive concept for each service, i.e. if the training examples indicate that a service is to be executed in two different complementary role, relation configurations, Find-S and Candidate Elimination will fail to construct several hypotheses (and thus sub-situations) for this one service. This is due to the fact that neither algorithm can construct disjunctive hypotheses.

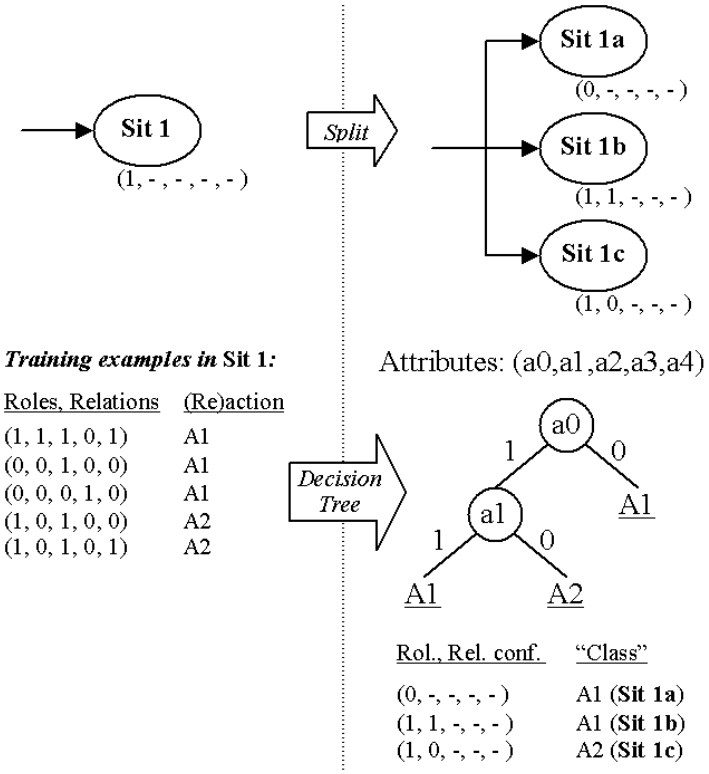


Figure 8.8: Splitting Situation using ID3

We consider decision tree learning method ID3, in order to address the limitation of conceptual learning methods. The idea is to construct a decision tree that classifies the different services found in the training examples of the initial situation (Figure 8.8). The attributes of this decision tree are the roles and relation values (0/1 values of the vector). Each leaf of the tree is labeled with a service (class). The path from the root of the tree to the leaf gives the representation, i.e. the characteristic role, relation configuration, for the sub-situation to be created for this (re)action. We can have several leaves with the same service, which corresponds to the creation

of several sub-situations for this service (disjunctive hypotheses).

The learning algorithms run on data base tables containing a representation of the current situation network and the training examples. A control process programmed in the forward chaining rule programming environment Jess [56] is used to execute the situation network. This situation network represented by rules is automatically generated from the data base tables of the learning algorithms. The supervisor feedback cannot be given while the user is acting in the environment (i.e. while the control process is running). Thus the control process and the learning algorithms are run sequentially and not in parallel.

To evaluate our method, two experiments have been executed on the predefined context model of the SmartOffice environment (Figure 8.5). The experiments have the same goal concerning the evolution of the system services. The supervisor gives feedback based on these goals during the experiments. As we focus on the correct execution of the system services, we do a cross-validation by adapting the predefined situation network using the supervisor feedback of the first experiment and by evaluating the second experiment on the adapted situation network (and inverse). The evaluation is done on the number of correctly classified training examples, i.e. correctly executed services, as well as on the review of the adaptations of the predefined situation network.

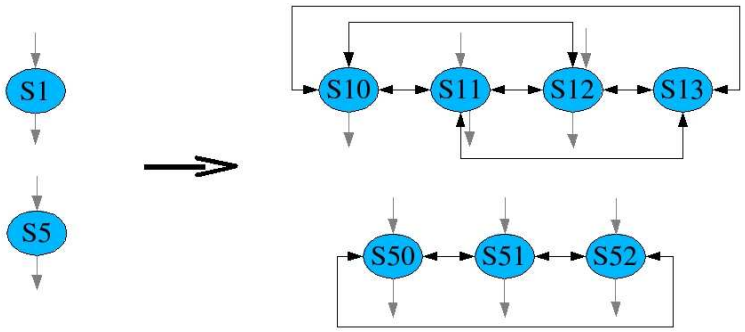


Figure 8.9: Structural adaptations performed on the predefined situation model of the SmartOffice environment by the method. Situations **S1** and **S5** have been split into sub-situations

The goal of both experiments was to integrate the correct turn-on and turn-off of the Linux music player depending on the activities (=roles, relations) of the user. The music player should be switched on when a newcomer sits on the couch to have a rest, and switched off when the newcomer starts speaking or leaves the couch (concerned situation: **S1**). The music player should similarly be switched on and off for a connected person (concerned situation: **S5**). Figure 8.9 shows the adaptations of the two concerned situations after the integration of the supervisor feedback. **S1** has been split into additional sub-situations integrating sitting down on couch (**S11**), speaking on couch (**S12**) and leaving couch (**S10**). **S13** is the sub-situation to which the initial service of **S1** is associated. The additional sub-situations of **S5** integrate leaving the couch

(**S50**), sitting down on the couch (**S51**) and speaking on couch (**S52**).

	A0	A8	A9
A0	0.6014	0.1884	0.2101
A8	0.25	0.75	0
A9	0.25	0	0.75

Class	TP rate	FP rate	Precision	Recall	F-measure
A0	0.6014	0.25	0.8704	0.6014	0.65
A8	0.75	0.0876	0.75	0.75	0.7286
A9	0.75	0.1178	0.7333	0.75	0.7302
Total	0.7005	0.1518	0.7846	0.7005	0.7029

Table 8.1: Confusion matrix and information retrieval statistics for Find-S

	A0	A8	A9
A0	0.6232	0.1884	0.1884
A8	0.5	0.5	0
A9	0.625	0	0.375

Class	TP rate	FP rate	Precision	Recall	F-measure
A0	0.6232	0.5714	0.4637	0.6232	0.5317
A8	0.5	0.0876	0.6667	0.5	0.5556
A9	0.375	0.1006	0.6667	0.375	0.4675
Total	0.4994	0.2532	0.5990	0.4994	0.5183

Table 8.2: Confusion matrix and information retrieval statistics for Candidate Elimination

	A0	A8	A9
A0	0.5985	0.1894	0.2121
A8	0	1	0
A9	0	0	1

Class	TP rate	FP rate	Precision	Recall	F-measure
A0	0.5985	0	1	0.5985	0.7134
A8	1	0.0881	0.8036	1	0.8901
A9	1	0.119	0.8	1	0.8889
Total	0.8662	0.0690	0.8679	0.8662	0.8308

Table 8.3: Confusion matrix and information retrieval statistics for decision tree algorithm ID3

Table 8.3, 8.2 and 8.1 show the results of the service execution in the form of confusion matrices and information retrieval statistics (A8 switches on the music player, A9 switches off the music

player, and A0 is the “do nothing” (re)action). In all experiments, the structural adaptation of the situation network corresponds to the expected changes. Concerning the correct classification of the training examples, i.e. the correct execution of the services, the decision tree algorithm (ID3) gives the best results. The improved results of decision tree approach are due to the fact that this algorithm supports disjunctive hypotheses. However, the decision tree algorithm tends to construct too general hypotheses for the sub-situations, which can lead to several inappropriate classifications. This is due to the fact that the decision tree algorithm prefers small trees to large trees, which means that general hypotheses are preferred to specific hypotheses for the sub-situations.

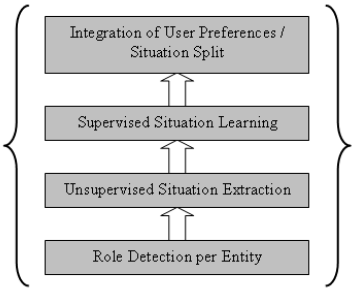
8.3 Conclusions

The method proposed in this chapter illustrates how a predefined (or learned) situation model can be evolved from user side by giving feedback on system service execution. The method relies on the situation split that creates sub-situations for a given initial situation. The moment for the split and the number of necessary sub-situations are determined by the feedback given on system service execution. The generic situation acquisition algorithm can be used to learn the representations of the created sub-situations. The proposed method has been tested in an augmented office environment using Find-S, Candidate Elimination and ID3 as learning algorithms. An expert modeled the augmented office as a situation network. This network is adapted according to feedback given by a supervisor. The results obtained for the test cases and employed learning algorithms are good, showing that the adaptation of a given situation model with feedback is possible. The system services desired by the human supervisor have been correctly integrated into the situation network structure. However, the precision of service execution for the created sub-situations is still not high enough in order to be acceptable to normal users.

Given feedback is sometimes not sufficient to decide which adaptation must be done to the situation network. Two different adaptations can cover the same (optimal) number of training examples (ambiguity of feedback). The two corresponding situation networks will, however, not have the same meaning for the user/supervisor. A possible solution is the extension of the learning to an interactive process. The learning system will verify ambiguous choices by asking the supervisor and the supervisor can intervene and correct when decisions of the learning system are wrong. However, one major drawback of the proposed method is that we assume that supervisor feedback is (more or less) consistent, which is not always the case in reality. Probabilistic or fuzzy learning algorithms can help to alleviate this drawback.

Chapter 9

Integration and Experimental Evaluation



In this chapter, we will propose an integration of the methods described in the precedent chapters into a whole system for an intelligent home environment. This system is to build up automatically and to evolve a situation model for human behavior in the scene. An initial situation model for the behavior in the scene in constructed using the segmentation of basic situations (chapter 6) and the supervised learning of situation labels (chapter 7). The resulting initial situation model is then evolved according to user preferences using feedback (chapter 8). In the following, we will first motivate the vision of a system for an intelligent home by a short example (section 9.1). Then, we will describe our current implementation comprising 3D tracking, role detection and observation generation (section 9.2). Finally, the conducted evaluations as well as the obtained results are described.

9.1 Motivation: Bob's dream...

Bob is dreaming of a new intelligent home. The new home provides services to make Bob's life easier and more convenient. Bob hopes to reduce all that technical stuff that he needs to

switch on/off, regulate, configure, etc. His ideal environment should provide entertainment and communication services with little or no configuration, adapting according to his preferences with a minimum of disruption and feedback. Bob should only need to indicate which service he wants and the system should adapt accordingly.

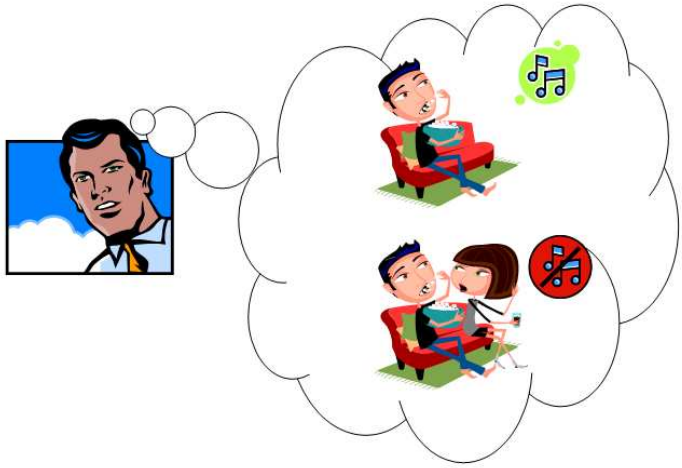


Figure 9.1: Bob's dream: an intelligent home anticipating his needs and wishes

For instance, Bob enjoys jazz music when he is eating on the couch, but he does not want to be disturbed when eating with his girlfriend (Figure 9.1). Bob is willing to give feedback for learning to the system by giving specific voice commands in the environment or even, if needed, by clicking on services to provide on his PDA.

To satisfy Bob, the environment needs to be equipped with visual and acoustic sensors. For example, video cameras and a video tracking system and microphone array with speech detection and recognition can provide basic information about Bob's current location and activity. Of course, hand crafting of detection routines is not sufficient for Bob's dream as he wants the system to evolve, constantly adapting to his preferences. Thus a general model of the environment needs to be designed and then adapted according to Bob's remarks. The situation model has proved to be very useful for this task, being applied to various problems and domains ([19] and chapter 4).

A situation is a temporal state describing activities and relations of detected entities (persons) in the environment. Perceptual information from the different sensors in the environment is associated to the situations. The different situations are connected within a network. A path in this network describes behavior in the scene. System services to provide are associated to the different situations in the network.

In order to build a situation model for Bob's home, we start with an initial model describing Bob's very basic behavior. This model can be seen as default configuration of the system for

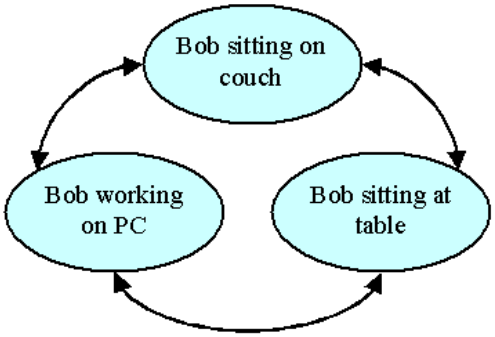


Figure 9.2: Initial situation model for Bob's home

the environment, being general and providing little detail. Figure 9.2 shows an example of such a default situation model for Bob's home. Normally, an expert constructs these models when setting up the system within environment. However, expert hours are expensive for Bob. So an expert should at least be aided by automatic processes.

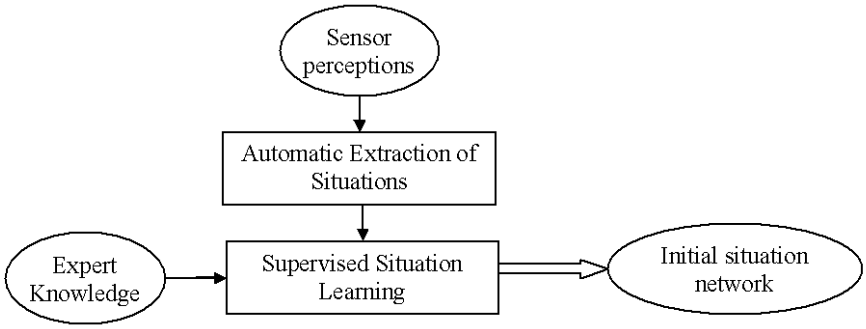


Figure 9.3: Overview of the process of creating an initial situation model

The schema of the process of creating an initial situation model is depicted in Figure 9.3. Given recorded sensor recordings of Bob's activity in the environment, the automatic extraction of situations (chapter 6) provides a first segmentation of sensor signals. We exploit the addition of human expertise only for providing the situation labels for the obtained segmentation. A supervised learning scheme (chapter 7) is used to associate the situation labels with the recorded perceptions.The outputs of the supervised situation learning scheme are the situation representations for the initial situation network. The connections between the situations are constructed by considering the recorded sensor perceptions and existing transitions between the detected situations.

The initial situation model is, however, simple, with insufficient detail about Bob's preferences. General situations, such as "Bob sitting on couch", must be refined to obtain sub-situations

incorporating the preferred system services. A possible adapted situation model could look like in Figure 9.4.

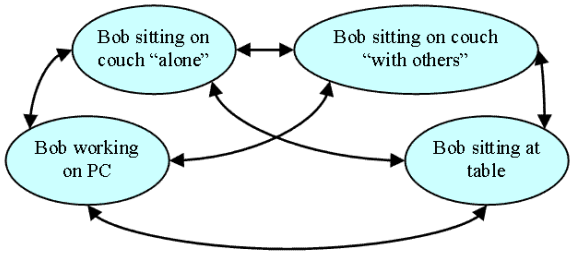


Figure 9.4: Adapted situation model for Bob's home

"Sitting on couch" has been split into "Sitting on couch -alone-" and "Sitting on couch -with others-". Sensor perceptions need to be associated to the new sub-situations. Bob is, however, not interested in recruiting a system engineer implementing the new sub-situations. This refining process, and in particular the association of sensor perceptions to the new sub-situations, should hence be as automatic as possible. The new sub-situations need thus to be learned from recorded sensor perceptions as well as Bob's given feedback (via his voice or PDA) using machine learning methods. The schema of the adaptation process can be seen in Figure 9.5. Bob's feedback and the initial situation network are the input; the output is the adapted situation network. The situation split proposed in chapter 8 can be adopted to refine and learn the new sub-situations. Once the sub-situations are learned, they are inserted into the whole network by eliminating conflicts and erasing obsolete situations. The result is an adapted situation network with new sub-situations integrating Bob's preferences.

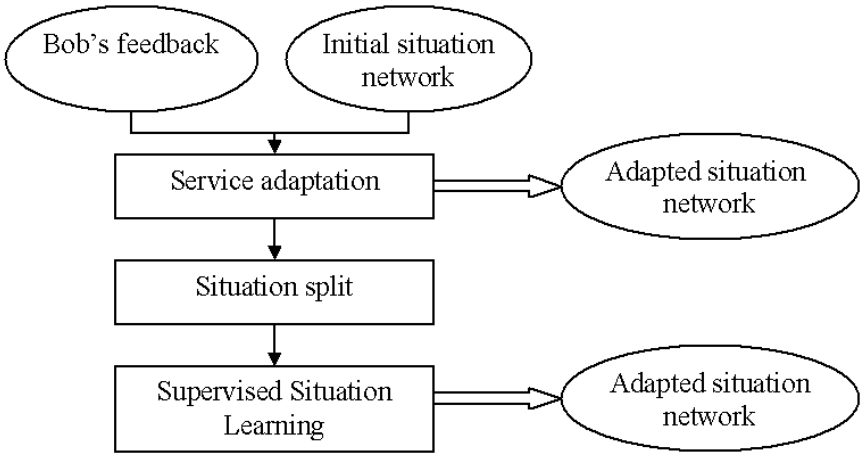


Figure 9.5: Overview of the process of integrating user feedback into the initial situation model

9.2 Implementation and Experimental Evaluation

In this section, we describe our current implementation as well as evaluation and obtained results. The implementation is based on a 3D tracking system that creates and tracks targets in our smart home environment. The extracted target are used to detect individual roles per entity (subsection 9.2.2). Using the role values of several entities, observations are generated (subsection 9.2.3) that are the input for unsupervised situation extraction. The results of the extraction are used for supervised situation learning. The learned situation model is then the basis for the integration of user preferences, i.e. associating and changing services. We did 2 different evaluations (Figure 9.6).

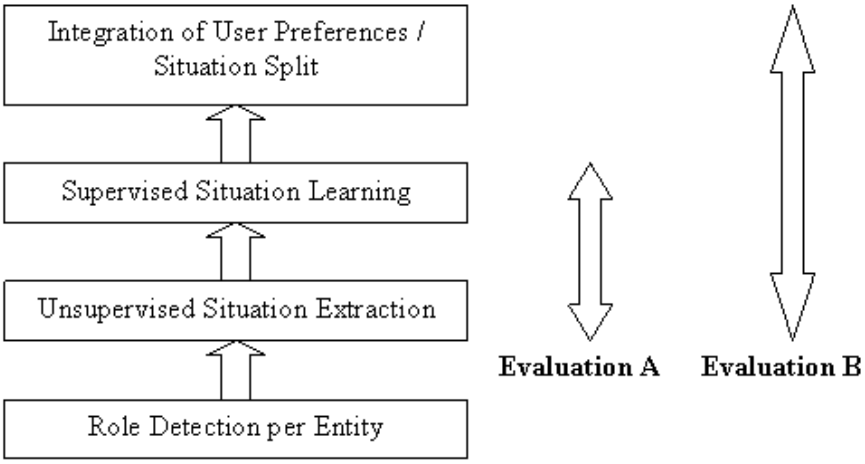


Figure 9.6: Different parts of the implementation and their evaluation: role detection per entity (chapter 5), unsupervised situation extraction (chapter 6), supervised situation learning (chapter 7) and integration of user preferences (chapter 8)

The aim of Evaluation A was to show the benefit of automatic situation extraction and multi-person observation fusion for situation recognition. Therefore, we recorded several small scenarios showing different situations like “presentation” or “siesta”. The recordings have been segmented using the method of chapter 6 and multi-person observations of situations have been fused. We evaluate the recognition of the situations in the scenarios with and without automatic presegmention/multi-person observations as input for supervised situation learning. The aim of Evaluation B was to show and validate the combination of the three methods: unsupervised situation extraction (chapter 6), supervised situation learning (chapter 7) and integration of user preferences (chapter 8). Therefore, we recorded 3 (longer) scenarios showing several situations like “aperitif”, “playing game” or “presentation”. The recordings have first been automatically segmented. Then, the extracted segments have been labeled and the situations have

been learned. Finally, the learned situation model has been evolved with user feedback. We evaluate the recognition of the labeled as well as the added situation (via situation split).

9.2.1 Smart Home Environment: 3D tracker and head set microphones

The experiments described in the following sections are again performed in our laboratory mockup of a living room environment in a smart home (Figure 5.3, subsection 5.2.1). We use the wide-angle camera plus two other normal cameras mounted in the corners of the smart room as well as the microphone arrays and head sets.

A 3D video tracking system [15] detects and tracks entities (people) in the scene in real-time using multiple cameras (Figure 9.7). The tracker itself is an instance of basic Bayesian reasoning [13]. The 3D position of each target is calculated by combining tracking results from several 2D trackers [24] running on the video images of each camera. Each couple camera-detector is running on a dedicated processor. All interprocess communication is managed with an object oriented middleware for service connection [46].

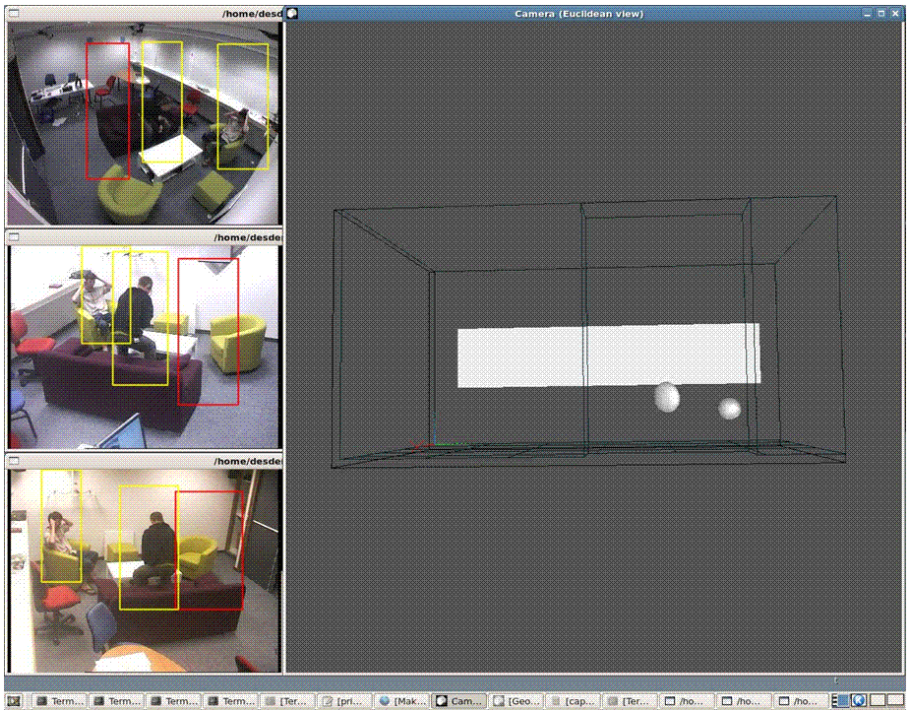


Figure 9.7: 3D video tracking system fusing information of 3 2D trackers to a 3D representation

The output of the 3D tracker are the position (x, y, z) of each detected target as well as the corresponding covariance matrix (3x3 matrix describing the form of the bounding ellipsoid of

the target). Additionally, a velocity vector $\vec{v}$ can be calculated for each target.

The 3D video tracking system provides high tracking stability. The generated 3D target positions correspond to real positions in the environment that can be compared to the position of objects. The extracted target properties (covariance matrix, velocity) provided by the 3D tracker are independent of the camera positions (after calibration). Further, tracking is robust against occlusions and against split and merge of targets.

The microphone array mounted against the wall of the smart environment is used for noise detection. Based on the energy of the audio streams, we determine whether there is noise in the environment or not (e.g. movement of objects on the table).

The people taking part in our experiments wear head set microphones. A real-time speech activity detector [20, 91] analyses the audio stream of each head set microphone and determines whether the corresponding person speaks or not.

The association of the audio streams (microphone number) to the corresponding entity (target) generated by the 3D tracker is done at the beginning of each recording by a supervisor.

Ambient sound, speech detection and 3D tracking are synchronized. As the audio events have a much higher frame rate (62.5 Hz) than video (up to 25 Hz), we add sound events (no sound, speech, noise) to each video frame (of each entity).

9.2.2 Role Detection per Entity

Role detection is conducted per entity and for each observation frame. The input are the extracted properties of each target (position (x, y, z), covariance matrix (3x3 matrix) and speed $|\ \vec{v}\ |$) provided by the 3D tracking system. The output is one of the role labels (Figure 9.11, codes 1-13).

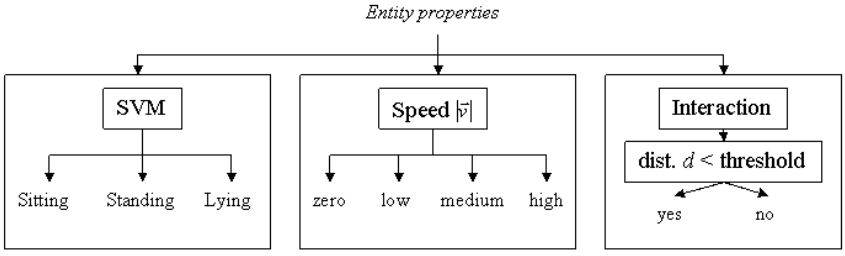


Figure 9.8: Role detection process: SVMs (left), Target Speed (middle), Distance to Interaction Object (right)

The role detection process consists of 3 parts (Figure 9.8). The first part is similar to the SVM method described in chapter 5. This first approach used SVMs as a black box learning method, without considering specific target properties. From first results obtained in our smart home environment, we concluded that, in order to optimize role recognition, we need to reduce the number of classes as well as the target properties used for classification (see also section 5.3). Additional classes are determined by using specific target properties (speed, interaction distance) and expert knowledge (parts 2 and 3 of our approach).

Figure 9.9: Basic individual roles “standing”, “lying” and “sitting” detected by the SVMs

The first part of the process (Figure 9.8 left) takes the covariance matrix values of each target as input. Trained SVMs detect, based on these covariance values, the basic individual roles “sitting”, “standing” and “lying“ (Figure 9.9). As we limit our implementation to a fixed number of role values, we only use SVMs and no hybrid classifier (chapter 5).

The second part of the process (Figure 9.8 middle) uses the speed value $\mid \vec{v} \mid$ of each target. Based on empirical values in our smart environment, we can then determine whether the speed of the target is $zero$, low, $medium$ or $high$.

The third part of the process (Figure 9.8 right) uses the position (x, y, z) of each target to calculate the distance to an interaction object. In our smart environment, we are interested in the interaction with a table at a known position (white table in Figure 5.4). So we calculate the distance d between the target and the table in the environment. If this distance is approaching zero (or below zero), the target is interacting with the table.

The results of the different parts of the detection process are combined to roles following the schema in Figure 9.10.

9.2.3 Multimodal Observation Generation

Based on role detection results as well as the ambient sound and speech detection, we derive multimodal observation codes for each entity created and tracked by the 3D tracking system.

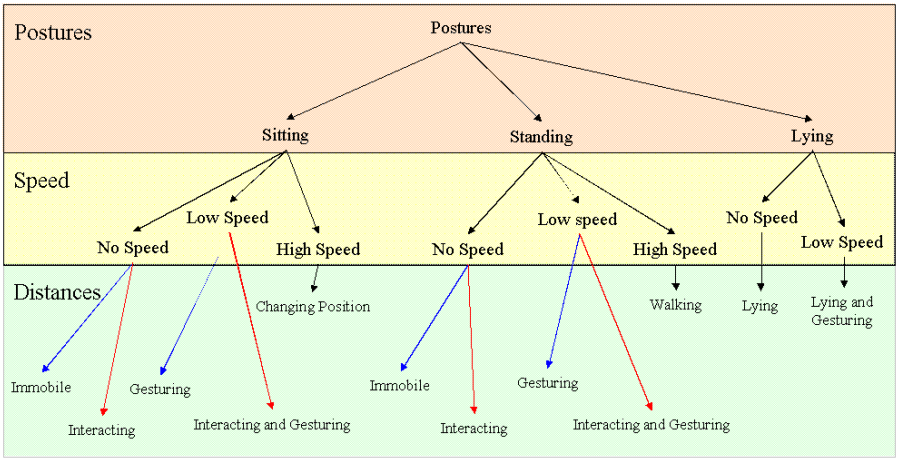


Figure 9.10: Schema describing the combination of basic individual role, speed and distance values to roles (blue arrows refer to ”no interaction distance with table“, red arrows refer to ”interaction distance with table“)

```
0  : entity does not exist
1  : standing immobile
2  : standing and interacting with table
3  : standing and gesturing
4  : standing and interacting with table (in movement)
5  : walking
6  : sitting
7  : sitting and interacting with table
8  : sitting and gesturing
9  : sitting and interacting with table (in movement)
10 : changing position while sitting
11 : lying
12 : lying and gesturing
13 : detection error

14-26 : entity is speaking
27-39 : there is noise in the environment
40-52 : entity is speaking and there is noise
```

Figure 9.11: Multimodal entity observation codes

12 individual role values (Figure 9.10) are derived for each entity by the role detection process. Further, the ambient sound detector indicates whether there is noise in the environment or not. The speech activity detector determines whether the concerned entity is speaking or not. This multimodal information is fused to 53 observation codes for each entity (Figure 9.11). Codes 1-13 (13 codes) are based on the role detection process. These 13 codes are combined with ambient sound detection (codes 27-39 and 40-52) and speech detection per entity (codes 14-26 and 40-52). As ambient sound and speech detection return binary values, $2^2 * 13 = 52$ different code values are necessary to represent role, ambient sound and speech detection. If we add an observation code value for a non-existing entity (code 0), we get 53 different observation code values.

Fusion algorithm

Input: (a, b), $0 \leq a, b \leq max_{code}$

Step 1: if ($a > b$) {exchange(a, b)},

Step 2: $code = \sum_{i=0}^{a-1}\{(max_{code}+1)-i\}+(b-a)$.

Figure 9.12: Fusion algorithm combining the role detection values (a, b) of two entities. For $max_{code} = 52$, the resulting codes are between 0 and 1430

As we can have several persons involved in a situation, we need to fuse the multimodal codes of several entities. We could simply combine the multimodal entity detection codes (for two entities: $53 * 53 = 2809$ codes). However, the result is a high number of possible observation values. Further, as we are interested in situation recognition, many of these values are redundant. For example, person A is lying and person B is sitting is a different observation code as person A is sitting and person B is lying, even though, from the perspective of activity recognition, the situation is identical. Therefore, we employ the small fusion algorithm shown in Figure 9.12. The idea is to attribute a code to the combination of two multimodal entity observation codes (without considering their order). The resulting observation code fuses the observation codes of two (or more) entities. In order to fuse the observation codes of more than two entities, the fusion algorithm can be applied several times, fusing successively all entities.

We would like to mention that the fact that multimodal observation code 0 (Figure 9.11) corresponds to the non-existence of the entity implies that a generated multi-person code (Figure 9.12) contains all generated lower codes. That is, if we generate, for example, a two person code for only one entity, the resulting code and the observation code of the entity are identical. This enables, for example, the comparison of one-person and multi-person observation codes.

9.2.4 Evaluation A

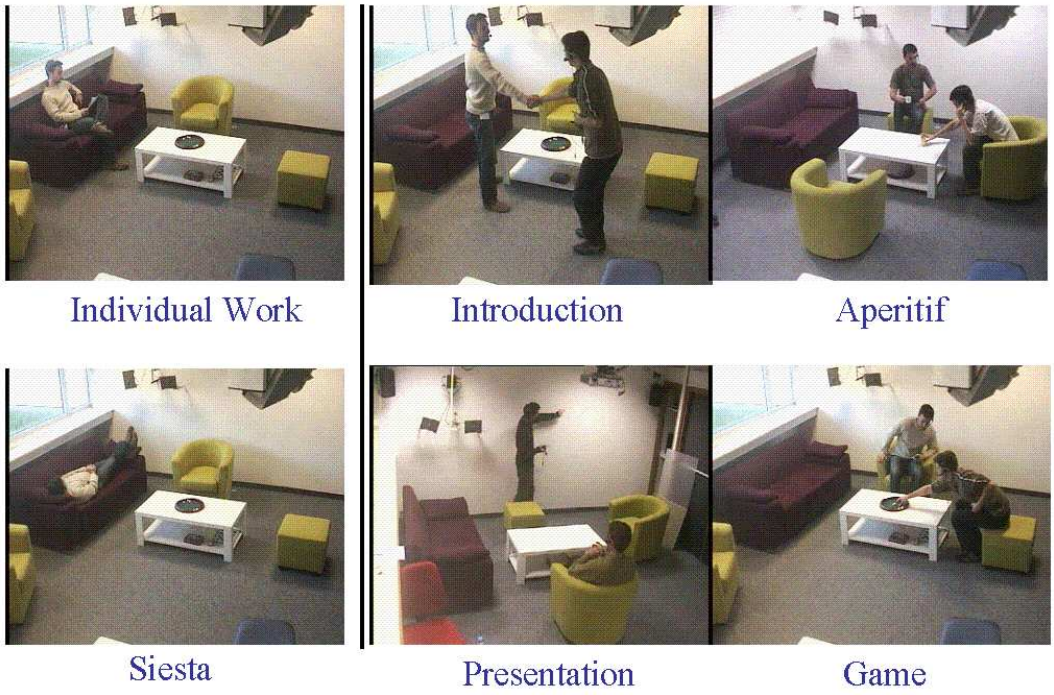


Figure 9.13: One person situations ”individual work“ and ”siesta“ (left side) and multi-person situations ”introduction“, ”aperitif“, ”presentation“ and ”game“

In this subsection, we aim at showing the benefit of automatic presegmentation and multi-person observation fusion for situation learning and recognition. Therefore, we made three different recordings of each of the following situations: ”siesta“, ”an individual working“, ”aperitif“, ”introduction/address of welcome“, ”presentation“, and ”playing a game“. ”Introduction/address of welcome“, ”aperitif“, ”presentation“ and ”playing a game“ involved two persons, while ”siesta“ and ”individual work“ concerned only one person. The role detection values have been generated as described in subsection 9.2.2. The sequences designated for learning are presegmented (see method chapter 6), i.e. only the segment containing the pure situation is used for learning. This means that, for recordings containing only one situation, disturbances at the beginning and at the end of the recording are automatically removed (see Figure 9.14 for an example). The supervised learning scheme (chapter 7) is then used for learning the situation representations from the sequences. We adopt hidden Markov models as unique learner class, iterating over left-right hidden Markov models of state numbers between 8 and 16 (=parameters of the class).

First, we did an evaluation on the situation detection for one person only (role detection value between 0 and 52). Situation recordings involving 2 people gave thus two one-person sequences.

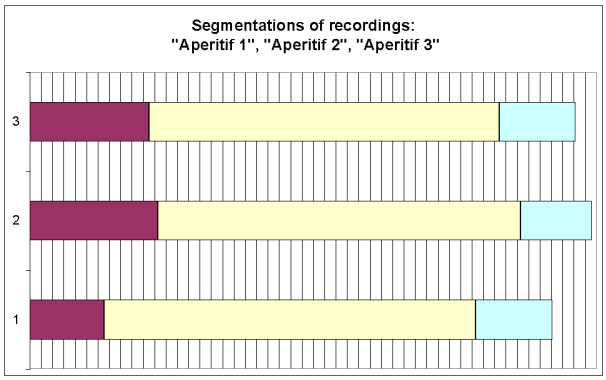


Figure 9.14: Extracted segments for situation recordings ”Aperitif 1”, ”Aperitif 2”, ”Aperitif 3”. Segments at the beginning and at the end of the recordings will be removed automatically

We did a 3-fold cross-validation, taking two third of the sequences as input for supervised learning and the remaining third of the sequences as basis for recognition. Table 9.1 and Table 9.3 show the results. The presegmentation improves the recognition results for the one-person recording sequences. In particular, ”aperitif“ and ”game“ can correctly be distinguished, while some wrong detections between ”introduction“ and ”presenter“ persist.

Additionally, we did an evaluation on the situation detection for two-person situations. Therefore, we use the fusion algorithm proposed in subsection 9.2.3, generating the multi-person observation codes. We did again a 3-fold cross-validation on the situation recognition after supervised situation learning of the given observation sequences. Table 9.2 and Table 9.4 show the results. The presegmentation also improves the recognition results for the two-person recordings. As for the one-person situation detection, situations ”aperitif“ and ”game” can correctly be distinguished with presegmentation. The two-person observation fusion further eliminates wrong detections between “aperitif” and “game”, resulting in a correct situation recognition rate of 100 % (Table 9.4). The obtained results indicate that presegmentation and the fusion of individual role detection codes is beneficial when learning and recognizing situations involving several persons.

	Siesta	Individual Work.	Aperitif	Introduc.	Presenter	Game	Audience
Siesta	1	0	0	0	0	0	0
Individual Work.	0	1	0	0	0	0	0
Aperitif	0	0	0.8333	0	0	0.1667	0
Introduc.	0	0	0	0.8333	0.1667	0	0
Presenter	0	0	0	0	1	0	0
Game	0	0	0	0	0	1	0
Audience	0	0	0	0	0	0	1

Class	TP rate	FP rate	Precision	Recall	F-measure
Siesta	1	0	1	1	1
Individual Work.	1	0	1	1	1
Aperitif	0.8333	0	1	0.8333	0.8889
Introduc.	0.8333	0	1	0.8333	0.8889
Presenter	1	0.037	0.8333	1	0.8889
Game	1	0.0417	0.8889	1	0.9333
Audience	1	0	1	1	1
Total	0.9524	0.0112	0.9603	0.9524	0.9429

Table 9.1: Confusion matrix and information retrieval statistics for one-person situation detection without presegmentation. The total recognition rate is 93.33 %

	Aperitif	Introduc.	Presentation	Game
Aperitif	0.6667	0	0.3333	0
Introduc.	0	1	0	0
Presentation	0	0	1	0
Game	0	0	0	1

Class	TP rate	FP rate	Precision	Recall	F-measure
Aperitif	0.6667	0	0.6667	0.6667	0.6667
Introduc.	1	0	1	1	1
Presentation	1	0.1111	0.8333	1	0.8889
Game	1	0	1	1	1
Total	0.9167	0.0278	0.875	0.9167	0.8889

Table 9.2: Confusion matrix and information retrieval statistics for two-person situation detection without presegmentation. The total recognition rate is 91.67 %

	Siesta	Individual Work	Aperitif	Introduc.	Presenter	Game	Audience
Siesta	1	0	0	0	0	0	0
Individual Work.	0	1	0	0	0	0	0
Aperitif	0	0	1	0	0	0	0
Introduc.	0	0	0	0.8333	0.1667	0	0
Presenter	0	0	0	0	1	0	0
Game	0	0	0	0	0	1	0
Audience	0	0	0	0	0	0	1

Class	TP rate	FP rate	Precision	Recall	F-measure
Siesta	1	0	1	1	1
Individual Work.	1	0	1	1	1
Aperitif	1	0	1	1	1
Introduc.	0.8333	0	1	0.8333	0.8889
Presenter	1	0.037	0.8333	1	0.8889
Game	1	0	1	1	1
Audience	1	0	1	1	1
Total	0.9762	0.0053	0.9762	0.9762	0.9683

Table 9.3: Confusion matrix and information retrieval statistics for one-person situation detection with presegmentation. The total recognition rate is 96.67 %

	Aperitif	Introduc.	Presentation	Game
Aperitif	1	0	0	0
Introduc.	0	1	0	0
Presentation	0	0	1	0
Game	0	0	0	1

Class	TP rate	FP rate	Precision	Recall	F-measure
Aperitif	1	0	1	1	1
Introduc.	1	0	1	1	1
Presentation	1	0	1	1	1
Game	1	0	1	1	1
Total	1	0	1	1	1

Table 9.4: Confusion matrix and information retrieval statistics for two-person situation detection with presegmentation. The total recognition rate is 100 %

9.2.5 Evaluation B

In this subsection, we intend to show and validate the combination of the three methods: unsupervised situation extraction (chapter 6), supervised situation learning (chapter 7) and integration of user preferences (chapter 8). Therefore, we evaluated the integral approach on 3 scenarios recorded in our smart home environment. The scenarios involved up to 2 persons doing different activities (situations: “introduction/address of welcome”, “presentation”, “aperitif”, “playing a game”, “siesta”{1 person}) in the environment. The role detection values have been generated as described in subsection 9.2.2 using 3D tracker as well as noise and speech detection (head set microphones). The role detection values have then been fused to observation codes as described in subsection 9.2.3.

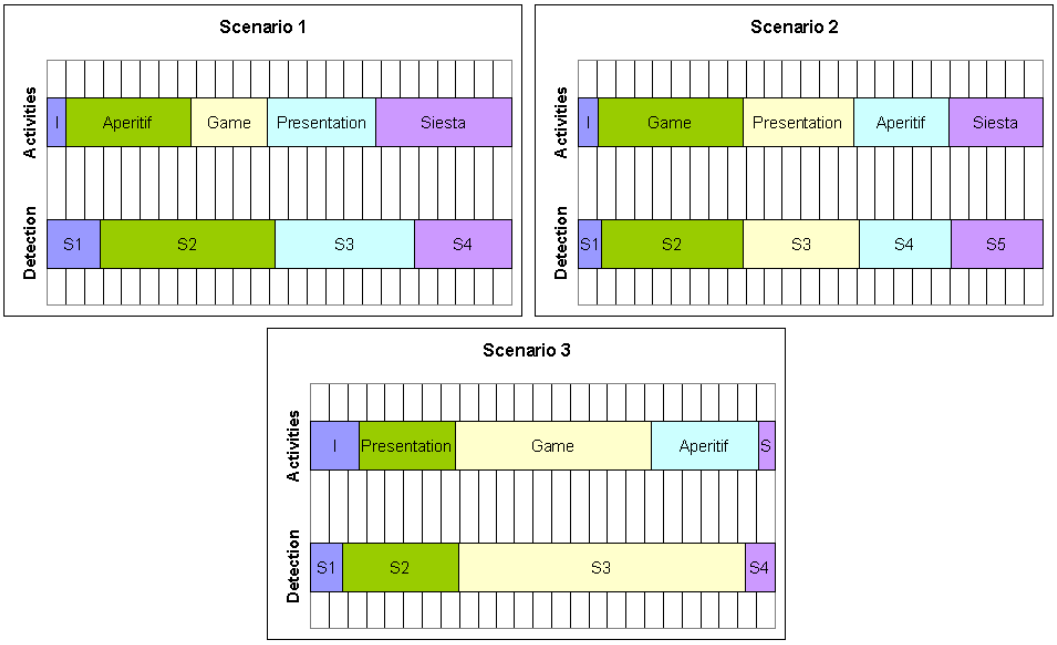


Figure 9.15: Extracted situation segments and the corresponding *ground truth* for scenario 1 (Q = 0.68), scenario 2 (Q = 0.95), scenario 3 (Q = 0.74)

The first step of our proposed approach is to create the initial situation model. We extract the situations from the sensor perceptions, i.e. the observations generated for the targets in the scene using our automatic segmentor (chapter 6). The automatically extracted segments and the *ground truth* for the scenarios are depicted in Figure 9.15. The overall segmentation exactitude Q (subsection 6.2.1) is best for scenario 2. This can be explained by the fact that the algorithm has difficulties to distinguish ground truth segments “game” and “aperitif”. In scenario 1 and scenario 3, “game” and “aperitif” are detected as one segment. Because in scenario 2, “playing game” and “aperitif” are separated by “presentation”, these segments can be correctly detected.

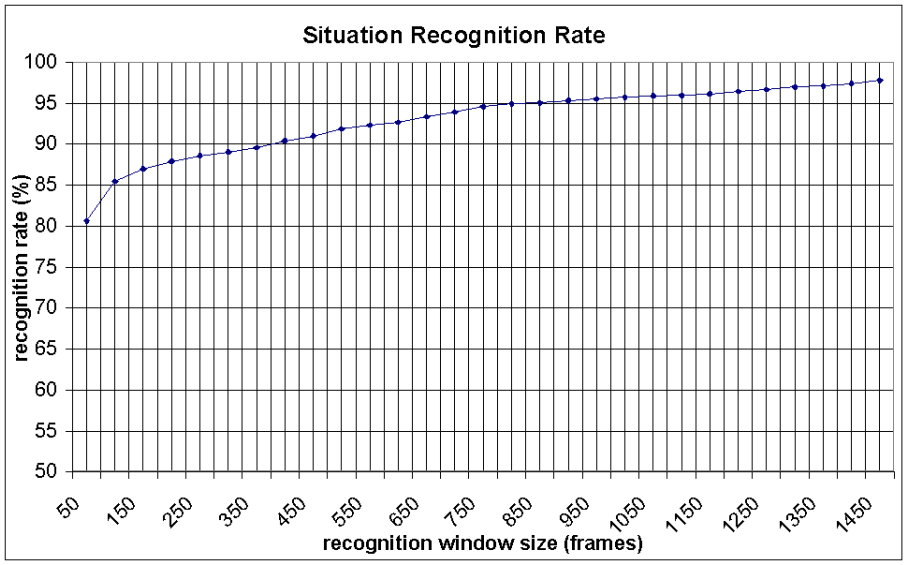


Figure 9.16: Recognition rate of situations “introduction”, “presentation”, “group activity” (=“aperitif” or “game”) and “siesta” for different recognition window sizes

The supervised learning scheme (chapter 7) is applied on the detected segments. As expert knowledge, we inject the situation labels: “introduction”, “presentation”, “group activity” (=“aperitif” or “game”), “siesta”. We will adopt hidden Markov models as unique learner class, iterating over left-right hidden Markov models of state numbers between 8 and 16 (=parameters of the class). To evaluate, we did 3-fold cross-validation, taking the detected segments + expert labels of 2 scenarios as input for learning and the third scenario as basis for recognition. As our system should be as responsive as possible, we evaluated different window sizes used for recognition. The obtained situation recognition rates are depicted in Figure 9.16. If we limit the observation time provided for recognition to 10 seconds (i.e. 250 frame with a frame rate of 25 frames/sec), we get a recognition rate of 88.58 % (Table 9.5). The recognition rate of “siesta” is poor due to the fact that in two of the three scenario recordings wrong targets have been created and detected when a person lay down on the couch, resulting in a disturbance of the existing target properties.

We have now learned an initial situation model with the situations “introduction”, “group activity”, “presentation” and “siesta”. In order to integrate user preferences into this model, a user can give feedback to our system. The feedback is recorded and associated to the particular frame when it has been given. The initially learned model is then adapted according to this feedback (chapter 8). For our scenarios, we want to integrate the following services:

- S1: Introduction $\Rightarrow$ normal light and no music

	Introduction	Group Activity	Presentation	Siesta
Introduction	0.9766	0	0	0.0234
Group Activity	0	1	0	0
Presentation	0.032	0.027	0.8372	0.1037
Siesta	0.224	0.0093	0.3204	0.4463

Class	TP rate	FP rate	Precision	Recall	F-measure
Introduction	0.9766	0.0552	0.6362	0.9766	0.7073
Group Activity	1	0.0154	0.9871	1	0.9934
Presentation	0.8372	0.0159	0.958	0.8372	0.8848
Siesta	0.4463	0.0392	0.8064	0.4463	0.4426
Total	0.815	0.0314	0.8469	0.815	0.757

Table 9.5: Confusion matrix and information retrieval statistics for each situation (observation window size=250). The overall situation recognition rate is 88.58 %

- S2: Aperitif $\Rightarrow$ dimmed light and jazz music
- S3: Game $\Rightarrow$ normal light and pop music
- S4: Presentation $\Rightarrow$ dimmed light and no music
- S5: Siesta $\Rightarrow$ dimmed light and yoga music

The user gives one feedback indicating the corresponding service during each situation. As the initial situation model does not contain any situation-service associations, S1, S4 and S5 can then be simply associated to the corresponding situations. For S2 and S3, there is only one situation “group activity” which is too general in order to associate both distinct services. This situation needs thus to be split into sub-situations. The learned situation representation for “group activity” (here: a HMM) is erased and two distinct situation representations (here: HMMs) for “aperitif” and “game” are learned. The observations necessary to learn these situations are taken around the time points when the user gave the corresponding feedback. The size of the observation window used for learning the new sub-situations can be varied. The situation recognition rates for different learning window sizes are depicted in Figure 9.17. We evaluated three different window sizes used for recognition (250, 500 and 1000 observations), corresponding to the three indicated curves. The larger the recognition window size, the better the total situation recognition rate. Thus recognition rates for window size 1000 are higher than for window sizes 250 and 500. However, the curves indicate that a larger learning window size does not always result in a better recognition rate. The total situation recognition rate can even drop with a larger learning window size. This is due to the fact that the best recognition results are obtained when the learning window contains a maximum of observation data being

characteristic for the concerned situation and a minimum of “foreign“ observations, i.e. wrong detections or observations corresponding to other situations. The resulting situation recognition curve tends upwards, but it contains local peaks corresponding to a learning window size with a good tradeoff between characteristic and foreign observations. For our scenario recordings, such a local peak is at a learning window size of 400, i.e. 400 observations around the feedback time points to learn “aperitif” and “game”. The obtained results of the 3-fold cross validation for recognition window sizes 250, 500 and 1000 are detailed in Tables 9.6, 9.7 and 9.8.

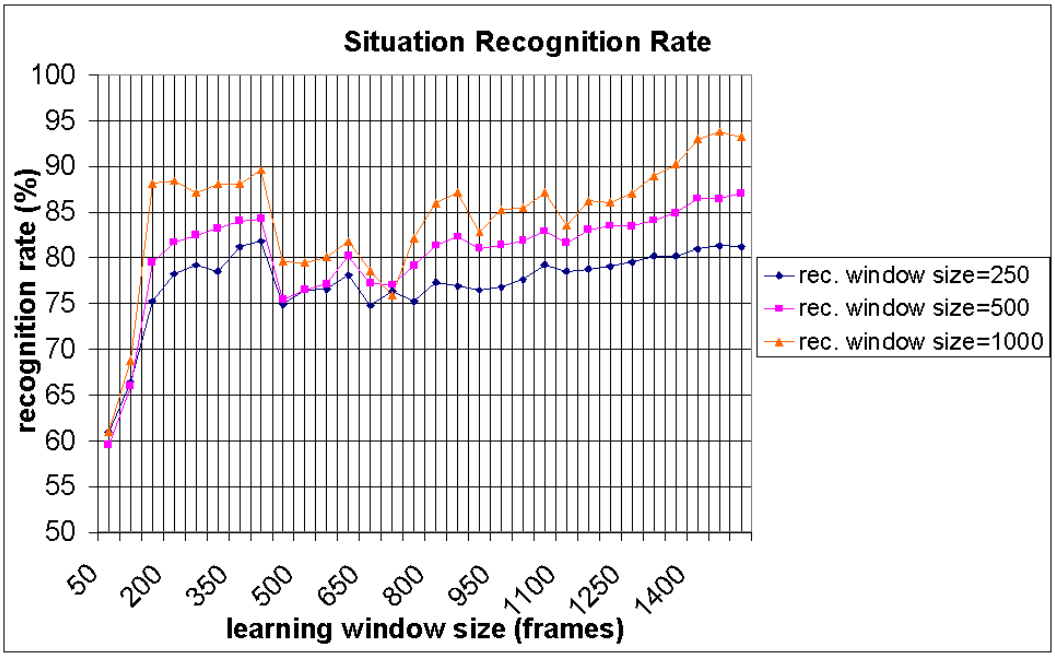


Figure 9.17: Recognition rate of situations “introduction”, “presentation”, “aperitif”, “game” (after split) and “siesta” for different learning window sizes. The three curves are for 250, 500 and 1000 observations (recognition window size)

	Introduction	Aperitif	Game	Presentation	Siesta
Introduction	0.9743	0	0	0	0.0257
Aperitif	0	0.697	0.2985	0	0.0045
Game	0	0.0088	0.9912	0	0
Presentation	0.0361	0	0.0262	0.8378	0.0999
Siesta	0.224	0	0	0.3297	0.4463

Class	TP rate	FP rate	Precision	Recall	F-measure
Introduction	0.9743	0.0586	0.6275	0.9743	0.7009
Aperitif	0.697	0.0042	0.9774	0.697	0.7959
Game	0.9912	0.0875	0.8062	0.9912	0.8823
Presentation	0.8378	0.0172	0.957	0.8378	0.8845
Siesta	0.4463	0.0412	0.8035	0.4463	0.4433
Total	0.7893	0.0417	0.8343	0.7893	0.7414

Table 9.6: Confusion matrix and information retrieval statistics for each situation (observation window size=250) after the split of “group activity”. The window size for learning the new sub-situations is 400. The overall situation recognition rate is 81.86 %

	Introduction	Aperitif	Game	Presentation	Siesta
Introduction	1	0	0	0	0
Aperitif	0	0.6476	0.3524	0	0
Game	0	0	1	0	0
Presentation	0.0026	0	0	0.9002	0.0972
Siesta	0.2175	0	0	0.3333	0.4491

Class	TP rate	FP rate	Precision	Recall	F-measure
Introduction	1	0.044	0.6535	1	0.6666
Aperitif	0.6476	0	1	0.6476	0.7604
Game	1	0.092	0.7996	1	0.8806
Presentation	0.9002	0.0002	0.9995	0.9002	0.9452
Siesta	0.4491	0.0414	0.4917	0.4491	0.4421
Total	0.7994	0.0355	0.7889	0.7994	0.739

Table 9.7: Confusion matrix and information retrieval statistics for each situation (observation window size=500) after the split of “group activity”. The window size for learning the new sub-situations is 400. The overall situation recognition rate is 84.27 %

	Introduction	Aperitif	Game	Presentation	Siesta
Introduction	1	0	0	0	0
Aperitif	0	0.8246	0.1754	0	0
Game	0	0	1	0	0
Presentation	0	0	0	1	0
Siesta	0.1889	0	0	0.3343	0.4768

Class	TP rate	FP rate	Precision	Recall	F-measure
Introduction	1	0.034	0.6672	1	0.6678
Aperitif	0.8246	0	1	0.8246	0.881
Game	1	0.0487	0.9336	1	0.9631
Presentation	1	0.0004	0.999	1	0.9995
Siesta	0.4768	0	0.6667	0.4768	0.534
Total	0.8603	0.0166	0.8533	0.8603	0.8091

Table 9.8: Confusion matrix and information retrieval statistics for each situation (observation window size=1000) after the split of “group activity”. The window size for learning the new sub-situations is 400. The overall situation recognition rate is 89.62 %

9.3 Conclusions

In this chapter, we combined and integrated different methods proposed in the precedent chapters (chapter 5,6,7,8) for learning and evolving situation models. To evaluate, several scenarios have been recorded in our smart home environment. A 3D tracking system created and tracked entities in the scene, while a microphone array detected noise and head sets microphones were used to detect speech of individuals. Different individual role values have been detected and fused to multi-person observation codes. A first evaluation (evaluation A) has been conducted on the recognition of learned situations with and without automatic presegmentation (chapter 6). The results showed that both the automatic presegmentation and the fusion of role values to multi-person observation codes are beneficial for situation recognition. A second evaluation was conducted for validating the integration of the different methods (chapter 6,7 and 8). The results indicate that the integration of the different methods into an intelligent home environment system is possible.

Although the obtained results are encouraging, the realization of Bob's dream (section 9.1) of an intelligent home anticipating his needs and desires is still far away. First products that Bob could buy in his local computer store and install himself are not mature enough. First, the sensors necessary for a reliable sensing of Bob's activities are still too invasive. Multiple cameras, microphones or other sensors must be installed and calibrated in Bob's home. These are still not auto-installing and not easy to use. Second, even though our results are encouraging, the error rates are still too high. Further improvements in detection and learning algorithms are necessary in order to provide a reliable system that could be accepted by Bob in his daily life. One way to alleviate this is to provide explanations to Bob. When errors occur (and corresponding system explanations are good), Bob could understand and correct wrong system perceptions and reasoning himself.

Chapter 10

Conclusion and Perspective

We showed that modeling context is necessary when we want to sense and respond correctly to human activity in intelligent environments. However, it is difficult (or even impossible) to model all possible contextual states reflecting human activity in the scene. Further, not all possible contextual cues can be sensed and integrated into a context model. As consequence, to realize functioning context-aware applications, we need to assume a closed world, modeling hence only human activity that is essential for correct system behavior and services. The situation model is proposed as intuitive declarative representation of context, providing both a simple mean for describing human (inter)actions and a powerful tool for implementation. Two possible implementations of the situation model have been presented and illustrated by two functioning applications: an automatic cameraman and an interaction group detector.

In order to cope with changing user behavior and environments, an intelligent environment must be able to evolve and acquire new models for human behavior. However, automatic acquisition and adaptation to changing human behavior in an intelligent environment is a non-trivial problem. On the one hand, there is the problem of usability of adaptive systems and on the other hand, of course, the problem of learning abstract models of human activity. Our objective here was to propose several methods that are part of an intelligible framework for acquiring and evolving context models. The situation model serves as frame and support for these methods. First, we described several methods for learning individual roles from observation data. Therefore, a Bayesian classifier, support vector machines and a hybrid classifier have been presented and evaluated. Further, we proposed a method for the unsupervised segmentation of possible situations from a stream of observations. This method has been successfully experimented with several (multimodal) data sets and applications. Then, we presented a third method constructing situation representations for given situation labels and perception segments. The CAVIAR data sets have been used to evaluate the method. A last method evolves a predefined situation network with feedback on system behavior (i.e. the provided system services). This method has

been tested in the PRIMA SmartOffice environment. Finally, the proposed methods have been integrated into a whole system for acquiring and evolving situation models in an intelligent home environment. Experimental results are encouraging. However, the error rates are still too high for being acceptable to users.

To conclude, we can say that learning abstract models reflecting human behavior is a hard problem. There are already many different models and implementations proposed for representing context. These models are normally adapted to a specific domain and application. Evolving and acquiring such a context model automatically entails an even more rigorous adjustment to a specific domain, application and learning algorithm in order to obtain satisfying results. There is no universal approach. However, we aimed at proposing a framework of different learning methods that have been successfully applied and that are based on our situation model defined in the first part of this thesis.

10.1 Contributions

This thesis aims at providing a novel intuitive framework for acquiring and evolving situation models in intelligent environments. Contributions are made in the areas of context modeling and implementation and learning of context models.

In the area of situation modeling and implementation:

Probabilistic Implementation of Situation Models Different implementations of situation models are presented. A novel probabilistic implementation based on hidden Markov models (HMMs) is proposed. Each state of the HMM represents one situation. A real-time interaction group detector has been realized and evaluated using the proposed implementation.

Contributions in the area of situation model learning for knowledge engineering concern:

Role Learning and Detection Bayesian classifier and support vector machines are tested and evaluated for the task of role learning and detection from video data. A novel hybrid classifier is proposed, combining both methods. The hybrid classifier outperformed support vector machines and Bayesian classifier when learning new unseen roles from our data sets.

Unsupervised Situation Discovery A novel method for the unsupervised extraction of situations from multimodal observations is proposed and evaluated on different multimodal data sets from meetings.

Supervised Situation Learning Scheme A supervised learning scheme is proposed to learn situation representations. The scheme is to be very general, permitting to use different parameterizations and learner classes. A first evaluation on the CAVIAR project data sets produced good results.

Contributions in the area of situation model learning for user preferences integration are:

Situation Split A novel algorithmic method for integrating user preferences into a given situation model is proposed. The user gives feedback on executed system services. The method refines system perceptions by splitting too general situations into sub-situations.

A major contribution of this thesis is the *integration* of the proposed methods into a whole system for acquiring and evolving situation models in an intelligent home environment. The whole system has been *successfully tested and evaluated*.

10.2 Perspectives

This thesis aimed at constructing a first framework for learning human behavior and needs from observation and feedback in order to provide context-aware services. Several problems still need to be addressed:

Error Rates If we aim at providing unobtrusive system behavior, the error rates of recognition and learning algorithms are still too high. Apart from permanently improving these algorithms, we can try to limit the action space of the system. Some actions are less critical and disruptive than others and can be automated with higher error rates in recognition algorithms.

Generation of Explanations We provide a first approach for intuitive modeling and reasoning about context. The layers of the framework are motivated by a human understandable representation of context. Even though the reasoning process can be easily tracked between these layers, no explicit explanations are generated for the user. Such explanations are especially important when errors occur. We need to find an unobtrusive way to provide the user with these explanations within the environment. The generated explanations may also be linked with system control, permitting the user to overrule system decisions.

Controllability and Human-Computer Interaction The user must be kept in control. Even though our framework and the learning algorithms are based on an intuitive context model, the user might want to take direct control of the whole system. In this case, an

important issue is how to visualize contextual information in an intuitive manner and how to enable the user to control system services easily and without high learning effort. Interface design and type of (explicit) human-computer interaction must carefully be chosen according to the expert level of the user and the physical configuration of the environment.

Interruptability and Action Cost Concerning unobtrusive service supply, we need to be aware of the disruptive power of each service. Disruptive power refers to the capacity of interrupting current user task or even current human-human interaction. A service or system action can be disruptive even if the service is pertinent. Therefore, the interruptability of the users must be derived. Based on this information, "cost" and benefit of each service must be estimated and balanced before service execution.

Publications during thesis period

International peer-reviewed journals

1. O. Brdiczka, J. Maisonnasse, P. Reignier, and J. Crowley. Detecting small group activities from multimodal observations. International Journal of Applied Intelligence, 2007, Springer. IN PRESS.

2. P. Reignier, O. Brdiczka, D. Vaufreydaz, J. Crowley, and J. Maisonnasse. Deterministic and probabilistic implementation of context in smart environments. Expert Systems: The Journal of Knowledge Engineering, 2007, Blackwell Publishing. IN PRESS.

3. O. Brdiczka, P. Yuen, J. Crowley, and P. Reignier. Asimo: Automatic acquisition of situation models. eMinds: International Journal on Human-Computer Interaction, issue 2, pages 3-15, 2007.

National peer-reviewed journals

1. O. Brdiczka, P. Reignier, and J. Crowley. Modéliser et faire évoluer le contexte dans des environnements intelligents. Ingénierie des Systèmes d'Information (ISI), 11(5):9-34, December 2006, Lavoisier. URL `http://isi.revuesonline.com/article.jsp?articleId=8949`

International peer-reviewed conferences

1. O. Brdiczka, P. Reignier, and J. Crowley. Detecting individual activities from video in a smart home. In Proceedings of 11th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES), 2007. IN PRESS.

2. O. Brdiczka, J. Crowley, and P. Reignier. Learning situation models for providing context-aware services. In HCI International, 2007. IN PRESS.

3. J. Crowley, O. Brdiczka, and P. Reignier. Learning situation models for understanding activity. In Proceedings of 5th International Conference on Development and Learning, Bloomington, USA, May 2006.

4. O. Brdiczka, J. Maisonnasse, P. Reignier, and J. Crowley. Learning individual roles from video in a smart home. In Proceedings of 2nd IET International Conference on Intelligent Environments, Athens, Greece, volume 1, pages 61-69, July 2006.

5. O. Brdiczka, J. Maisonnasse, P. Reignier, and J. Crowley. Extracting activities from multimodal observation. In Proceedings of 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES), Bournemouth, UK, volume 2, pages 162-170, October 2006. URL http://dx.doi.org/10.1007/11893004_21

6. O. Brdiczka, P. Yuen, S. Zaidenberg, P. Reignier, and J. Crowley. Automatic acquisition of context models and its application to video surveillance. In Proceedings of 18th International Conference on Pattern Recognition (ICPR), Hong Kong, volume 1, pages 1175-1178, August 2006. URL http://dx.doi.org/10.1109/ICPR.2006.292

7. J. Maisonnasse, N. Gourier, O. Brdiczka, and P. Reignier. Attentional model for perceiving social context in intelligent environments. In 3rd IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI) 2006, Athens, Greece, pages 171-178, June 2006. URL http://dx.doi.org/10.1007/0-387-34224-9_20

8. S. Zaidenberg, O. Brdiczka, P. Reignier, and J. Crowley. Learning context models for the recognition of scenarios. In 3rd IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI) 2006, Athens, Greece, pages 86-97, June 2006. URL http://dx.doi.org/10.1007/0-387-34224-9_11

9. O. Brdiczka, D. Vaufreydaz, J. Maisonnasse, and P. Reignier. Unsupervised segmentation of meeting configurations and activities using speech activity detection. In 3rd IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI) 2006, Athens, Greece, pages 195-203, June 2006. URL http://dx.doi.org/10.1007/0-387-34224-9_23

10. O. Brdiczka, J. Maisonnasse, and P. Reignier. Automatic detection of interaction groups. In Proceedings of International Conference on Multimodal Interfaces (ICMI), pages 32-36, October 2005. URL http://doi.acm.org/10.1145/1088463.1088473

11. O. Brdiczka, P. Reignier, and J. L. Crowley. Supervised learning of an abstract context model for an intelligent environment. In Proceedings of Smart Object and Ambient Intelligence Conference (sOc-EUSAI), pages 259-264, October 2005. URL http://doi.acm.org/10.1145/1107548.1107612

12. M. Muehlenbrock, O. Brdiczka, D. Snowdon, and J.-L. Meunier. Learning to detect user activity and availability from a variety of sensor data. In Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom'04), pages 13-23, March 2004. URL http://dx.doi.org/10.1109/PERCOM.2004.1276841

National peer-reviewed conferences

1. J. Maisonnasse, N. Gourier, O. Brdiczka, P. Reignier, and J. Crowley. Gestion d'applications confidentielles sur la base d'un modèle attentionnel. In Troisièmes Journées Francophones: Mobilité et Ubiquité (UbiMob), pages 143-146, September 2006.

2. J. Maisonnasse and O. Brdiczka. Détection automatique des groupes d'interactions. In Deuxièmes Journées Francophones: Mobilité et Ubiquité (UbiMob), pages 201-204, May 2005.

International peer-reviewed workshops

1. J. Maisonnasse, N. Gourier, O. Brdiczka, P. Reignier, and J. Crowley. Detecting privacy in attention aware systems. In Proceedings of 2nd IET International Conference on Intelligent Environments, Athens, Greece, volume 2, pages 231-239, July 2006.

2. O. Brdiczka, P. Reignier, J. Crowley, D. Vaufreydaz, and J. Maisonnasse. Deterministic and probabilistic implementation of context. In Proceedings of IEEE International Conference on Pervasive Computing and Communications Workshops, pages 46-50, March 2006. URL http://dx.doi.org/10.1109/PERCOMW.2006.40

3. O. Brdiczka, P. Reignier, and J. Maisonnasse. Unsupervised segmentation of small group meetings using speech activity detection. In ICMI Workshop Proceedings, International Workshop on Multimodal Multiparty Meeting Processing (MMMP), pages 47-52, October 2005.

4. O. Brdiczka, P. Reignier, and J. L. Crowley. Automatic development of an abstract context model for an intelligent environment. In Proceedings of IEEE International Conference on Pervasive Computing and Communications Workshops, pages 35-39, March 2005. URL http://dx.doi.org/10.1109/PERCOMW.2005.17

Bibliography

[1] G. D. Abowd and E. D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58, 2000. ISSN 1073-0516. URL http://doi.acm.org/10.1145/344949.344988.

[2] G. D. Abowd, C. G. Atkeson, A. Feinstein, C. Hmelo, R. Kooper, S. Long, N. Sawhney, and M. Tani. Teaching and learning as multimedia authoring: the classroom 2000 project. In *MULTIMEDIA '96: Proceedings of the fourth ACM international conference on Multimedia*, pages 187–198, New York, NY, USA, 1996. ACM Press. ISBN 0-89791-871-1. URL http://doi.acm.org/10.1145/244130.244191.

[3] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer London, 1999. ISBN 3-540-66550-1. URL http://link.springer.de/link/service/series/0558/bibs/1707/17070304.htm.

[4] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11): 832–843, 1983. ISSN 0001-0782. URL http://doi.acm.org/10.1145/182.358434.

[5] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(1):123–154, 1984. URL http://dx.doi.org/10.1016/0004-3702(84)90008-0.

[6] J.-M. Andreoli, S. Castellani, A. Grasso, J.-L. Meunier, M. Muehlenbrock, J. O'Neill, F. Ragnet, F. Roulland, and D. Snowdon. Augmenting offices with ubiquitous sensing. In *Proceedings of Smart Objects Conference (SOC)*, Grenoble, France, 2003.

[7] S. Antifakos. *Improving Interaction with Context-Aware Systems*. PhD thesis, ETH Zurich, 2005.

[8] P. M. Aoki, M. Romaine, M. H. Szymanski, J. D. Thornton, D. Wilson, and A. Woodruff. The mad hatter's cocktail party: a social mobile audio space supporting multiple simul-

taneous conversations. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 425–432, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-630-7. URL http://doi.acm.org/10.1145/642611.642686.

[9] J. E. Bardram, T. R. Hansen, M. Mogensen, and M. Soegaard. Experiences from real-world deployment of context-aware technologies in a hospital environment. In *Proceedings of UbiComp*, pages 369–386. Springer Berlin / Heidelberg, 2006. URL http://dx.doi.org/10.1007/11853565_22.

[10] L. Barkhuus and A. Dey. Is context-aware computing taking control away from the user? three levels of interactivity examined. In *Proceedings of UbiComp*, pages 149–156. Springer Berlin / Heidelberg, 2003. URL http://dx.doi.org/10.1007/b93949.

[11] S. Basu. *Conversational scene analysis*. PhD thesis, MIT Media Lab, 2002.

[12] V. Bellotti and K. Edwards. Intelligibility and accountability: Human considerations in context-aware systems. *Human-Computer Interaction*, 16:193–212, 2001. URL http://dx.doi.org/10.1207/S15327051HCI16234_05.

[13] P. Bessiere and B. R. Group. Survey: Probabilistic methodology and techniques for artefact conception and development. Technical report, INRIA Rhône-Alpes, February 2003.

[14] J. A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-021, University of Berkeley, 1998.

[15] A. Biosca-Ferrer and A. Lux. A visual service for distributed environments: a bayesian 3d person tracker. Technical Report, 2007.

[16] A. Bobick, S. Intille, J. Davis, F. Baird, C. Pinhanez, L. Campbell, Y. Ivanov, A. Schutte, and A. Wilson. The kidsroom: a perceptually-based interactive and immersive story environment. *Presence (USA)*, 8(4):369–393, August 1999.

[17] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM Press. ISBN 0-89791-497-X. URL http://doi.acm.org/10.1145/130385.130401.

[18] O. Brdiczka, J. Maisonnasse, and P. Reignier. Automatic detection of interaction groups. In *Proceedings of International Conference on Multimodal Interfaces (ICMI)*, pages 32–36, October 2005. URL http://doi.acm.org/10.1145/1088463.1088473.

[19] O. Brdiczka, P. Reignier, J. Crowley, D. Vaufreydaz, and J. Maisonnasse. Deterministic and probabilistic implementation of context. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom) Workshops*, pages 46–50, 13-17 March 2006. URL http://dx.doi.org/10.1109/PERCOMW.2006.40.

[20] O. Brdiczka, D. Vaufreydaz, J. Maisonnasse, and P. Reignier. Unsupervised segmentation of meeting configurations and activities using speech activity detection. In *Proceedings of 3rd IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI) 2006, Athens, Greece*, pages 195–203, June 2006. URL http://dx.doi.org/10.1007/0-387-34224-9_23.

[21] O. Brdiczka, P. Yuen, J. Crowley, and P. Reignier. Asimo: Automatic acquisition of situation models. *eMinds: International Journal on Human-Computer Interaction*, 2: 3–15, March 2007. ISSN 1697-9613.

[22] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. Easyliving: technologies for intelligent environments. In *Proceedings of Second International Symposium Handheld and Ubiquitous Computing (HUC)*, pages 12–29. Springer Berlin / Heidelberg, September 2000. URL http://link.springer.de/link/service/series/0558/bibs/1927/19270012.htm.

[23] S. Burger, V. MacLaren, and H. Yu. The isl meeting corpus: The impact of meeting type on speech style. In *Proceedings of International Conference on Spoken Language Processing*, pages 301–304, 2002.

[24] A. Caporossi, D. Hall, P. Reignier, and J. L. Crowley. Robust visual tracking from dynamic control of processing. In *Proceedings of 6th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, Prague, Czech Republic, May 2004.

[25] CAVIAR. Caviar: Context aware vision using image-based active recognition, european commission project ist 2001 37540, 2001. URL http://groups.inf.ed.ac.uk/vision/CAVIAR/.

[26] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[27] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a context-aware electronic tourist guide: some issues and experiences. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24, New York, NY, USA, 2000. ACM Press. ISBN 1-58113-216-6. URL http://doi.acm.org/10.1145/332040.332047.

[28] K. Cheverst, N. Davies, K. Mitchell, and C. Efstratiou. Using context as a crystal ball: Rewards and pitfalls. *Personal Ubiquitous Computing*, 5(1):8–11, 2001. ISSN 1617-4909. URL http://dx.doi.org/10.1007/s007790170020.

[29] K. Cheverst, H. E. Byun, D. Fitton, C. Sas, C. Kray, and N. Villar. Exploring issues of user model transparency and proactive behaviour in an office environment control system. *User Modeling and User-Adapted Interaction*, 15(3-4):235–273, 2005. ISSN 0924-1868. URL http://dx.doi.org/10.1007/s11257-005-1269-8.

[30] T. Choudhury and A. Pentland. Sensing and modeling human networks using the sociometer. In *ISWC '03: Proceedings of the 7th IEEE International Symposium on Wearable Computers*, pages 216–222, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-2034-0. URL http://dx.doi.org/10.1109/ISWC.2003.1241414.

[31] H. H. Clark. *Using Language*. Cambridge University Press, May 1996. URL http://dx.doi.org/10.2277/0521567459.

[32] B. Clarkson. *Life Patterns: Structure from Wearable Sensors*. PhD thesis, MIT Media Lab, September 2002. Supervisor Alex Sandy Pentland.

[33] M. H. Coen. Design principles for intelligent environments. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 547–554, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL http://portal.acm.org/citation.cfm?id=295240.295733.

[34] M. H. Coen. A prototype intelligent environment. In *CoBuild '98: Proceedings of the First International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, pages 41–52. Springer Berlin / Heidelberg, 1998. ISBN 3-540-64237-4. URL http://link.springer.de/link/service/series/0558/bibs/1370/13700041.htm.

[35] D. Cook, M. Youngblood, I. Heierman, E.O., K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. Mavhome: an agent-based smart home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 521–524, 23-26 March 2003. URL http://dx.doi.org/10.1109/PERCOM.2003.1192783.

[36] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1022627411411.

[37] J. Coutaz and G. Rey. Foundations for a theory of contextors. In *Proceedings of the 4th International Conference on Computer Aided Design of User Interfaces*, 2002.

[38] J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan. Context is key. *Communications of the ACM*, 48(3):49–53, 2005. URL http://doi.acm.org/10.1145/1047671.1047703.

[39] J. L. Crowley, J. Coutaz, G. Rey, and P. Reignier. Perceptual components for context aware computing. In *Proceedings of UbiComp*, pages 117–134. Springer London, 2002. ISBN 3-540-44267-7. URL http://link.springer.de/link/service/series/0558/bibs/2498/24980117.htm.

[40] C. U. I. M. P. de Barcelona. "tecnologies de la llengua: darrers avenços" and "llenguatge, cognició i evolució". http://www.cuimpb.es, July 2004.

[41] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1): 4–7, 2001. URL http://dx.doi.org/10.1007/s007790170019.

[42] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16:97–166, 2001. URL http://dx.doi.org/10.1207/S15327051HCI16234_02.

[43] P. Dourish. The appropriation of interactive technologies: Some lessons from placeless documents. *Computer Supported Cooperative Work*, 12(4):465–490, 2003. URL http://dx.doi.org/10.1023/A:1026149119426.

[44] P. Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8:19–30, 2004. URL http://dx.doi.org/10.1007/s00779-003-0253-8.

[45] N. Eagle and A. S. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006. ISSN 1617-4909. URL http://dx.doi.org/10.1007/s00779-005-0046-3.

[46] R. Emonet, D. Vaufreydaz, P. Reignier, and J. Letessier. O3miscid: an object oriented opensource middleware for service connection, introspection and discovery. In *Proceedings of 1st IEEE International Workshop on Services Integration in Pervasive Environments*, June 2006.

[47] T. Erickson. Some problems with the notion of context-aware computing. *Communications of the ACM*, 45(2):102–104, 2002. ISSN 0001-0782. URL http://doi.acm.org/10.1145/503124.503154.

[48] Fame. Fame: Facilitating agent for multi-cultural exchange (wp4), european commission project ist-2000-28323, October 2001.

[49] S. Fine, J. Navratil, and R. Gopinath. A hybrid gmm/svm approach to speaker identification. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 417–420, 7-11 May 2001. URL http://dx.doi.org/10.1109/ICASSP.2001.940856.

[50] R. Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8: 376–386, 1938.

[51] Forum. Forum2004: Universal forum of cultures, barcelona. http://www.barcelona2004.org, July 2004.

[52] S. Greenberg. Context as a dynamic construct. *Human-Computer Interaction*, 16:257–268, 2001. URL http://dx.doi.org/10.1207/S15327051HCI16234_09.

[53] D. Hall, C. L. Gal, J. Martin, O. Chomat, and J. L. Crowley. Magicboard: A contribution to an intelligent office environment. *Robotics and Autonomous Systems*, 35(3-4):211–220, June 2001. URL http://dx.doi.org/10.1016/S0921-8890(01)00125-7.

[54] F. Hoeppner. Learning temporal rules from state sequences. In *Proceedings of IJCAI'01 Workshop on Learning from Temporal and Spatial Data*, pages 25–31, Seattle, USA, 2001.

[55] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[56] Jess. Jess : the rule engine for java. http://herzberg.ca.sandia.gov/jess/.

[57] J. Kay, R. Kummerfeld, and P. Lauder. Managing private user models and shared personas. In *Proceedings of the UM03 Workshop on User Modeling for Ubiquitous Computing*, pages 1–11, Johnstown, PA, 2003.

[58] L. Kleinrock. Nomadic computing - an opportunity. *ACM SIGCOMM Computer Communication Review*, 25(1):36–40, 1995. ISSN 0146-4833. URL http://doi.acm.org/10.1145/205447.205450.

[59] J. D. Lee and K. A. See. Trust in automation: Designing for appropriate reliance. *Human Factors*, 46(1):50–80, 2004. URL http://dx.doi.org/10.1518/hfes.46.1.50.30392.

[60] C. LeGal, J. Martin, A. Lux, and J. L. Crowley. Smart office: Design of an intelligent environment. *IEEE Intelligent Systems*, 16(4):60–66, 2001. ISSN 1541-1672. URL http://dx.doi.org/10.1109/5254.941359.

[61] J. Lindenberg, W. Pasman, K. Kranenborg, J. Stegeman, and M. A. Neerincx. Improving service matching and selection in ubiquitous computing environments: a user study. *Personal Ubiquitous Computing*, 11(1):59–68, 2006. ISSN 1617-4909. URL http://dx.doi.org/10.1007/s00779-006-0066-7.

[62] S. W. Loke. Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. *The Knowledge Engineering Review*, 19(3): 213–233, 2005. URL http://dx.doi.org/10.1017/S0269888905000263.

[63] C. Lueg. Looking under the rug: On context-aware artifacts and socially adept technologies. In *Proceedings of Workshop "The Philosophy and Design of Socially Adept Technologies", ACM SIGCHI Conference on Human Factors in Computing Systems*, 2002.

[64] R. Mayrhofer. *An architecture for context prediction*. PhD thesis, Johannes Kepler University of Linz, 2004.

[65] L. McCowan, D. Gatica-Perez, S. Bengio, G. Lathoud, M. Barnard, and D. Zhang. Automatic analysis of multimodal group actions in meetings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):305–317, March 2005. URL http://dx.doi.org/10.1109/TPAMI.2005.49.

[66] T. M. Mitchell. *Machine Learning*. McGraw Hill, New York, USA, international edition, 1997.

[67] M. C. Mozer. The neural network house: An environment that adapts to its inhabitants. In *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, pages 110–114. Menlo Park, CA, AAAI Press, 1998.

[68] M. Muehlenbrock, O. Brdiczka, D. Snowdon, and J.-L. Meunier. Learning to detect user activity and availability from a variety of sensor data. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 13–23, March 2004. URL http://dx.doi.org/10.1109/PERCOM.2004.1276841.

[69] B. M. Muir. Trust in automation. part i: theoretical issues in the study of trust and human intervention in automated systems. *Ergonomics*, 37(11):1905–1922, November 1994.

[70] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77 (4):541–580, 1989. URL http://dx.doi.org/10.1109/5.24143.

[71] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, Aug. 2000. URL http://dx.doi.org/10.1109/34.868684.

[72] J. Pascoe. Adding generic contextual capabilities to wearable computers. In *Proceedings of Second International Symposium on Wearable Computers (Digest of Papers)*, pages 92–99, 19-20 Oct. 1998. URL http://dx.doi.org/10.1109/ISWC.1998.729534.

[73] A. S. Pentland. Are we one? on the nature of human intelligence. In *Proceedings of 5th International Conference on Development and Learning*, Bloomington, May 31-June 3 2006.

[74] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut fuer Instrumentelle Mathematik, Bonn, Germany, 1962.

[75] J. C. Platt. *Probabilities for SV Machines*, chapter 5, pages 61–74. MIT Press, 1999.

[76] J. Puzicha, T. Hofmann, and J. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 267–272, 17-19 June 1997. URL http://dx.doi.org/10.1109/CVPR.1997.609331.

[77] R. Qian, M. Sezan, and K. Mathews. Face tracking using robust statistical estimation. In *Proceedings of Workshop on Perceptual User Interfaces*, San Francisco, 1998.

[78] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1022643204877.

[79] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. URL http://dx.doi.org/10.1109/5.18626.

[80] P. Reignier and J. Crowley. Context model specification tool. Caviar IST Project Deliverable, September 2004. URL http://www-prima.inrialpes.fr/reignier/Documents/caviar.pdf.

[81] P. Ribeiro and J. Santos-Victor. Human activity recognition from video: modeling, feature selection and classification architecture. In *Proceedings of International Workshop on Human Activity Recognition and Modelling (HAREM)*, pages 61–70, 2005.

[82] A. Salovaara and A. Oulasvirta. Six modes of proactive resource management: a user-centric typology for proactive behaviors. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 57–60, New York, NY, USA,

2004. ACM Press. ISBN 1-58113-857-1. URL http://doi.acm.org/10.1145/1028014.1028022.

[83] M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10–17, August 2001. URL http://dx.doi.org/10.1109/98.943998.

[84] B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, Sept.-Oct. 1994. URL http://dx.doi.org/10.1109/65.313011.

[85] B. N. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994. URL http://dx.doi.org/10.1109/MCSA.1994.512740.

[86] A. Schmidt. *Ambient Intelligence*, chapter 9, pages 159–178. IOS Press, 2005.

[87] A. Schmidt, M. Beigl, and H.-W. Gellersen. There is more to context than location. *Computers and Graphics*, 23(6):893–901, 1999. URL http://citeseer.ist.psu.edu/schmidt98there.html.

[88] Y. Shi, W. Xie, G. Xu, R. Shi, E. Chen, Y. Mao, and F. Liu. The smart classroom: Merging technologies for seamless tele-education. *IEEE Pervasive Computing*, 2(2):47–55, 2003. ISSN 1536-1268. URL http://dx.doi.org/10.1109/MPRV.2003.1203753.

[89] N. A. Streitz, C. Rocker, T. Prante, D. van Alphen, R. Stenzel, and C. Magerkurth. Designing smart artifacts for smart environments. *Computer*, 38(3):41–49, 2005. ISSN 0018-9162. URL http://dx.doi.org/10.1109/MC.2005.92.

[90] L. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press, 1987.

[91] D. Vaufreydaz. Ist-2000-28323 fame: Facilitating agent for multi-cultural exchange (wp4). Deliverable, October 2001. European Commission project IST-2000-28323.

[92] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco. Probabilistic finite-state machines - part ii. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1026–1039, July 2005. URL http://dx.doi.org/10.1109/TPAMI.2005.148.

[93] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco. Probabilistic finite-state machines - part i. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025, July 2005. URL http://dx.doi.org/10.1109/TPAMI.2005.147.

[94] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, 1991.

[95] M. Weiser and J. S. Brown. Designing calm technology. *PowerGrid Journal*, 1(1), 1996.

[96] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report TR 95-041, University of North Carolina at Chapel Hill, 2004.

[97] T. Winograd. Architectures for context. *Human-Computer Interaction*, 16:401–419, 2001. URL http://dx.doi.org/10.1207/S15327051HCI16234_18.

[98] W. Xie, Y. Shi, G. Xu, and D. Xie. Smart classroom - an intelligent environment for tele-education. In *PCM '01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia*, pages 662–668. Springer Berlin / Heidelberg, 2001. ISBN 3-540-42680-9. URL http://link.springer.de/link/service/series/0558/bibs/2195/21950662.htm.

[99] G. M. Youngblood, L. B. Holder, and D. J. Cook. Managing adaptive versatile environments. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 351–360, 2005. URL http://dx.doi.org/10.1109/PERCOM.2005.23.

[100] S. Zaidenberg, O. Brdiczka, P. Reignier, and J. Crowley. Learning context models for the recognition of scenarios. In *Proceedings of 3rd IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI) 2006, Athens, Greece*, pages 86–97, June 2006. URL http://dx.doi.org/10.1007/0-387-34224-9_11.

[101] D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, and G. Lathoud. Multimodal group action clustering in meetings. In *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 54–62, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-934-9. URL http://doi.acm.org/10.1145/1026799.1026810.

[102] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Modeling individual and group actions in meetings with layered hmms. *IEEE Transactions on Multimedia*, 8(3):509–520, June 2006. URL http://dx.doi.org/10.1109/TMM.2006.870735.

[103] S. K. Zhou, R. Chellappa, and B. Moghaddam. Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*, 13(11):1491–1506, Nov. 2004. URL http://dx.doi.org/10.1109/TIP.2004.836152.

Index