SAPIENZA
UNIVERSITÀ DI ROMA

Instrumenting and Mining Smart Spaces

Dipartimento di Ingegneria Informatica, Automatica e Gestionale A. Ruberti,
Università di Roma "La Sapienza"

Dottorato di Ricerca in Ingegneria Informatica – XXVI Ciclo

Candidate

Francesco Leotta
ID number 799650

Thesis Committee

Prof. Massimo Mecella
Prof. Luca Iocchi

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Engineering in Computer Science

March 2014

Thesis not yet defended

Instrumenting and Mining Smart Spaces
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Website: http://www.dis.uniroma1.it/∼leotta
Author's email: leotta@dis.uniroma1.it

Contents

Extended Abstract iv

1 Research Context and Background 1
- 1.1 Sensors and Actuators . . . 4
  - 1.1.1 Sensors . . . 4
  - 1.1.2 Actuators . . . 6
- 1.2 Learning and Representing the Knowledge . . . 7
  - 1.2.1 Handling Quantitative Sensor Data . . . 7
  - 1.2.2 Managing Uncertainty . . . 8
  - 1.2.3 The Sensor Data Windowing Problem . . . 9
- 1.3 Decision Making . . . 11
- 1.4 Context Extraction . . . 13
  - 1.4.1 Specification-based methods . . . 14
  - 1.4.2 Supervised Learning-based methods . . . 17
  - 1.4.3 Unsupervised Learning-based methods . . . 20
- 1.5 Software Platforms and Middleware . . . 22

2 Inhabitants in the Viewfinder 23
- 2.1 Preliminaries . . . 23
  - 2.1.1 Marker-less vs Marker-based Systems . . . 25
  - 2.1.2 Sensing Technology . . . 25
  - 2.1.3 Identity Recognition Capabilities . . . 28
  - 2.1.4 Level of Support for Posture Estimation . . . 28
  - 2.1.5 Physical vs Symbolic Coordinates . . . 29
- 2.2 Adopted Techniques . . . 29
  - 2.2.1 Foreground Extraction . . . 31
  - 2.2.2 Tracking and Pose Estimation . . . 31
  - 2.2.3 Identity Recognition . . . 32
- 2.3 Software and System Architecture . . . 33
  - 2.3.1 Single-Instance Software Architecture . . . 33
  - 2.3.2 System Architecture . . . 42
- 2.4 Deployment . . . 44
  - 2.4.1 Deployment of a single instance . . . 45
- 2.5 Experimental Evaluation . . . 47
  - 2.5.1 Position Test . . . 47

2.5.2 Robustness to Sudden Illumination Changes and Background Changes . . . 52
2.5.3 Performance of Identity Recognition . . . 52
2.6 Related Work . . . 53
2.7 Discussion and Conclusions . . . 55
2.8 Appendix: Details on Techniques . . . 57
2.8.1 Foreground Extraction . . . 57
2.8.2 Tracking and Pose Estimation . . . 60

3 The Quest for Datasets 63
3.1 Dataset Generation . . . 63
3.1.1 Preliminaries . . . 64
3.1.2 Dataset Generation Strategy . . . 67
3.1.3 Habit Modeling and Scheduling . . . 68
3.1.4 Continuous Planning . . . 69
3.1.5 Example . . . 70
3.1.6 Virtual Environment Execution . . . 71
3.1.7 Evaluation . . . 72
3.1.8 Related Work and Limitations . . . 73
3.2 Gamification and Crowdsourcing . . . 74
3.3 Future Work: Mixing through Mining . . . 76

4 Mining Smart Spaces 77
4.1 Mining Constraints for Decision Making . . . 77
4.1.1 Preliminaries . . . 78
4.1.2 Restricted Itemsets . . . 80
4.1.3 Constraint Mining Algorithm Description . . . 82
4.1.4 Validation . . . 83
4.1.5 Related Work . . . 88
4.2 Extracting Habits Using Process Mining . . . 89
4.2.1 Preliminaries . . . 89
4.2.2 The Proposed Technique . . . 96

5 Conclusions and Future Work 100

Extended Abstract

The progress of information and communication technologies (ICTs) has many faces; while computing speed, reliability and level of miniaturization of electronic devices increase year after year, their costs decrease. This allows a widespread adoption of *embedded systems* (e.g., appliances, sensors, actuators) and of powerful computing devices (e.g., laptop, smartphones), thus turning *pervasive* (or *ubiquitous*) computing into reality. Pervasive computing embodies a vision of computers seamlessly integrating into everyday life, responding to information provided by sensors in the environment, with little or no direct instruction from users [203]. At the same time, connecting all these computing devices together, as *networked artefacts*, using local and global network infrastructures has become easy. The rise of *applications* that exploit these technologies represents a major characteristic of the *Internet of Things* (IoT) [191].

Cyber-physical systems (CPSs) represent an emerging class of IoT-based applications. A CPS is defined in [117, 164] as the "integration of computation with physical processes" where "embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa". As simple and general as this definition is, nowadays its meaning is strongly influenced by the rise of IoT [39]. IoT is a key driver for current CPS design and, in particular, the Internet represents the element that connects the system to cyberspace services and data, hence the users are supposed to interact with a CPS using both their digital identities and the physical interaction.

The concept of *smart space*, for example, can be easily extended to model a CPS. The universAAL specification [189] defines a smart space as "an environment centered on its human users in which a set of embedded networked artefacts, both hardware and software, collectively realize the paradigm of *ambient intelligence*". In order for this definition to cover the CPS main features, the smart environment has to be intended as including both the physical space and the digital world where useful services for the control of the physical space can be retrieved.

Many different definitions of ambient intelligence (AmI) are provided in the literature, e.g., [58] introduces a set of distinguishing keywords characterizing AmI systems, namely: sensitivity, responsiveness, adaptivity, ubiquity and transparency. The term *sensitivity* refers to the ability of an AmI system to sense the environment and, more generally, to understand the *context* it is interacting with. Strictly related to sensitivity are *responsiveness* and *adaptivity*, which denote the capability to timely act, in a reactive or proactive manner, in response to changes in the context according to user preferences (*personalization*). Sensitivity, responsiveness and adaptivity all contribute to the concept of *context awareness*. Finally, the terms *ubiquity* and

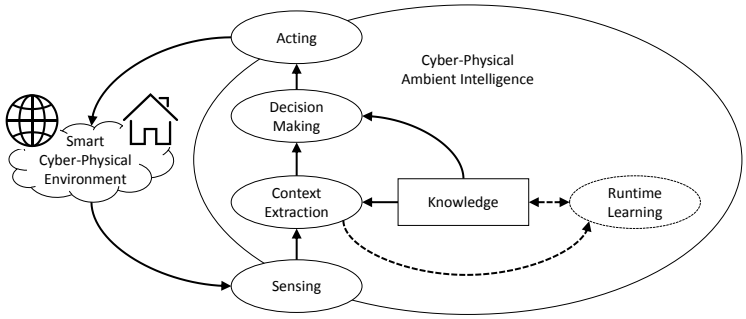


Figure 1. The ambient intelligence closed loop.

transparency directly refer to concept of pervasive computing. AmI is obtained by merging techniques from different research areas [189] including artificial intelligence (AI) and human-computer interaction (HCI).

Among smart spaces, smart homes and offices are representative examples. They are an evolution of the concept of building automation as this term refer to a certain level of automation in the environment that does not include what we denoted as AmI.

Figure 1 depicts the closed loops that characterize a running smart cyber-physical space [21]. The main closed loop, depicted using solid arrows and shapes, shows how the *knowledge* of environment dynamics and of users behaviors and preferences is employed to interpret *sensors* output in order to perform appropriate *actions* on the environment. Sensor data is first analyzed to extract the current *context*, which is an internal abstraction of the state of the environment from the point of view of the AmI system. The extracted context is then employed to *make decisions* on the actions to perform on the controlled space. Actions related to these decisions modify the environment (both physical and digital) by means of *actuators* of different forms; they can be physical objects that directly change the state of the environment or, alternatively, they can be software entities that invoke, following the CPS philosophy, Internet services using the digital identities of the users.

It is noteworthy that AmI is not intended to be provided by a centralized entity but, to the contrary, its nature is distributed with embedded devices and software modules, possibly unaware one of each other, contributing to its features.

Knowledge plays a central role in AmI systems. As it intervenes both for context extraction and decision making, it takes the form of a set of models describing *(i)* users behavior, *(ii)* environment/device dynamics, and *(iii)* user preferences.

Knowledge should not be considered as a static resource as both users behavior and preferences change over time. Vast majority of works in the area of ambient intelligence suppose the knowledge to be obtained off-line, independently from the system runtime. A second optional loop in Figure 1, depicted using dashed arrows, shows that the current context could be employed to update the knowledge by applying learning techniques at runtime.

The way an AmI system make decisions on the actions can be compared to decision making in AI agents. For example, *reflex agents with state*, as introduced in [178], take as input the current state of the world and a set of Condition-Action

rules to choose the action to be performed.

Models employed for decision making should not concern themselves with the individual pieces of sensor data (which room the user is in, what his heart rate or blood pressure is): rather, this information should be interpreted into a higher, domain-relevant concept, such as whether the user is suffering a heart attack or exercising. Thus between sensing and decision making we put an additional layer called *context extraction*.

Sensor data is aggregated into *context objects* (COs), which symbolically represent the environment in which the system operates. A CO expresses what can be said about one or more *entities*. For example, "John is in the Kitchen" is a CO where John is an entity of type `person` and the Kitchen is a the value of the context `location`. A situation is instead an abstraction of the events occurring in the real world derived from COs and hypotheses about how observed context relates to factors of interest for designers and applications. Situations typically fuse several CO sources, as well as models of the expected behavior of the phenomena being observed.

Research Contributions

Sensors and actuators represent respectively the input and the output units of an AmI system with respect to the controlled environment. Their choice strongly influences the ability of an AmI system to interpret the current context and, in turn, to fulfill users needs and wishes.

The list of available sensors have a deep impact on context extraction and situation identification techniques [125]. The required contextual information depends, in turn, on the final goals the system is intended to achieve. If, for example, the goal is "energy consumption monitoring", it will be easier for the system to achieve it if amp meters are available instead of simple switch sensors connected to the appliances. Similarly, as an AmI system is supposed to provide customized services to users, their positions are fundamental to this aim; the availability of specifically designed sensors, called *indoor localization systems*, eases this task with respect to inferring positions by employing, for example, switch sensors connected to doors.

Equivalently, the availability of a certain set of actuators directly determines the ability of the AmI system to perform the actions in the controlled environment that are needed to pursue a specific goal.

O1 We define "instrumenting a smart space" as the task of choosing and designing sensors and actuators in order to maximize the performance of AmI techniques employed in a specific smart space. One of the objectives of this thesis is proposing methods that support this task.

The correct design and deployment of smart spaces in terms of sensors and actuators (from the practitioner's point of view), as well as the definition of novel architectures, techniques and methods (from the researcher's point of view) require extensive experimental facilities. Some freely-available datasets are published in the

research community (cf. CASAS[1], MIT House_n[2] and Tracebase [170][3] projects); smart facilities are planned to be built in the nearby future (cf. EU initiatives for Living Labs[4], US NIST initiative for pervasive computing [176], etc.).

Unfortunately, available installations and testbeds only contain limited types and number of sensors that are not sufficient to validate algorithms for smart spaces in a thorough way; without customizable datasets available, it is hard to design a smart space, in such a way that maximizes employed algorithms performance. In order to face this issue, CASAS project has recently launched the "Smart Home in a Box" program [60][5], which is intended to employ crowdsourcing to gather a huge amount of sensor data from worldwide deployed installations.

With reference to objective **O1**, the following research contributions have been achieved:

R1-1 In [118] we introduce PLaTHEA, an indoor localization system intended to be employed as a complex sensor into a smart space. The system provides at any given time the position and the identities of the users that populate the monitored environment. Differently from other proposed localization systems, PLaTHEA provides clear interfaces for the integration into an ambient intelligence platform. Additionally, as the system is designed to be easily installed into a smart space, a methodology to employ network attached (IP) cameras is introduced; in particular, a technique for dealing with not synchronized video streams on local area networks is described.

R1-2 The evaluation of indoor localization systems has been usually addressed by academy using ad-hoc methods that do not ensure a good performance of the proposed systems in real life conditions. In [119] we proposed an evaluation platform for indoor localization systems called PERICLES. In this work we introduce a test methodology and tool supporting it, which allow to evaluate every kind of indoor localization system independently from the underlying technology (e.g., computer vision, radio transmission). This evaluation platform has been employed to perform test on PLATHEA [118]; [118] adds a set of new performance measures to the original paper [119].

R1-3 A possible approach to deal with the lack of a sufficient amount of dataset to evaluate techniques for AmI against, is the definition of dataset generation techniques. In [47, 46] we propose a configurable dataset generation tools able to generate realistic logs based on a virtual smart space consisting of agents that behave "as if" they were real inhabitants of the actual smart space. The tool takes as input a set of hand-made models of human habits expressed using a declarative process modeling language. Such a virtual environment can be configured to contain a certain set of sensors and actuators, hence allowing to evaluate technique for AmI over a set of different smart space configurations.

[1] http://wsucasas.wordpress.com/
[2] http://architecture.mit.edu/house_n/
[3] http://www.tracebase.org/
[4] http://www.openlivinglabs.eu/
[5] http://smarthome.ailab.wsu.edu/

R1-4 An additional approach for gathering datasets of human habits is crowdsourcing by means of serious games that allow people to perform their habits inside a virtual environment. In this thesis we present the first results obtained with respect to this approach. Dataset obtained this way can be also employed to enhance the tool presented in [47, 46] (see **R1-3**), whose main drawback is that the models that the dataset generation tool takes as input are hand-made. Such a facility would allow to obtain models describing human habits employing mining techniques (see **R2-2**).

Intelligent techniques for AmI rely on a certain degree of knowledge expressed using a set of models of human behavior and environmental dynamics. Models in the literature can be roughly divided into *specification-based* and *learning-based* [203]. Research in the field of AmI started when few kinds of sensors were available and the relationships between sensor data and underlying phenomena were easy to establish. Specification-based approaches represent hand-made *expert* knowledge in logic rules and apply reasoning engines to infer conclusions and to make decisions from sensor data. The growing availability of different kind of sensors made hand-made models impractical to be produced. In order to solve this problem, learning-based methods employ techniques from *machine learning* [34] and *data mining* [93].

Learning-based techniques can be divided into *supervised* and *unsupervised* techniques. The former expect the input to be previously labeled according to the required output function, hence they require a big effort for organizing input data in terms of training examples, even though active learning can be employed to ease this task. Unsupervised techniques (or weakly supervised ones) can be used to face this challenge but a limited number of works is available in the literature; a particular category of these techniques comes from the so called *temporal data mining* research field [84], which include *(i)* classification, *(ii)* clustering, *(iii)* pattern and rule discovery, and *(iv)* prediction techniques.

Unsupervised techniques for AmI knowledge modeling can be useful for other two reasons. In first place, as previously stated, knowledge should not be considered as a static resource; instead it should be updated at runtime without a direct intervention of the users [167], hence updating techniques should rely on labeling of sensor data as little as possible. In second place, unsupervised techniques may also result useful in supporting *passive users*, such as guests, that do not participate in the configuration of the system but should benefit from its services as well.

O2 We define "mining a smart space" as the task of extracting knowledge from a smart space with a few intervention from the users in terms of labeling effort. This task is a major challenge in the AmI research field. One of the objectives of this thesis is to employ human readable formalisms in conjunction with unsupervised machine learning techniques to pursue this task.

With reference to objective **O2**, the following research contributions have been achieved:

R2-1 Reactive rules represent a standard way for AmI system to deal with decision making. In [70] we introduce a mining techniques to obtain reactive rules (constraints) from a previously acquired, not labeled, sensor log. The proposed method introduces the notion of restricted itemset, which models the concept of an unwanted sensor value combination. Restricted itemsets can then be transformed into reactive rules that automatically trigger whenever the unwanted situation is detected.

R2-2 In this thesis, we also outline a solution to tackle the problem of mining models of human habits by employing techniques borrowed from process mining. In particular, we discuss a technique to bridge the gap between sensor logs produced by smart spaces and process execution traces that are usually given as input to process mining algorithms.

Parts of our work were published in the following papers. Each is labeled with the research challenges addressed. Research challenges **R1-4** and **R2-2**, which are still unpublished, will be discussed in Chapter 3 and Chapter 4 respectively.

[119] **F. Leotta, M. Mecella, F. Aloise**
PERICLES: A Performance Evaluation Platform for Indoor Localization Systems. 3rd International Workshop on Indoor Spatial Awareness (ISA 2011), Chicago, USA, 1 November 2011, held in conjunction with the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS 2011).
Research Contributions: **R1-2**

[47] **M. Caruso, F. Leotta, M. Mecella, S. Vassos**
Benchmarking Smart Spaces Through Autonomous Virtual Agents (Extended Abstract). 12th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2013), Saint Paul, USA, 6–10 May 2013.
Research Contributions: **R1-3**

[46] **M. Caruso, C. Ilban, F. Leotta, M. Mecella, S. Vassos**
Synthesizing Daily Life Logs Through Gaming and Simulation. 2nd Workshop on Recent Advances in Behavior Prediction and Pro-Active Pervasive Computing (AwareCast 2013), Zurich, Switzerland, 8 September 2013, held in conjunction with 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013).
Research Contributions: **R1-3**

[70] **V. Degeler, A. Lazovik, F. Leotta, M. Mecella**
Itemset-based Mining of Constraints for Enacting Smart Environments. 1st Symposium on Activity and Context Modeling and Recognition (ACOMORE 2014), Budapest, Hungary, 24 March 2014, held in conjunction with the 2014 IEEE International Conference on Pervasive Computing and Communications (PerCom 2014).
Research Contributions: **R2-1**

[118] **F. Leotta, M. Mecella**
PLaTHEA: A Markerless People Localization and Tracking System for Home Automation. Software: Practice and Experience (SPE), To Appear.
Research Contributions: **R1-1**, **R1-2**

Thesis Outline

- Chapter 1 introduces background concepts and definitions related to smart spaces and ambient intelligence, and provides a systematic view of the different

approaches and methodologies that have emerged to support them. This serves as the basis for positioning the performed work.

- Chapter 2 first introduces a taxonomy of localization and tracking systems focusing on indoor ones. Then PLaTHEA – People Localization and Tracking for HomE Automation – system is explained in terms of employed techniques; this part of the thesis also serves as a short guide to the modules necessary for a computer vision based localization system. PLaTHEA is then described as a component of an AmI architecture where a localization system is seen as an additional sensor whose measurements are accessed through well defined service-based interfaces. In the last part of the chapter the PERICLES evaluation platform is described and employed to test PLaTHEA. Reported accuracy and precision results are then compared to those of obtained by indoor localization systems proposed in both the commercial and the academic field.
- Chapter 3 first introduces a platform designed for generating realistic datasets of human activities into a smart space. At the current stage the platform employs hand-made models of human habits written using a formalism for declarative process modeling. In the last part of the chapter a platform for obtaining this data via crowdsourcing is described and a future strategy to merge the first and the second approach is outlined.
- Chapter 4 first introduces a method based on itemset mining to mine constraints in smart spaces. These constraints can be employed to produce automatic rules that triggers whenever a non conventional context is detected. In the second part of the chapter the topic of applying process mining to human habits is tackled.
- Chapter 5 concludes the discussion and traces some future developments that might arise from the basis of this work.

Additional Work

During the Ph.D. programme, the author has been involved in two additional research projects connected to the area of smart spaces.

People affected by severe motor disabilities may find in brain-computer interfaces (BCIs) an effective way to communicate with their surroundings and to act on them. Unfortunately, very few steps have been done in bringing this king of technology out of research laboratories.

The author has been a member (in the years 2010-2011) of the TOBI[6] – Tools for Brain-Computer Interfaces – FP7 European Research Project, whose aim was to develop practical technology for brain-computer interaction (BCI) to improve the quality of life of disabled people and the effectiveness of rehabilitation. In the context of this project the author worked on topics related to bio-signal real-time processing and human computer interaction.

The author has been also a member (in the years 2012-2013) of the BrIndiSys[7] – brain-computer interface devices to support individual autonomy in locked-in indi-

[6] http://www.tobi-project.org/
[7] http://www.brindisys.it/

viduals – project, funded by AriSLA – Italian Agency for Research on Amyotrophic Lateral Sclerosis (ALS), which was aimed at providing users with disabilities with the tool useful for interacting with a smart space. During the project the author faced issues connected to those approached during the TOBI project.

We report the contributions to these projects here in terms of published papers for sake of completeness, though the themes covered in their subject are out of scope for this thesis.

[111] **S. Kleih, T. Kaufmann, C. Zickler, S. Halder, F. Leotta, F. Cincotti, F. Aloise, A. Riccio, C. Herbert, D. Mattia**
Out of the frying pan into the fire–the P300-based BCI faces real-world challenges. Progress in brain research, 2010, 196:27–46.

[13] **F. Aloise, F. Schettini, P. Aricò, F. Leotta, S. Salinari, D. Mattia, F. Babiloni, F. Cincotti**
Toward Domotic Appliances Control through a Self-paced P300-based BCI. 2011 International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSIGNALS 2011), Rome, Italy, 26–29 January 2011.

[173] **A. Riccio, F. Leotta, L. Bianchi, F. Aloise, C. Zickler, E.J. Hoogerwerf, A. Kübler, D. Mattia, F. Cincotti**
Workload measurement in a communication application operated through a P300-based brain–computer interface. Journal of neural engineering, 2011, 8(2).

[12] **F. Aloise, F. Schettini, P. Aricò, F. Leotta, S. Salinari, D. Mattia, F. Babiloni, F. Cincotti**
P300-based brain–computer interface for environmental control: an asynchronous approach. Journal of neural engineering, 2011, 8(2).

[144] **G. Müller-Putz, C. Breitwieser, F. Cincotti, R. Leeb, M. Schreuder, F. Leotta, M. Tavella, L. Bianchi, A. Kreilinger, A. Ramsay**
Tools for brain-computer interaction: a general concept for a hybrid BCI. Frontiers in neuroinformatics, 2011, 5.

[208] **C. Zickler, A. Riccio, F. Leotta, S. Hillian-Tress, S. Halder, E. Holz, P. Staiger-Sälzer, E.J. Hoogerwerf, L. Desideri, D. Mattia**
A brain-computer interface as input channel for a standard assistive technology software. Clinical EEG and Neuroscience, 2011, 42(4):236–244.

[145] **G. Müller-Putz, C. Breitwieser, M. Tangermann, M. Schreuder, M. Tavella, R. Leeb, F. Cincotti, F. Leotta, C. Neuper**
Tobi hybrid BCI: principle of a new assistive method. International Journal of Bioelectromagnetism, 2011, 13(3):144–145.

[172] **A. Riccio, F. Leotta, F. Aloise, L. Bianchi, D. Mattia, F. Cincotti**
Evaluation of a P300 overlaid stimulation for controlling an assistive technology software. International Journal of Bioelectromagnetism, 2011, 13(3):141–143.

[55] **F. Cincotti, F. Pichiorri, P. Arico, F. Aloise, F. Leotta, F. de Vico Fallani, J. del R Millan, M. Molinari, D. Mattia**
EEG-based Brain-Computer Interface to support post-stroke motor rehabilitation of the upper limb. 2012 IEEE International Conference on Engineering in Medicine and Biology Society (EMBC 2012), San Diego, USA, 28–31 August 2012.

[45] **M. Caruso, F. Cincotti, F. Leotta, M. Mecella, A. Riccio, F. Schettini, L. Simione, T. Catarci**
My-World-in-My-Tablet: An Architecture for People with Physical Impairment. 15th International Conference on Human-Computer Interaction. Interaction Modalities and Techniques (HCII 2013), Las Vegas, USA, 21–26 July 2013.

Chapter 1

Research Context and Background

The progress of information and communication technologies (ICTs) has many faces; while computing speed, reliability and level of miniaturization of electronic devices increase year after year, their costs decrease. This allows a widespread adoption of *embedded systems* (e.g., appliances, sensors, actuators) and of powerful computing devices (e.g., laptop, smartphones), thus turning *pervasive* (or *ubiquitous*) computing into reality. Pervasive computing embodies a vision of computers seamlessly integrating into everyday life, responding to information provided by sensors in the environment, with little or no direct instruction from users [203]. At the same time, connecting all these computing devices together, as *networked artefacts*, using local and global network infrastructures has become easy. The rise of *applications* that exploit these technologies represents a major characteristic of the *Internet of Things* (IoT) [191].

Cyber-physical systems (CPSs) represent an emerging class of IoT-based applications. A CPS is defined in [117, 164] as the "integration of computation with physical processes" where "embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa". As simple and general as this definition is, nowadays its meaning is strongly influenced by the rise of IoT [39]. IoT is a key driver for current CPS design and, in particular, the Internet represents the element that connects the system to cyberspace services and data, hence the users are supposed to interact with a CPS using both their digital identities and the physical interaction.

The concept of *smart space*, for example, can be easily extended to model a CPS. The universAAL specification [189] defines a smart space as "an environment centered on its human users in which a set of embedded networked artefacts, both hardware and software, collectively realize the paradigm of *ambient intelligence*". In order for this definition to cover the CPS main features, the smart environment has to be intended as including both the physical space and the digital world where useful services for the control of the physical space can be retrieved.

Many different definitions of ambient intelligence (AmI) are provided in the literature, e.g., [58] introduces a set of distinguishing keywords characterizing AmI systems, namely: sensitivity, responsiveness, adaptivity, ubiquity and transparency.

The term *sensitivity* refers to the ability of an AmI system to sense the environment and, more generally, to understand the *context* it is interacting with. Strictly related to sensitivity are *responsiveness* and *adaptivity*, which denote the capability to timely act, in a reactive or proactive manner, in response to changes in the context according to user preferences (*personalization*). Sensitivity, responsiveness and adaptivity all contribute to the concept of *context awareness*. Finally, the terms *ubiquity* and *transparency* directly refer to concept of pervasive computing. AmI is obtained by merging techniques from different research areas [189] including artificial intelligence (AI) and human-computer interaction (HCI).

Smart spaces can have an immediate impact on our society by pursuing important goals such as:

- *Daily Life and Work Experience.* As the amount of time spent indoors is considerable, improving the daily life and work experience is a fundamental concern for smart spaces such as smart homes and offices. In order to do this the smart space should be aware of users *habits* and assist them during their *routines*.

- *Energy Saving.* Energy consumption represents an important concern in modern buildings. As the price of energy increases and laws are issued in order to reduce pollution, a smart space that preserves energy may result in a huge money saving for owners and public administrations.

- *Safety and Security.* Users in private and public spaces are liable to a series of threats that can be either accidental (i.e., safety threats) or malicious (i.e., security threats). An important trend of research in AmI has been focused on the detection of such kind of threats.

Among smart spaces, smart homes and offices are representative examples. They are an evolution of the concept of building automation as this term refer to a certain level of automation in the environment that does not include what we denoted as AmI.

Smart homes represent controlled environments where pervasive computing could take advantage of ambient intelligence more easily than in other spaces where AI problems soon become intractable [17]. In addition to the services above mentioned, smart homes are often analyzed with regards to Ambient Assisted Living (AAL), which is the discipline aimed at automatizing homes such as those kind of services can be provided automatically, allowing user to independently continuing to perform the so called Activities of Daily Living (ADL) [109]. AAL is strictly connected to the problem of sustainable aging, which arises from the progressively aging population of western countries that generates increasing costs for providing public and private health services.

A study about the current level of adoption of commercial smart home systems is provided in [137]. This study reveals how people understanding of the term "smart" has a more general meaning than what we presented here as AmI; in particular it also includes non-technological aspects such as the spatial layout of the house. Additionally, an automated behavior is considered as smart, especially from people without a technical background, only if it performs a task quicker than the user could

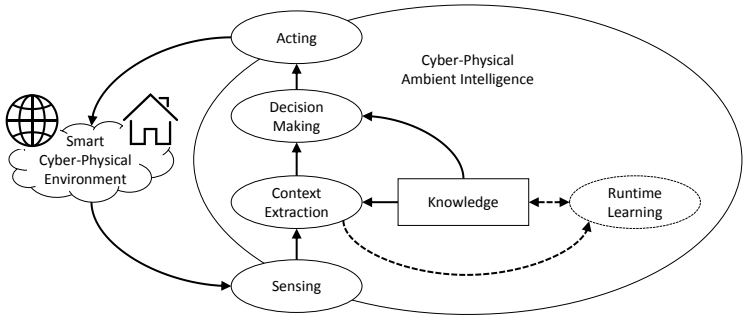


Figure 1.1. The ambient intelligence closed loop.

do by himself. The research also reveals that interest in smart homes systems is subject to a virtuous circle such that people experiencing benefits from their services feel the need of upgrading them.

Smart offices started raising a certain interest in research community during the early 90's [165]. The aim of a smart office is basically to improve productivity by easing work experience. Services offered by such kind of facilities range from automatic call transfer according to users last detected positions to visitors automatic visit guidance to decision support during meetings.

Figure 1.1 depicts the closed loops that characterize a running smart cyber-physical space [21]. The main closed loop, depicted using solid arrows and shapes, shows how the *knowledge* of environment dynamics and of users behaviors and preferences is employed to interpret *sensors* output in order to perform appropriate *actions* on the environment. Sensor data is first analyzed to extract the current *context*, which is an internal abstraction of the state of the environment from the point of view of the AmI system. The extracted context is then employed to *make decisions* on the actions to perform on the controlled space. Actions related to these decisions modify the environment (both physical and digital) by means of *actuators* of different forms; they can be physical objects that directly change the state of the environment or, alternatively, they can be software entities that invoke, following the CPS philosophy, Internet services using the digital identities of the users.

It is noteworthy that AmI is not intended to be provided by a centralized entity but, to the contrary, its nature is distributed with embedded devices and software modules, possibly unaware one of each other, contributing to its features. Recently introduced smart space appliances, such as the NEST Thermostat, contain sensors, actuators and AmI features in a single small package.

Knowledge plays a central role in AmI systems. As it intervenes both for context extraction and decision making, it takes the form of a set of models describing *(i)* users behavior, *(ii)* environment/device dynamics, and *(iii)* user preferences. Anyway, knowledge should not be considered as a static resource as both users behavior and preferences change over time. Vast majority of works in the area of ambient intelligence suppose the knowledge to be obtained off-line, independently from the system runtime. A second optional loop in Figure 1.1, depicted using dashed arrows, shows that the current context could be employed to update the knowledge by applying learning techniques at runtime.

1.1 Sensors and Actuators

Sensors and actuators represent respectively the input and the output units of an AmI system with respect to the controlled environment. It is worth to note here that those actuators that include sensors that monitor the current state of the controlled device can be considered as belonging to both the categories.

As the history of home and building automation started many years before what we call AmI, a set of commercial standards is available for connecting these units together in order to read measurements from sensors and perform actions through actuators. These standards are still a major component of every AmI system as they represent the lowest level of every proposed architecture (see Section 1.5).

The X10 standard was introduced back in 1975 but it remains very popular with millions of units installed. In this standard, digital information travels on preexisting power lines (PLC – Power Line Communication) in the form of radio frequency bursts. Modern standards, such as KNX (pronounced "konnex"), support a broader band, a larger set of transmission medium (including both power lines and ad-hoc buses), and more sophisticated addressing technologies.

Beside the above mentioned wired standards, wireless standards are nowadays available. Such standards were originally intended to be used in WSN – Wireless Sensor Networks where a set of sensor nodes (also called *motes*) are connected together to monitor a specific environment. Modern wireless standards, such as Zigbee and Z-Wave, can connect heterogeneous nodes including both sensors and actuators.

1.1.1 Sensors

Sensing technologies have made significant progress on designing sensors with smaller size, lighter weight, lower cost, and longer battery life[1]. Sensors can thus be embedded in an environment and integrated into everyday objects and onto human bodies without affecting users comfort. Nowadays, sensors do not only include those traditionally employed for home and building automation (e.g. presence detectors, smoke detectors, contact switches for doors and windows, network-attached and close circuit cameras) but also more modern units (e.g. IMUs - Inertial Measurements Units such as accelerometer and gyroscopes, WSN nodes), which are growingly available as off-the-shelf products.

Sensors can be roughly divided in *physical* ones, which provide information about the environment (e.g., humidity, brightness, temperature), the devices and the users, and *cyber* ones, which provide digital information such as user calendars and weather forecasts. The list of available sensors have a deep impact on context extraction and situation identification techniques [125], which will be the subject of Section 1.4. The required contextual information depends, in turn, on the final goals the system is intended to achieve. If, for example, the goal is "energy consumption monitoring", it will be easier for the system to achieve it if amp meters are available instead of simple switch sensors connected to the appliances. Similarly, as an AmI system is supposed to provide customized services to users, their positions are fundamental to this aim; the availability of specifically designed sensors, called *indoor localization*

[1] This trend has been noticeably accelerated by the advent of Systems-on-a-Chip (SOCs).

systems, which will be the subject of Chapter 2, eases this task with respect to inferring positions by employing, for example, switch sensors connected to doors.

The term *sensor data* encompasses raw (or minimally-processed) data retrieved from both physical sensors and cyber sensors. We can imagine a smart space producing, at runtime, a *sensor log* containing raw measurements from available sensors.

Definition 1.1 (Sensor Log)**.** *Given a set S of sensors, a sensor log is a sequence of measurements of the kind $\langle ts, s, v \rangle$ where ts is the timestamp of the measurement, $s \in S$ is the source sensor and v the measured value, which can be either nominal (categorical) or numeric (quantitative).*

Measurements can be produced by a sensor on a periodic base (e.g., temperature measurements) or whenever a particular event happens (e.g., door openings).

As many of the algorithms proposed in the literature borrow the terminology of data mining, the sensor log could be conceived as a sequence of events instead of a sequence of measurements. Hence we can introduce an alternative definition of sensor log as an event log:

Definition 1.2 (Event Log)**.** *Given a set $E = \{e_1, \ldots, e_{n_E}\}$ of* event types*, an* event sequence *is a sequence of pairs $\langle e, t \rangle$, where $e \in E$ and t is an integer, the occurrence time of the event type e.*

Definition 1.2 is more restrictive than Definition 1.1. Translating a sensor log into an event log could cause a loss of information especially if discretization of periodic sensor measurements is required (see Section 1.2.1).

Measured values are usually affected by a certain degree of uncertainty. Sensors have indeed their own technical limitations as they are prone to breakdowns, disconnections from the system and environmental noise (e.g., electromagnetic noise). As a consequence, measured values can be out of date, incomplete, imprecise, and contradictory with each other. Techniques for cleaning sensor data do exist [7], but uncertainty of sensor data may still lead to wrong conclusions about the current context, which in turn potentially lead to incorrect behaviors of final services.

We define "instrumenting a smart space" as the task of choosing and designing sensors in order to maximize the performance of AmI techniques employed in a specific smart space.

The correct design and deployment of smart spaces (from a practitioner's point of view), as well as the definition of novel architectures, techniques and methods (from the researcher's point of view) require extensive experimental facilities. Some freely-available datasets are published in the research community (cf. CASAS[2], MIT House_n[3] and Tracebase [170][4] projects); smart facilities are planned to be built in the nearby future (cf. EU initiatives for Living Labs[5], US NIST initiative for pervasive computing [176], etc.).

Unfortunately, available installations and testbeds only contain limited types and number of sensors that are not sufficient to validate algorithms for smart spaces

[2] http://wsucasas.wordpress.com/
[3] http://architecture.mit.edu/house_n/
[4] http://www.tracebase.org/
[5] http://www.openlivinglabs.eu/

in a thorough way; without customizable datasets available, it is hard to design a smart space, in such a way that maximizes employed algorithms performance. In order to face this issue, CASAS project has recently launched the "Smart Home in a Box" program [60][6], which is intended to employ crowdsourcing to gather a huge amount of sensor data from worldwide deployed installations.

Crowdsourcing can be applied also by means of serious games that allow people to perform their habits inside a virtual environment. Such a facility would allow to obtain models describing human habits. These models could be, in turn, employed to realize a configurable dataset generation tools able to generate realistic logs based on a virtual smart space consisting of agents that behave "as if" they were real inhabitants of the actual smart space. Such a virtual environment can be configured to contain a certain set of sensors and actuators, hence allowing to evaluate technique for AmI over a set of different smart space configurations. This approach will be explored in Chapter 3.

Despite of the miniaturization level that characterizes modern sensors, they can be perceived by users as a *privacy* threat [154]. Strategies to preserve privacy and security in smart environments are proposed in [153, 152]. The designer of a smart space has to take into account what kind of sensors users are going to accept as a means to acquire informations about their behavior. It is worth to note here how the perception of what is acceptable and what is not depends on the particular environment; as an example, cameras are generally accepted into public spaces (e.g., an airport) as they are used for security surveillance, but user perception strongly changes in private spaces (e.g., homes).

1.1.2 Actuators

Most common actuators in building automation are switches and dimmers, which are usually employed to control lights, and motors, which control blind/roller shutters, doors, windows and ventilation flaps.

As an AmI system is supposed to assist users in the widest range possible of daily routines, more complex devices need to be controlled. First available examples of such devices have been those belonging to the HVAC – Heating, Ventilation and Air Conditioning category. Nowadays many other devices, including many home appliances, provides interfaces that allow to use them without physical interaction. As an example, LG recently proposed Homechat; devices compliant to the Homechat standard can be controlled using a smartphone through natural language processing.

As our definition of smart space includes both a physical space and a cyberspace, *software services* on the Internet can be considered as an additional form of actuators following the CPS philosophy.

Example 1.1. *Typical examples of software services that can be used as actuators in smart spaces are e-commerce services. Suppose that an AmI system is able to detect what kind of products are missing in the fridge. At this point an e-commerce service can be employed, using the digital identity of one of the user, to buy missing products.*

[6] http://smarthome.ailab.wsu.edu/

1.2 Learning and Representing the Knowledge

Knowledge is represented in AmI systems using models. Models in the literature can be roughly divided into *specification-based* and *learning-based* [203]. Research in the field of AmI started when few kinds of sensors were available and the relationships between sensor data and underlying phenomena were easy to establish. Specification-based approaches represent hand-made *expert* knowledge in logic rules and apply reasoning engines to infer conclusions and to make decisions from sensor data. These techniques evolved in the last years in order to take into account uncertainty. Henricksen [99] proposes a full-fledged graphic/logic language Context Modeling Language (CML) suitable for both context extraction and decision making. A model contains static facts as well as dynamic facts derived by sensors. All these models work as an ontology expressed in predicate logic to be used by an AmI system. The growing availability of different kind of sensors made hand-made models impractical to be produced. In order to solve this problem, learning-based methods employ techniques from *machine learning* [34] and *data mining* [93].

Specification-based models are *human-readable* (even though a basic experience with formal logic languages is required), but creating them is very expensive in terms of human resources. Most learning-based models are instead represented using mathematical and statistical formalisms (e.g., Hidden Markov Models (HMMs), Hierarchical HMMs (HHMMs), Support Vector Machines (SVMs)), which make them difficult to be revised by experts and understood by final users.

Learning-based techniques can be divided into *supervised* and *unsupervised* techniques. The former expect the input to be previously labeled according to the required output function, hence they require a big effort for organizing input data in terms of training examples, even though active learning can be employed to ease this task. Unsupervised techniques (or weakly supervised ones) can be used to face this challenge but a limited number of works is available in the literature; a particular category of these techniques comes from the so called *temporal data mining* research field [84], which include *(i)* classification, *(ii)* clustering, *(iii)* pattern and rule discovery, and *(iv)* prediction techniques.

Unsupervised techniques for AmI knowledge modeling can be useful for other two reasons. In first place, as stated in the introduction, knowledge should not be considered as a static resource; instead it should be updated at runtime without a direct intervention of the users [167], hence updating techniques should rely on labeling of sensor data as little as possible. In second place, unsupervised techniques may also result useful in supporting *passive users*, such as guests, that do not participate in the configuration of the system but should benefit from its services as well.

In order to have the best of two worlds, human readable formalisms should be used in conjunction with unsupervised machine learning techniques; this is the major challenge in the AmI research field. We define “mining a smart space” as this challenge.

1.2.1 Handling Quantitative Sensor Data

Formalisms employed for representing knowledge in AmI systems often need interested environmental variables to be *binary* or *categorical*. A wide category of sensors (e.g., temperature sensors) produce instead numerical values, making it necessary to discretize sensor data before it can be used for reasoning.

Discretization methods in machine learning and data mining are usually classified according to the following dimensions [122]:

- *Supervised* vs *Unsupervised*. Unsupervised methods do not make use of class information in order to select cut-points. Classic unsupervised methods are equal-width and equal-frequency binning, and clustering. Supervised methods employs instead class labels in order to improve discretization results.
- *Static* vs *Dynamic*. Static discretization methods perform discretization, as a preprocessing step, prior to the execution of the learning/mining task. Dynamic methods instead carry out discretization on the fly.
- *Global* vs *Local*. Global methods, such as binning, are applied to the entire n-dimensional space. Local methods, as the C4.5 classifier, produce partitions that are applied to localized regions of the instance space. A local method is usually associated with a dynamic discretization method.
- *Top-down* vs *Bottom-up*. Top-down methods start with an empty list of cut-points (or split-points) and keep on adding new ones to the list by splitting intervals as the discretization progresses. Bottom-up methods start with the complete list of all the continuous values of the feature as cut-points and remove some of them by merging intervals as the discretization progresses.
- *Direct* vs *Incremental*. Direct methods directly divide the range of a quantitative attribute in k interval, where the parameter k is provided as input by the user. Conversely, incremental methods start from a simple discretization and improve it step by step in order to find the best value of k.

An additional dimension classifies distraction methods as either *univariate* or *multivariate*. Multivariate methods take into account the interactions between the discretized attribute and the other attributes in the dataset. Univariate methods work instead on each single attribute separately. This distinction is particularly interesting in the case of set and association rules mining.

If, on the one hand, supervised methods usually outperform unsupervised ones [79], on the other hand, in many application, presuming data to be labeled is simply unrealistic. Unfortunately, as pointed out in [185, 141], classical unsupervised methods for discretizing numerical attributes are not influenced from the overall distribution of the data into the database, hence, methods that preserve the semantics of the data are needed.

1.2.2 Managing Uncertainty

As sensor data contains errors reasoning techniques for AmI should take into account this factor.

The theory of *fuzzy sets* [205] is widely used to deal with uncertainty of vagueness, which is represented by a membership function which denote to what degree an element belongs to a fuzzy set. A probability value reflects the reliability of data, which can be provided from the domain expert, or obtained from the user experience. It allows imprecise knowledge to be modeled, such that an approximation of a numerical value, or a vague information is expressed. For example in the domain of temperature, fuzzy functions on temperature terms like "cold", "lukewarm", and "warm" can be mapped to a (possibly overlapping) range of temperature degrees.

Dempster-Shafer theory (DST) [184, 183] is a mathematical theory of evidence, which propagates uncertainty values and consequently provides an indication of the certainty of inferences. The core concepts of DST are the mass functions, the mass functions, the frame of discernment and combination rules. Mass functions distribute belief from sensors across choices or hypotheses in the frame of discernment. The combination rule is used to fuse belief from multiple sources.

As a generalized probability approach, DST has distinct features: it quantifies and preserves ignorance due to the lack of information and it aggregates beliefs when new evidence is accumulated. One of its important aspects is the combination of evidence obtained from multiple sources and the modeling of conflict between them. As for the model itself, the significant innovation of DST is that it allows for the allocation of a probability mass to sets or intervals. The process of using DST is described as follows. First of all, developers need to apply expert knowledge to construct an evidential network that describes how sensors lead to activities. Second, developers need to determine the evidence space and degree of belief for each evidence. Then the degrees of belief on sensor observations will be translated into the degrees of belief on the associated object context node by using the multi-valued mapping. The context can be lifted to higher-level context, while the mass function on the original context will be propagated to the higher-level context through an evidential mapping.

1.2.3 The Sensor Data Windowing Problem

Performing learning or mining from sequences of sensor measurements poses the issue of how to group events into aggregates of interests (i.e., actions, activities, situations). Even with supervised learning techniques, if labeling is provided at learning time, the same does not hold at runtime where a stream of events is fed into the AmI system. Even though most proposed approaches in the AmI literature (especially supervised learning ones) ignore this aspect, windowing mechanism are needed. As described in [114] different windowing methods can be classified into three main classes:

- **Explicit segmentation.** In this case the stream is divided into chunks usually following some kind of classifier previously instructed over a training data set. Unfortunately, as the training data set simply cannot cover all the possible combinations of sensor events, the performance of such a kind of approach usually results in single activities divided into multiple chunks and multiple activities merged.

- **Time based windowing.** This approach divides the entire sequence into

equal size time intervals. This is a good approach when dealing with data obtained from sources (e.g., sensors like accelerometers and gyroscopes) that operate continuously in time. As can be easily argued, the choice of the window size is fundamental especially in the case of sporadic sensors as a small window size could not contain enough information to be useful whereas a large window size could merge multiple activities when burst of sensors occur.

- **Sensor Event based windowing.** This last approach split the entire sequence into bins containing an equal number of sensor events. Usually a bins are in this case overlapping with each window containing the last event arrived together with the previous events. Whereas this method usually performs better than the other, it shows drawbacks similar to those introduced for time based windowing.

In order to apply in a meaningful way whichever approach among the previous ones, different issues have to be addressed. Take as an example the sensor event based windowing approach, which looks to be the most suitable for discontinuous event sources. This approach has, as a main drawback, the possibility that events in the window belong to different activities either because they are widely spread apart in time or because they represent the quick passage from one activity to another.

Authors in [114] propose different enhancements to the basic sensor event based windowing schema:

- A first improvement consists of exponentially reducing the contribution of an event following its distance in time from the reference event (i.e., the last event) for the current window.
- A second improvement consists of weighting the contribution of a sensor event based on how much two consecutive events are *historically* correlated. The mutual information among events is calculated over a training dataset.
- Another improvement consists of adding to the current window some information about the past windows.
- A final, more complex, improvement consists of implementing a dynamic window size mechanism based on some labeled training data.

Those AmI techniques that stem from data mining methods for *market basket analysis* (e.g., the Apriori algorithm [9]) apply a windowing mechanism in order to transform the event/sensor log into what is called a *database of transactions*. Market basket analysis is a special case of *affinity analysis* that discovers co-occurrence relationships among purchased items inside a single or more transactions. Hence, it is worth here to report some of the definitions related to this topic as they will be useful during the rest of the thesis.

Definition 1.3 (Database of Transactions)**.** *Let* $I = \{i_1, \ldots, i_{n_E}\}$ *be a set of binary variables corresponding to sensor event types. A* transaction *is an assignment that binds a value to each of the variables in* V*, where the values* 0 *and* 1 *respectively denote the fact that a certain event happened or not during the considered window. A* database of transactions $\mathcal{T}$ *is an (usually ordered) sequence of transactions each having a, possibly empty, set of properties (e.g., a timestamp).*

Strictly connected to Definition 1.3 are those of item, itemset and support.

Definition 1.4 (Item, Itemset and Support)**.** *An* item *is an assignment of the kind* $i_k = \{0, 1\}$*. An* itemset *is an assignment covering a proper subset of the variables in* I*. An itemset* C *has support* $Supp_{\mathcal{T}}(C)$ *in the database of transactions* $\mathcal{T}$ *if a fraction of* $Supp_{\mathcal{T}}(C)$ *of transactions in the database contain* C*.*

The kind of AmI techniques following this strategy turn the input log into a database of transactions each of them corresponding to a window.

Other AmI techniques employ windows obtained from the input log in more complex ways, e.g., analyzing ordering relations between events.

1.3 Decision Making

The way an AmI system make decisions on the actions can be compared to decision making in AI agents. For example, *reflex agents with state*, as introduced in [178], take as input the current state of the world and a set of Condition-Action rules to choose the action to be performed.

Similarly, Augusto [19] introduces the concept of Active DataBase (ADB) composed by Event-Condition-Action (ECA). An ECA rule basically has the form "ON *event* IF *condition* THEN *action*", where conditions can take into account time. This work has been extended in [18] with more complex temporal operators (`ANDlater` and `ANDsim`) and adding uncertainty management. An example of ECA rule is the following:

```
ON occurs(Shower, Off, t0)
IF context(BathroomHumidityLevel (>, 75))
THEN do(On, BathroomFan, t) when t = t0 + 10s
```

The APUBS [20] system extends previous works by mining ECA rules instead of relying on a specification-based approach. APUBS makes clear the difference between different categories of sensors. Sensors are divided into three classes defined as follows:

- **Type O** sensors installed in objects, thus providing direct information about the actions of the users.
- **Type C** sensors providing information about the environment (e.g., temperature, day of the week).
- **Type M** sensors providing information about the position of the user inside the house (e.g., in the bedroom, near the sink).

Events in the *event* part of the ECA rule always come from sets O and M. Conditions are usually expressed in terms of the values provided by Type C sensors. Finally, the *action* part contains only Type O sensors.

The set of Type O sensor is called *mainSeT*. The first step of the APUBS method consists of discovering, for each sensor in the *mainSeT*, the set *associatedSeT* of potential O and M sensors that can be potentially related to it as triggering events. The method employed is APriori for association rules [9]; the only difference is that

possible association rules $X \Rightarrow Y$ are limited to those where cardinality of both X and Y is unitary and Y only contains events contained in *mainSeT*. Obviously this step requires a window size value to be specified in order to create transactions.

As a second step, the technique discovers the temporal relationships between the events in *associatedSeT* and those in *mainSeT*. During this step, not significant relations are pruned.

As a third step the conditions for the ECA rules are extracted by applying a JRip classifier [200].

Another way of reactively responding to changes in the environment is solving a CSP – Constraint Satisfaction Problem. The system introduced in [69, 148] takes as input a set of rules expressed as formulas in a predicate logic. Rules can be either *necessary* or *desirable*. At any given time the system ensures that these rules are satisfied by modifying *controllable environmental variables* that correspond to actuators.

Knowing about the current state of the environment is not always enough to decide what to do. Sometimes, as well as the current state, AI agents need some sort of goal information; these agents are called *goal-based agents* [178].

Such an approach has been followed for AmI by the SM4All architecture [75]; here, knowledge takes the form of *service models* consisting of transition systems, which are behavioral representations consisting in states connected by transitions. Transitions between states are triggered through the invocation of actions specified in terms of preconditions and postconditions expressed in terms of logic formulas over environmental state variables. On the basis of the service descriptions, the *composition engines* of SM4All are in charge of providing complex services by suitably composing the available ones aiming at specific goals directly requested by the user:

- **Off-line synthesis** composition engine. In the off-line mode, at design/deployment time of the house, a desiderata (i.e., not really existing) target service is defined, as a kind of complex routine, and the synthesis engine synthesizes a suitable orchestration of the available services realizing the target one. Such an orchestration specification is used at execution-time (i.e., when the user chooses to invoke the composite/desiderata service) by the Orchestration Engine in order to coordinate the available services (i.e., to interact with the user on one hand and to schedule service invocations on the other hand). In this approach, the orchestration specification is synthesized off-line (i.e., not triggered by user requests, at run time) and executed on-line as if it were a real service of the home. The off-line mode is based on the so called Roman Model [67]; goals are in this case target transition systems, which the engine must realize by simulation, on top of the set of available services. Pre- and post-conditions are expressed as constraints over the TS transitions, on top of environmental variables. Once the orchestration is computed, the target itself is stored into the Service Repository: it can be invoked at any time in the future, like any other service.

- **On-line planning** composition engine. The user, during its interaction with the home, may decide not to invoke a specific service (either available/real or composite), but rather to ask the home to realize a *goal*; in such a case, the engine, on the basis of specific planning techniques [106], synthesizes and

executes available service invocations in order to reach such a goal. Goals are expressed as logic formulas over environmental state variables, which the user expects to become true due to the enactment of the synthesized plan. The reasoning core is a planner that solves the underlying planning problem through the reduction of it into a CSP.

The output of the two composition engines are different. In the case of off-line planning a new transition system is obtained, which respect the constraints imposed by the original transition systems. The on-line planning engine returns instead a plan containing actions belonging to the original services without ensuring that precedence relationships will be respected.

Another way of approaching the problem of decision making is to taking advantage of workflow specifications to anticipate user actions. A workflow is composed by a set of tasks related by qualitative and/or quantitative time relationships. Authors in [85] presents a brief survey of techniques for *temporal calculus* (namely Allen's Temporal Logic and Point Algebra) and *spatial calculus* mainly aiming at decision making.

The SPUBS system [22, 23] automatically retrieve these workflows from sensor data by merging concepts from both data mining and workflow mining [3]. The methodology basically consists of the following steps:

1. **Identifying Frequent Sets.** During this step, thoroughly inspected in [22], frequent item-sets are mined by employing the seminal APriori algorithm [9]. Transaction database is obtained in this case by segmenting the dataset on a per time-slot basis. Frequent item-sets are than refined in order to include frequent events that have been discarded by APriori.

2. **Identifying Topology.** This step has been deeply influenced by previous works on Workflow Mining. Once the a frequent item-set has been obtained during the previous step, a weighted directed graph is constructed where weights are computed as the number of times vertices of the corresponding edges come one after the other in the sensor log. As a first refinement, events that happens multiple time are replicated in such a way to reflect the context when they happen. Then events without any ordering relation are merged into single events.

3. **Identifying Time Relations.** At this step, for each edge in the graph created at the previous step, a either quantitative or qualitative time relation is mined.

4. **Identifying Conditions.** During the last step, conditions are mined for those edges that happens only if certain conditions of the environment (e.g., temperature, day of the week) are satisfied.

The main disadvantage of the SPUBS approach consists of the necessity for the sensor log to be already segmented before mining.

1.4 Context Extraction

Models employed for decision making should not concern themselves with the individual pieces of sensor data (which room the user is in, what his heart rate

or blood pressure is): rather, this information should be interpreted into a higher, domain-relevant concept, such as whether the user is suffering a heart attack or exercising. Thus between sensing and decision making we put an additional layer called *context extraction*. Some of the works presented in this section face both the challenges of context extraction and decision making but, differently from the techniques introduced in Section 1.3, here the focus is more on the first aspect.

In order to define the concept of context as a layered comprehension of the environment (see [50]), we briefly introduce the concepts of *context* and *situation* as defined in [203, 78].

Sensor data is aggregated into *context objects* (COs), which symbolically represent the environment in which the system operates. A CO expresses what can be said about one or more *entities*. A first dimension to classify CO is that dividing *intrinsic COs*, which involve a single related entity, and *relational COs*, which involve multiple related entities.

Example 1.2. *"John is in the Kitchen" is an example of intrinsic CO where John is an entity of type* `person` *and the Kitchen is a the value of the context* `location`*. "John is close to both the Phone and the Television" is an example of relational context where Phone and Television are the value of the context* `closeto`*.*

COs can be further sub-divided into those derived directly from sensors (*primary COs*) and those inferred and/or derived from several sensor streams using *multimodal fusion* (*secondary COs*). An important type of secondary COs is represented by *derived events* representing small, higher-level inferences about contextual information, such as the action of "chopping" derived by observing motion over time (a survey of vision-based human action recognition algorithm is given in [163]). As well as primary COs inherit sensor uncertainty, secondary COs may introduce an additional uncertainty level due the inference task itself.

The concept of situation is well defined in [32] as a "semantic abstraction from low-level contextual cues". A situation abstracts the events occurring in the real world derived from COs and hypotheses about how observed context relates to factors of interest for designers and applications. Situations typically fuse several CO sources, as well as models of the expected behavior of the phenomena being observed. A situation may correspond to an activity (intended as a task that involves the execution of many steps from one or many users), or expressing a more complex concept about what is happening in the monitored environment.

Example 1.3. *"A meeting is happening in Room A" is an example of activity that describes a situation. A situation can even depict very generic concepts such as "The user needs assistance".*

Another way of looking at context extraction is to consider every specific inference task as a *virtual sensor* that triggers whenever an action or a situation is detected.

1.4.1 Specification-based methods

First approaches to the development of context-aware systems able to recognize situations where based on *predicate logic*. Loke [126] introduced a PROLOG extension called LogicCAP; here the "in-situation" operator captures a common form of reasoning in context-aware applications, which is to ask if an entity E is in a given

situation S (denoted as $S^*{>}E$). In particular a situation is defined as a set of constraints imposed on output or readings that can be returned by sensors, i.e. if S is the current situation, we expect the sensors to return values satisfying some constraints associated with S. LogicCAP rules use backward chaining like PROLOG but then utilizes forward chaining in determining situations, i.e., a mix of backward and forward chaining is used in evaluating LogicCAP programs. The work introduce different reasoning techniques with situations including selecting the best action to perform in a certain situation, understand what situation a certain entity is in (or the most likely) and defining relationships between situations. The same author [127] later defined a formal methodology to compose different context-aware pervasive systems. The following script represents a situation program that follows this approach:

```
if in_meeting_now(E) then
   with_someone_now(E) ,
   has_entry_for_meeting_in_diary(E) .
if with_someone_now(E) then
   location*(E, L) , people_in_room*(L, N) , N > 1.
if has_entry_for_meeting_in_diary(E) then
   current_time*(T1) ,
   diary*(E, 'meeting', entry(StartTime, Duration)) ,
   within_interval(T1, StartTime, Duration) .
```

In the above situation program, a situation `in_meeting_now` of a user entity E is defined on two situations `with_someone_now` and `has_entry_for_meeting_in_diary`. Each of these situations has its own program that is defined on sensor predicates; for example, `with_someone_now` refers to two sensor predicates: `location*(E, L)` that returns the location of the entity, and `people_in_room*(L, N)` that returns the number of people in the location. In this way, situation programs are made amenable to formal analysis, and the inference procedure of reasoning about situations is decoupled from the acquisition procedure of sensor readings.

Ontologies have increasingly gained attention as a generic, formal and explicit way to "capture and specify the domain knowledge with its intrinsic semantics through consensual terminology and formal axioms and constraints" [202]. They provide a formal way to represent sensor data, context, and situations into well-structured terminology, which makes them understandable, shareable, and reusable by both humans and machines. A considerable amount of knowledge engineering effort is expected in constructing the knowledge base, while the inference is well supported by mature algorithms and rule engines. Up to 2007 [202], most relevant systems based on ontologies were CoBrA [49] (based on OWL), Gaia [166] (based on DAML + OIL) and SOCAM [88] (also called CONON, based on OWL). The following example shows how SOCAM define the situation of 'sleeping':

```
(?user rdf:type socam:Person),
(?user, socam:locatedIn, socam:Bedroom),
(?user, socam:hasPosture, 'LIEDOWN'),
(socam:Bedroom, socam:lightLevel, 'LOW'),
```

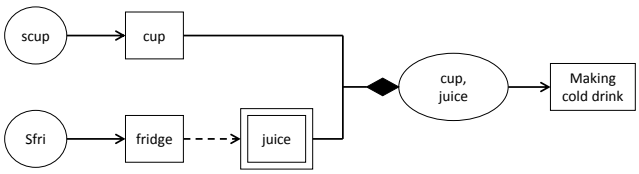


Figure 1.2. An example of DST evidential network.

```
(socam:Bedroom, socam:doorStatus, 'CLOSED')
-> (?user socam:status 'SLEEPING')
```

Ontologies have been also used in the analysis of mainly visual scenes [146]. Fundamental features of ontologies to be used in a pervasive systems are analyzed in [186].

Another example of using ontologies in situation identification is given by [171]. Instead of using ontologies to infer activities, they use the ontologies to validate the result inferred from statistical techniques as those introduced in Section 1.4.2.

As the temporal dimension is important in human activities and physic phenomena, temporal logics have been often employed for context extraction. Authors in [103] employ temporal logic to infer situation from different kind of secondary contexts obtained through computer vision. In [132] Real/time Past Linear Temporal Logic (R-PTL), derived from Linear Temporal Logic (LTL) [162], is employed for situation identification through the use of model checking [56].

Managing uncertainty and evidences

Fuzzy Logic theory has been employed for example in [14]. Here authors apply fuzzy inference on a conceptual hierarchy of situational context ontologies. The inference process is to find a situation that is most similar to the current unknown situation by evaluating the similarity of the specifications of situations in the knowledge base and the current context input. The aim is to calculate the *situational involvement*, that is the degree of belief that a user is involved in a recognized/estimated situation, applying a fuzzy function.

Fuzzy rules are also employed, by [54]. In this framework, authors propose a robust and general rule-based approach to manage situation awareness. Semantic Web reasoning, fuzzy logic modeling, and genetic algorithms are used to handle, respectively, situational/contextual inference, uncertain input processing, and adaptation to the user's behavior.

Figure 1.2 depicts an example DST evidential network taken from [100]. Here we have two sensors `sfri` and `scup`, which are respectively connected to the context objects `fridge` and `cup`. As soon as the fridge sensor is triggered, the state of the fridge context is changed to indicate that the user is interacting with the fridge (opening the fridge and getting food out of the fridge). However, it is not possible to infer what food is removed from the fridge by simply considering the current state of the fridge door. If the inhabitant wants to make a cold drink, it is more likely that the juice is removed. Then we have a composite object, `(cap,juice)`, which has two compulsory context objects namely `cup` and `juice`. This composite object

is in turn compulsory for `makingColdDrink` situation. Using DST theory (see [100] for details) it is possible to estimate the belief of a certain situation depending on the available context objects.

An additional approach is represented by *probabilistic description logic*. For example [98] exploits Log-linear DLs to model both probabilistic and deterministic dependencies between DL axioms through extending uncertain axioms with weights. Regarding the expressiveness of the language, Log-linear DL supports the same operators of the OWL 2 language.

1.4.2 Supervised Learning-based methods

Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. A vast collection of works employ supervised learning techniques in order to learn models of situations, activity and simple actions.

Bayesian Derivatives

Bayesian classification techniques are based on the well known Bayes theorem $P(H|X) = \frac{P(X|H)P(H)}{P(X)}$, where H denotes the hypothesis (e.g., a certain activity is happening) and X represents the set of evidences (i.e., the current value of context objects). As calculating $P(X|H)$ can be very expensive, different assumptions can be made to simplify the computation. For example, *naïve Bayes* (NB) is a simple classification model, which supposes the n single evidences composing X independent (that the occurrence of one does not affect the probability of the others) given the situational hypothesis; this assumption can be formalized as $P(X|H) = \prod_{k=1}^{n} P(x_k|H)$. The inference process within the naïve Bayes assumption chooses the situation with the maximum a posteriori (MAP) probability.

If dependencies between some attributes are known, Bayesian (belief) networks (BN) can be applied to replace naïve Bayes, whose performance degrades in this condition. A Bayesian network is a directed acyclic graph where each node represents a variable that can be discrete or continuous, and each arc is the causal relationship between nodes. If there is an arc from a node A to another node B, then A is called a parent of B, implying that the variable B is regarded as depending directly on A. In this sense, a naïve Bayes model can be viewed as a Bayesian network where each input attribute has an output classification as the sole parent and the classification has no parents.

Bayesian networks have been employed to add uncertainty management to ontology based situation reasoner. This approach has been for example employed in [87] to the already cited SOCAM system. Here, each context predicate present in the ontology is mapped onto a node of the BN and an arc is drawn between two nodes if a dependency relation exists (the dependency between two context predicates is hand-made and represented using an ontology property). After the BN is created, it is trained on real data to get probability distributions of the various nodes.

Bayesian networks provide a clear and well understood method for incorporating how the likelihood of a possibility of an event is conditioned on another event.

They are best suited to applications where there is no need to represent ignorance, where conditioning is easy to extract through probabilistic representation and prior probabilities are available.

Hidden Markov Models (HMMs) are a statistical model where a system being modeled is assumed to be a Markov chain that is a sequence of events. A HMM is composed of a finite set of hidden states (e.g., s_{t-1}, s_t, and s_{t+1}) and observations (e.g., o_{t-1}, o_t, and o_{t+1}) that are generated from states. HMM is built on three assumptions: *(i)* each state depends only on its immediate predecessor, *(ii)* each observation variable only depends on the current state, and *(iii)* observations are independent from each other. In a HMM, there are three types of probability distributions: *(i)* prior probabilities over initial state $p(s_0)$; *(ii)* state transition probabilities $p(s_t|s_{t-1})$; and *(iii)* observation emission probabilities $p(o_t|s_t)$. The training of a HMM is performed using the Baum-Welch algorithm while situation identification is performed using the Viterbi algorithm. HMMs for activity recognition using accelerometer sensor data are employed in [195].

A problem with the use of a standard HMM for activity detection is that due to the first order Markov assumption (a state depending only on the previous one) the probability of an event state being observed for a certain interval of time declines exponentially with the length of the interval. Also the probability that there is a change in the hidden state does not depend on the amount of time that has elapsed since entry into the current state, which could be an important parameter in modeling human activities. To model the prior duration of an event state, HMM can be extended to *semi-HMM*, where the hidden process is semi-Markovian. This semi-HMM performs better at approximating visual primary and composite activities.

Another drawback of using a standard HMM is its lack of hierarchical modeling for representing human activities. To deal with this issue, several other HMM alternatives have been proposed, such as *hierarchical* and *abstract* HMMs. In a hierarchical HMM each of the hidden states can be considered as an autonomous probabilistic model on its own; that is, each hidden state is also a hierarchical HMM. In [147] for example, the performance of this technique is compared to that of a flat HMM. In this work every HMM contains a special 'end' state which returns the control to the parent HMM.

In an abstract HMMs the bottom part of this model consists of hidden states and observations as in a typical HMM, but the states are linked to abstract policy (or activity) variables which are placed in a hierarchy. Flag variables are used to indicate whether the current policy or activity continues or terminates in the next time step [42].

HMMs generally assume that all observations are independent, which could possibly miss long-term trends and complex relationships. *Conditional Random Fields* - CRFs, on the other hand, eliminate the independence assumptions by modeling the conditional probability of a particular sequence of hypothesis, Y , given a sequence of observations, X; succinctly, CRFs model $P(Y|X)$. Modeling the conditional probability of the label sequence rather than the joint probability of both the labels and observations $P(X, Y)$, as done by HMMs, allows CRFs to incorporate complex features of the observation sequence X without violating the independence assumptions of the model. The graphical model representations of a

HMM (a directed graph) and a CRF (an undirected graph) makes this difference explicit. For example in [192] a comparison between HMM and CRF is shown, where CRF outperforms HMM in terms of timeslice accuracy, while HMM outperforms CRF in terms of class accuracy.

Authors in [59] compare the performance of NB, HMM and CRF on different datasets taken from the CASAS project.

Decision Trees, Neural Networks and Support Vector Machines

A decision tree (DT) is a predictive model where each leaf represents a classification and each branch represents a conjunction of features that lead to the target classifications. A decision tree is built on information entropy in that its construction works by choosing a variable at each step that is the next best variable to use in splitting the set of items. Compared to Bayesian models, a decision tree does not take into account the dependence assumption or potential temporal sequence between classifications. In [27] a set of features is extracted from the data acquired from a set of wearable sensors (accelerometers) in fixed size time slots. Different machine learning techniques are compared to DTs which however results in the best performance.

Support Vector Machines (SVM) allow to classify both linear and nonlinear data. An SVM uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane that separates the training data of one class from another. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated. SVMs are good at handling large feature spaces since they employ overfitting protection, which does not necessarily depend on the number of features. SVMs are employed for example in [114] for activity recognition. An interesting approach is presented in [38] where a mixed supervised/unsupervised learning method is employed. First SVM (supervised) are employed to learn roles, then an unsupervised technique (Jeffrey divergence) is employed to extract situation models, and last SVM are employed again to improve discovered models.

Artificial neural networks (ANNs) are a sub-symbolic technique, originally inspired by biological neuron networks. They can automatically learn complex mappings and extract a non-linear combination of features. A neural network is composed of many artificial neurons that are linked together according to a specific network architecture. A neural classifier consists of an input layer, a hidden layer, and an output layer. Mappings between input and output features are represented in the composition of activation functions f at a hidden layer, which can be learned through a training process performed using gradient descent optimization method or resilient backprogagation algorithms. In [201] the current situation is recognized by employing ANNs learned over accelerometer sensor data.

Emerging Patterns

Given Definition 1.3 and Definition 1.4, and given to different databases of transactions T_1 and T_2, the *growth rate* of an itemset C from T_1 to T_2 is defined as $\frac{Supp_{T_2}(X)}{Supp_{T_2}(X)}$. *Emerging patterns* are those itemsets showing a growth rate greater than a certain

threshold ρ. The ratio behind this definition is that an itemset that has high support in its target class (database) and low support in the contrasting class can be seen as a strong signal of in order to discover the class of a test instance containing it.

Gu et al. propose an emerging patterns based approach to recognize sequential, interleaved and concurrent activities for the single user case [89] and the multiple user case [90]. The training dataset is made up by a sensor log where each event is labeled with the corresponding activity of daily life (ADL). A distinguishing feature of this work is that it takes into account numerical sensors. In particular, the numeric measurements coming from sensors are discretized on a per-second basis.

The input sensor log is translated into an event log. The emerging patterns for a specific ADL are obtained by putting in the first database only events belonging to other ADLs whereas the second database contains only instances of the given ADL.

1.4.3 Unsupervised Learning-based methods

As obtaining labeled data is expensive in terms of human resources a big research effort has been put in devising weakly supervised learning and unsupervised learning techniques. In weakly supervised learning only a small part of the dataset is annotated. [187] makes use of Multi-Instance SVMs (a weakly supervised learning techniques based on SVMs), which is reported to be robust to labeling noise and can achieve competitive classification using only few labels.

Unsupervised learning techniques employed for computer recognition of activities and contexts are mainly based on data mining techniques that do not need the input data to be labeled.

Ontology Mining

Vast majority of ontology-based methods are specification-based. An interesting approach [151], based on *web mining* techniques, extracts key terms for each targeted activity from a collection of online "how to" documents, and determines the relevance of each term. The assumption underlying this approach is that in most activities of daily living, the lists of relevant objects for a particular activity are similar and do not vary significantly even when the activity is performed in different ways. An advantage of the proposed technique is that there is no need for the training data collection phase to be performed using humans acting in the smart environment.

Authors in [113] propose a technique to mine a taxonomy (i.e., an ontology) of human activities from a set labeled datasets (the labels do not refer to the taxonomy itself). Such a taxonomy is useful to have an insight of the hierarchical relationships among activities and can be used to have a common vocabulary for the AmI field.

Pattern Mining

Approaches related to pattern analysis in data mining take as input an event log and extract patterns of events a transaction database obtained by windowing.

Authors in [133] mine episodes, which are patterns characterized by a partial order among events. GAIS genetic algorithm [177] detects interleaved event patterns in a unsupervised learning fashion.

CASAS project employs a pattern mining technique [168, 61] in order to discover in an unsupervised manner human activity patterns whereas several classical methods for activity recognition (e.g., HMM, BN, SVN) are employed for recognizing them at runtime. The proposed method allows to discover *discontinuous patterns and variations* by iteratively trying to compress the dataset. A pattern is basically defined as a set of events where order is not specified and events can be repeated multiple times. Patterns are mined by iteratively compress the sensor log. The data mining method employed for activity discovery is completely unsupervised without the need of manually segmenting the dataset or choosing window sizes.

The main goal of the algorithm proposed in [61] is to improve the performance of activity recognition algorithms by trying to reduce the part of the dataset that has not been labeled during data acquisition. Thus, evaluation is performed following this research direction. Experiments in [168] focus instead on analyzing how much the system is able to mine pattern corresponding to the activities (known) repeated multiple times into the environment by participants. As the method is able to mine discontinuous patterns, tests have been performed over both sequential and interwoven ADLs.

A set of works in pattern mining from event sequences took inspiration from the Vector Space Model (VSM) used in information retrieval. In the simplest case for each entity of interest, topic models [102] are mined where each model is essentially a *bag of events*. As an improvement *bag of n-grams* [92] can be mined. A performance improvement to the bag of n-grams model is represented by *suffix trees* [91].

A suffix tree enumerates all unique subsequences of events in a certain activity, where any subsequence of events from the root node to a non-root node is called a event motif. [92] proposes a framework to discover temporal patterns of finer-grained events in everyday human activities. Here an activity is defined as a finite sequence of events, while an event is defined as a particular interaction among a subset of key objects over a finite duration of time. A key object is an object present in an environment that provides functionalities that may be required for the execution of an activity of interest. The idea is to extract events from interactions between human users and key objects, organize events into a Suffix-Tree, and mine the patterns of events in each activity.

Association Rules

Authors in [129] applies (point-wise) intertransaction association rules (IARs) [120] mining in order to detect *emerging* (unusual) behaviors. IARs extend the concept of association rule defined in [9] to include relationships that span more then one transaction. The basic step consists of mining IAR rules from sensor logs, whereas emerging ones are identified, as a second step, by comparison. The paper introduces the Extended Frequent Pattern Tree (EFP-Tree) algorithm, which outperforms both the previously proposed EH-Apriori and FITI (First Intra Then Inter) algorithms.

The real world data consisted of event logs from an array of state-change sensors. These events are transformed into a database of transaction by inserting in each transaction the events gathered in windows of fixed size (five minutes intervals). The effectiveness of the algorithm with respect to emergent behaviors has been assessed only from a qualitative point of view, by performing a visual inspection.

1.5 Software Platforms and Middleware

Many research projects proposed reference architectures for smart spaces. For example, the SM4All project [75] focus on making services provided by the different components in the smart space available to different categories of users (including those with disabilities) using the suitable interface.

universAAL [189] is a recently proposed reference architecture, similar to that proposed by other projects such as SOPRANO, and a software framework[7] for AmI systems. The middleware here is in charge of connecting together the *nodes* composing a smart space (here called AAL space); it defines three specific *buses*, namely: the *context bus*, the *service bus*, and the *user interaction bus*.

The context bus transports *context events*, which are modeled as triples with subject, predicate and object (e.g., user is in the kitchen); they are published by *context publishers* and consumed by *context subscribers*, according to the *push* paradigm. The service bus allows *service callers* to ask for services (e.g., switch on a light) provided by a set of *service callees*, according to the *pull* paradigm. The semantics of both context events and services is defined using an *ontology*. Finally, the user interaction bus is in charge of transporting user interaction *forms*, which are generated by *interaction callers*, to the nodes (e.g., a Web server), called *interaction handlers*, which in turn are able to render the forms for the physical interaction with users.

Applications connect to a single bus or to multiple ones to perform their tasks. Some low level applications, called *managers*, are part of the universAAL framework as infrastructural components, and provide basic services, e.g., the *situation reasoning* or the *context history*.

The universAAL middleware is representative of the features that a middleware for AmI should provide [169]. DPWS[8] [150] is specifically designed to enable devices (sometimes resource-constrained) to expose their services to other software components. DPWS is similar to UPnP – Universal Plug and Play [140] and the universAAL [189] middleware in terms of final goals, being in addition fully compliant with Web Service specifications. A DPWS-compliant device is able to notify its presence to the system as soon as it is connected and can provide its services via both the remote procedure call (blocking) and the publish/subscribe (non-blocking) models.

Recently Microsoft started the HomeOS initiative [77] aimed at bringing middleware for AmI to the level of a real operating system for smart spaces.

[7] http://universaal.org/
[8] http://ws4d.e-technik.uni-rostock.de/

Chapter 2

Inhabitants in the Viewfinder

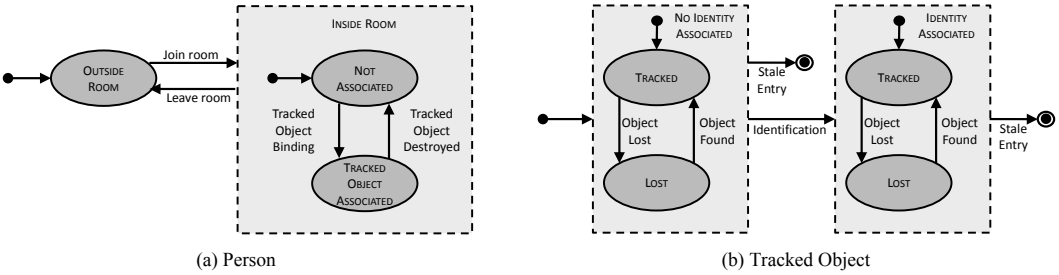
People moving in outdoor environments are used to relying on computer-based services that make specific daily activities easier. In particular, localization services are daily used by millions of persons in the form of Global Navigation Satellite Systems (GNSSs); while traditional GNSS-based applications, such as car navigation, are continuously being improved by integrating GISs (Geographical Information Systems) and real-time traffic-driven routing algorithms, new applications are arising ranging from augmented reality to social-network based recommender services.

On the other side, the time spent indoors is considerable.

Current smart spaces require more and more sophisticated sensors able to acquire the state of the environment in order to provide advanced and customized services. Among the most important environmental variables, locations of users and their identities represent a primary concern for smart home applications. Despite some years of investigation in indoor positioning, the availability of systems designed as components pluggable into complex home automation platforms is limited. In this chapter, we present PLaTHEA [118] – People Localization and Tracking for HomE Automation, a vision-based indoor localization system specifically tailored for Ambient Intelligence (AmI) systems. PLaTHEA features a novel technique to acquire a stereo video stream from a couple of independent (and not synchronized) network-attached cameras, thus easing its physical deployment. The input stream is processed by integrating well-known techniques with a novel tracking approach targeted to indoor spaces. The system has a modular architecture that offers clear interfaces exposed as Web services, and it can run on off-the-shelf and cheap hardware (both in terms of sensing devices and computing units). We evaluated PLaTHEA in real usage conditions and we reported the measured performance in terms of precision and accuracy. Low light, crowded and large monitored environments might slightly decrease the performance of the system, nevertheless the results here presented show that it is perfectly suitable to be employed in the typical domestic day-to-day life settings.

2.1 Preliminaries

An indoor localization system is in charge of providing information about dynamic positions (e.g., trajectories) and, optionally, postures and identities of persons (and,



(a) Person (b) Tracked Object

Figure 2.1. State transition diagrams for persons and tracked objects.

more generally, mobile agents) detected in a smart indoor environment.

A *tracked object* is a limited-life artifact that models the knowledge about a real agent actually followed by the localization system. An agent will be associated, during its unlimited (with respect to the system) life, to several tracked objects in sequence for a limited period of time each; for example, when a person associated to a certain tracked object leaves the monitored room and then it comes back afterwards, it will be probably associated to a new tracked object.

At a specific time step t with $t \geq 0$, the system is aware of a set of tracked objects $\mathcal{T}_t = \{o_1^t, \ldots, o_{n_t}^t\}$, with $\mathcal{T}_0 = \emptyset$, where n_t is the cardinality of the set. For each o_j^t with $j = 1 \ldots n_t$, a state $\mathcal{S}\left(o_j^t\right)$ is stored, consisting of the current estimated position and, optionally, *identity* (see Section 2.1.3) and/or *postures* (see Section 2.1.4). $\mathcal{S}\left(o_j^t\right)$ additionally includes a set of features that are necessary to track the agent (time) step by step; the actual features included depend on the particular localization system.

At time step $t+1$, the localization system detects in the scene a set of agents $\mathcal{D}_{t+1} = \left\{d_1^{t+1}, \ldots d_{m_{t+1}}^{t+1}\right\}$ where m_{t+1} is the cardinality of the set. If the set of features extracted from a detected agent d_a^{t+1} match those stored in the state of a certain tracked agent o_b^t, the state of the latter is updated with the information extracted from the former. At any time step, the subset of tracked objects for which an association cannot be found is considered temporarily or definitely lost depending on the expired time. Those detected agents that are not similar to any tracked object correspond to persons who just entered the room, for which new tracked objects have to be created.

In order to further inspect this aspect, the reader should consider the different life cycles of real agents and tracked objects. At any given time, an agent can be inside or outside the monitored room (see Figure 2.1a). When it is inside, it can be either associated or not to a tracked object; the transitions between these two states are driven by binding events and tracked object destruction events. A tracked object (see Figure 2.1b) is created at a certain time using as template a detected object that cannot be associated to any already tracked object (binding event). In case the system supports identity recognition, at a certain time, a specific identity could be associated to it. In both these macro states (namely identified and not-identified), the object could get lost; from this point it could return back to the tracked state if the system was able to recover its position or it could get stale and then destroyed.

In the last years, different solutions for indoor localization have been proposed

in both the academic and commercial fields. The following sections classifies the major approaches according to several dimensions. A direct comparison between performance achieved by specific systems will be presented later in Section 2.6.

2.1.1 Marker-less vs Marker-based Systems

A first classification dimension splits localization systems between those requiring mobile agents to wear special markers or tags (*marker-based systems*) and those supposing mobile agents to passively participate in the ongoing localization task, i.e., not requiring any body area tag (*marker-less systems*). As of this definition, the terms *active* and *passive* are respectively used to denote the two kinds of localization systems. While passive location systems are easily recognizable as less invasive from the point of view of humans, their complexity is usually higher relative to active systems.

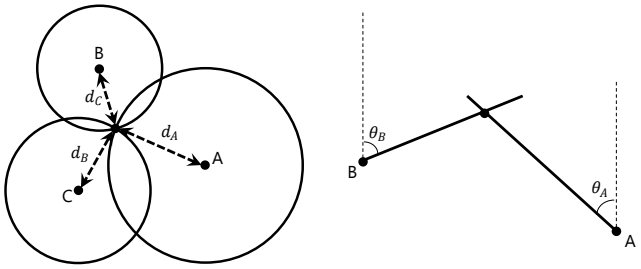
In active systems each tag usually has a unique ID. So, provided that the system is able to calculate step by step the position of a tag, it is possible to track a mobile agent by simply grouping measurements from a certain tag ID, which represents the only feature in $\mathcal{S}\left(o_j^t\right)$. These tag IDs can also be used to directly deal with identity recognition. In passive systems, the association between agents detected at time t and those already tracked at time $t-1$ is more complex, as it implies the extraction and comparison of a set of features dependent on the particular system.

The extraction of these features, in passive systems, comes after *foreground extraction*, which basically involves separating the so called *foreground*, composed by mobile agents to monitor, from the rest of the environment, namely the *background*. This task is not trivial due to the fact that background is subject to long term changes (e.g., furnitures moved) and short term changes (e.g., tree leaves, curtains). Depending on the adopted technology, additional issues, e.g., transient electromagnetic noise in the case of radio-based systems, can make this task even more complex. Additionally, foreground extraction is followed by a *segmentation* phase, which involves discerning mobile agents into the foreground, that may become more difficult in *crowded* environments. A review of techniques used for background modeling and foreground extraction with different sensing technology can be found in [64].

Whereas active systems are generally simpler, in terms of implementation, than passive ones, they present a set of limitations. One example is the difficulty in obtaining person postures (see Section 2.1.4). More remarkably, active systems can be used only if the system has to deal with collaborative agents that decide to become traceable. In some application scenarios (e.g., security and traffic analysis), this is not a reasonable assumption.

2.1.2 Sensing Technology

A second dimension classifies localization systems by considering the technology employed to *sense* the environment. As we will see, this choice heavily influences other features of the system.



(a) Lateration (b) Angulation

Figure 2.2. Geometric properties exploited in radio-based systems.

Radio-based Systems

Systems belonging to this category rely on the *transmission* of radio signals. In the marker-based case, signals are exchanged, through mutual transmission or modulation, either between agents and an infrastructure of readers (centralized localization), or between agents only (mutual localization). In the marker-less case, radio signals emitted by a set of sources, are influenced by the agents moving into the environment, and the analysis of this phenomenon helps to reveal their positions.

Marker-based systems, which represent the vast majority of radio-based localization systems (due to the fact that radio tags are easy to be deployed and less prone to occlusions), often rely on *triangulation*. Triangulation techniques are based either on *lateration* or *angulation*:

- Lateration computes the position of an object by measuring its distance from multiple reference positions using different approaches such as *time-of-flight*, *attenuation* and *phase of arrival*. In the 2-dimensional case (see Figure 2.2a) in order to infer the exact position of a tag, we need 3 reference points (A, B and C); the tag lays on the intersection between the circles having the reference points as centers and the estimated distance from the tag as ray. An additional reference point is required in the 3-dimensional case.

- Angulation is similar to lateration but, instead of distances, angles are used for determining the position of an object (see Figure 2.2b). The tag lies at the intersection between the lines which start from the reference nodes with the calculated angles. It is worth noting that angulation requires a minor number of reference points with respect to lateration (two in the 2-dimensional case and three in the 3-dimensional case) as the "north" represents a fixed additional reference, but it is usually less precise due to the major difficulty in calculating arrival angles with respect to distances.

Figure 2.3 shows how the employed radio technology (e.g., RFID, UWB, WLAN, Bluetooth, Infrared, Ultrasound, Proprietary UHF) influences the accuracy performance (see Section 2.5.1) obtainable by a marker-based system. Additionally, the figure shows how different technologies can scale to an outdoor environment. Noteworthy, several GPS-based technologies [142, 28, 15] have been employed to provide seamless (urban area) indoor-outdoor localization features, but the accuracy is worse than the one obtained with indoor ad-hoc techniques.

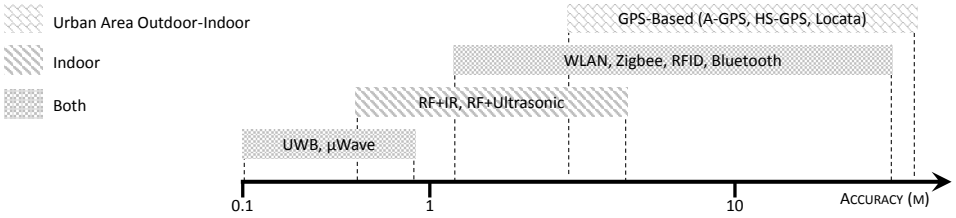


Figure 2.3. Main marker-based wireless approaches for localization with respect to *accuracy* and *scale*.

Radio marker-less localization systems are less frequent. In order to deal with foreground extraction they usually rely on an electromagnetic background model of the scene with respect to some signal source. Agents moving in the environment influence how the emitted signal propagates throughout the air. The analysis of such changes allows inferring the positions of agents.

An exhaustive analysis of radio techniques for localization can be found in [123, 124].

Vision-based Systems

These systems, which are marker-less in the vast majority of cases, use images from cameras in order to detect and track mobile agents. While a trivial solution may be represented by mounting ceiling cameras, this solution raises a set of issues including the limited field of view of cameras (with the subsequent, expensive, installation of multiple cameras) and the impossibility of monitoring heights and framing faces for identity recognition. For these reasons, cameras are usually placed in a convenient position in order to maximize the size of the monitored area (e.g., in a corner), which, on the other hand, introduces an issue due to possible occlusions in the field of view.

A video camera projects a 3-dimensional scene onto a 2-dimensional image plane, thus a common issue vision-based systems need to address is third dimension reconstruction. Different solutions can be employed towards addressing the issue:

- Systems relying on a single camera usually exploit a *3d model of the human body* to deal with the lack of depth. As an example, [206] uses a human model made by a limited number of ellipsoids; for each agent the model can be configured using two parameters, namely the *size*, which models height and scaling of the human, and the *thickness*, which captures the extra scaling in the horizontal direction.

- Systems relying on multiple *time synchronized* cameras can exploit *n-View* geometry [94]. In particular, the so called *stereo vision* localization systems exploit 2-View geometry [24, 95]. A continuously up-to-date comparison of techniques for stereo correspondence in 2-View geometry is available at `http://vision.middlebury.edu/stereo/` (based on the classification introduced in [180, 182]). High accuracy stereo maps can be computed using *structured light* [181]. In particular, the recent advent of PrimeSense Light Coding (cf. `http://www.primesense.com/en/technology`) – e.g., employed by Microsoft Kinect (cf.

http://www.microsoft.com/en-us/kinectforwindows/), which exploits infrared structured light to compute a depth map, opens new scenarios for vision-based localization systems. The processing modules composing a stereo-based vision localization system will be inspected in Section 2.2.

- Finally, other kind of systems, such as robots, exploit *3d laser and ultrasound* scanners in order to obtain the third dimension of the scene. Merging data collected by the camera with measurements obtained by the laser falls into what is known as *sensor fusion*.

Vast majority of academic vision-based systems employ stereo vision. Industrial solutions for localization usually do not rely on vision, as it considered a yet-not-enough mature technology for commercialization.

Inertial Navigation Systems

An *inertial navigation system* (INS) exploits a cluster of inertial measurement units (IMUs) (mostly accelerometers and gyroscopes) to estimate position, velocity and posture of a moving object (a vehicle or a human). An INS is initially provided with its position and velocity from another source (a human operator, a GPS satellite receiver, etc.), and thereafter computes its own updated position and velocity by integrating information received from the motion sensors. An introduction to this kind of systems is given in [86]. INSs are inherently imprecise and can be used in conjunction with GPS systems to provide seamless indoor/outdoor navigation [110].

2.1.3 Identity Recognition Capabilities

Optionally, indoor localization systems may integrate identity recognition as an additional feature, associating, whenever possible, identities of corresponding persons to tracked agents. In marker-based systems, identity recognition can be easily addressed by knowing the association between a tag ID and the agent who wears the tag. In marker-less systems, the task may be much more complex. A particular issue is that, in certain application scenarios (such as security surveillance), agents are not always truthful and so they will probably behave against the identity recognition system.

In vision-based systems, identity recognition is usually addressed by using *face recognition* [121] or more generally *physiological biometry*. An emerging research field, called *behavioral biometry*, tries to recognize humans on the basis of physical or behavioral cues. Currently the most promising example of behavioral biometry is human gait [179].

2.1.4 Level of Support for Posture Estimation

With the term posture, we denote an estimation of the space occupied by an agent. Some localization systems offer this kind of information together with the position of an agent. The provided accuracy may range from simple bounding volumes to accurate estimations employing geometric body models [155].

System providing posture estimation capabilities are usually marker-less (and vision-based). Marker-based systems, in the vast majority of cases, provide only the

position of a mobile agent tag in the space and cannot provide information about the posture, as each mobile agent is usually provided with a sole tag. In order to use marker-based systems for posture estimation, the mobile agent has to wear a huge number of tags; this strategy can yield a very high precision and is used in gaming industry for motion capture but it is, in general, very expensive, thus having a limited applicability in off-the-shelf localization systems.

2.1.5 Physical vs Symbolic Coordinates

A localization service can provide two kinds of information: *physical* and *symbolic*. In contrast to a physical location, a symbolic location encompasses an abstract idea of where something is: in the kitchen, next to a door, next to the fridge, etc.

Physical positions can be expressed in terms of either absolute or relative coordinates. An absolute localization system uses a reference coordinate system shared by all the agents, whilst in a relative localization system each agent can have its own coordinate system (e.g., in rescue scenarios).

Using symbolic locations can be considered as a way to add semantics to physical positions and this feature can be useful for activity recognition services as well as for coordination services [105, 44]. An example of localization systems providing symbolic coordinates is given by proximity systems, where objects corresponding to symbolic positions themselves provide information about agents detected in their proximity.

2.2 Adopted Techniques

Following the terminology introduced in Section 2.1, PLaTHEA is classified as a marker-less stereo-vision based system with identity recognition and pose estimation (bounding-box) capabilities. Figure 2.4 depicts the pipeline of processing modules (the small rectangles in the figure), each one processing the output of the previous one(s). Some of the processing modules access and modify shared data structures represented as smoothed rectangles in the figure.

PLaTHEA takes as input a *stereo video stream*, which is a sequence $\langle (l_0, r_0), (l_1, r_1), \dots \rangle$ of couples of frames coming from two cameras (left and right camera) resembling as much as possible a *frontal parallel* setting (we call *stereo camera* such a device). Frames composing a certain couple (l_t, r_t) are ideally shot at the very same time t; in this sense, left and right video streams are said to be *synchronized*.

The two optical systems composing a stereo camera, though identical in principle, slightly differ, due to the manufacturing process, in terms of internal characteristics (e.g., the focal length) and introduced distortion (e.g., the barrel - or fish-eye - effect). In addition, the frontal parallel setup cannot be perfect (cameras focus may be converging or not aligned).

As a consequence, corrections (namely *undistortion* and *rectification*) are needed to ensure the geometrical properties necessary to perform stereo vision [37]. These tasks strongly depend on the characteristics of both cameras and on their relative positions; these parameters are contained into the *stereo camera model* and obtained through a *stereo calibration* procedure (see Section 2.4.1).

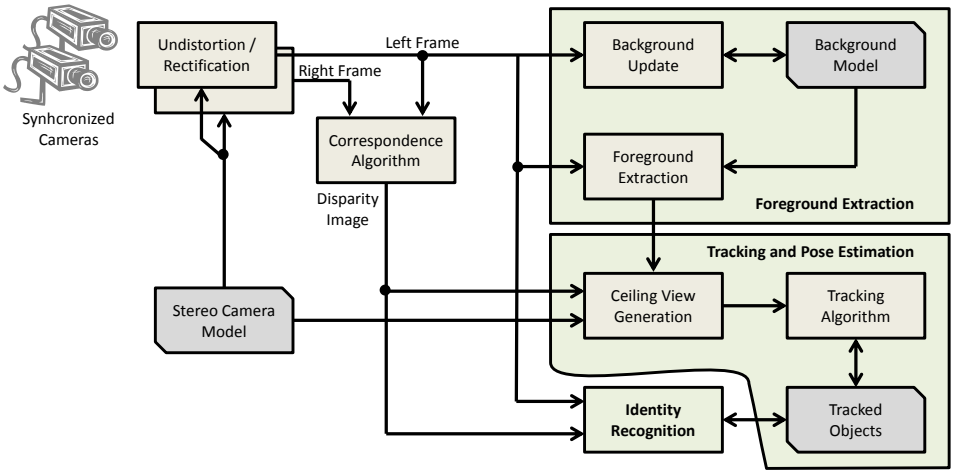


Figure 2.4. A model describing how PLaTHEA works.

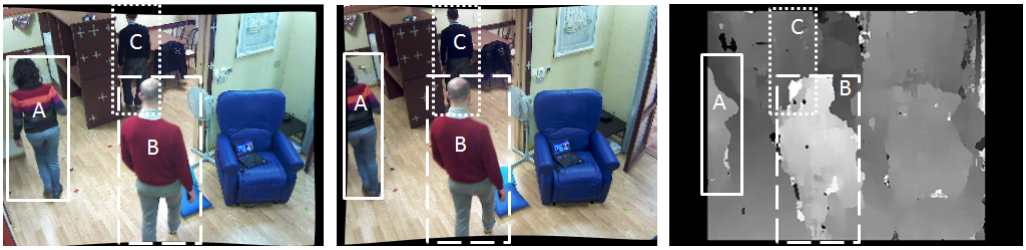


Figure 2.5. The disparity map m_t extracted by a stereo couple (l_t, r_t).

The stereo video stream is employed to obtain (through a stereo correspondence algorithm) at each time step t a *depth* (or disparity) map m_t mapping each pixel of the left camera[1] to a position in the coordinate system having the stereo-camera as origin. At time t the stereo frame (l_t, r_t) is fed into a stereo correspondence algorithm in charge of computing the depth (or disparity) map m_t (see Figure 2.5). As the stereo calibration procedure outcomes l_t and r_t as undistorted and rectified, a pixel $l_t(x_l, y)$ corresponds to a unique pixel $r_t(x_r, y)$ on the very same row. In this way, we can set a point $m_t(x_l, y)$ in the depth map to $x_l - x_r$. The bigger the value of $m_t(x_l, y)$ is (brighter pixels of the depth map in Figure 2.6), the closer to the camera the physical point depicted by the pixel $l_t(x_l, y)$ will be.

If, on the one hand, *commercial* stereo cameras usually offer synchronization, undistortion and rectification for free (sometimes, as in the case of Kinect, a depth map is also provided), on the other hand, native stereo cameras have some drawbacks (e.g., high price, limited range). PLaTHEA offers the support for handling *compound* stereo cameras, allowing to perform these operations via software (see Section 2.3.1).

The right part of Figure 2.4 depicts the core of the system as a set of interconnected macro-modules processing the above described stereo video stream, namely: foreground extraction, tracking and pose estimation, and identity recognition modules.

[1] Choosing the left frame instead of the right one is a purely conventional choice. In the following the left frame will always be employed as reference frame.

2.2.1 Foreground Extraction

The first step needed to track agents inside a generic space is to separate pixels belonging to them (i.e., the *foreground*), from those belonging to the scene (i.e., the *background*). We call this task *foreground extraction*.

At time t the module in charge of foreground extraction takes as input the left frame l_t and the current model of the background, and compares them in order to *extract* the scene foreground. This operation is conceptually comparable to a subtraction. Nevertheless, in the case of vision based systems, the foreground extraction algorithm has to detect video artifacts including *shadows*, *sudden illumination changes*, *light hot-spots* and those deriving from *short term dynamicity*. The latter term refers to all those phenomena caused by periodic activity of the background such as tree leaves moving, curtains, ventilators and screens changing their content. The employed background model must contain all the information needed to recognize these conditions.

In addition, the background model should offer a mechanism to cope with the *long term dynamicity* of the environment. This term refers to the fact that the positions of the objects in the scene (e.g., chairs) as well as the scene itself (e.g., illumination conditions) could change over time. The employed background modeling technique should therefore foresee an online mechanism (called *time adaptivity*) to update the model over time [64].

Background is usually modeled following a statistical approach. Gaussian distributions are often employed to model a pixel belonging to the background. This approach, traditionally applied to background colors, can be in principle applied to a wide variety of background pixels features. Moreover, the use of gaussians can be made arbitrarily complex by using a set of different gaussian distributions for each pixel; this approach is referred to as Mixture of Gaussians (MOGs). A subclass of MOGs is represented by Time Adaptive Per-Pixel Mixture of Gaussians (TAPPMOGs) methods that add time adaptivity to the background model. For example, [95] introduces a stereo vision system which makes use of the TAPPMOG-based approach described in [96]. In this case, time adaptivity is obtained by applying an algorithm derived from Expectation-Maximization (EM) [71].

PLaTHEA employs a novel solution for background modeling, which guarantees both robustness to illumination artifacts, obtained as an improvement over the integration of the techniques introduced in [101] and [65], and time adaptivity by applying the method proposed in [24]. The interested reader can find a thorough description in Appendix 2.8.1.

2.2.2 Tracking and Pose Estimation

A considerable class of stereo-vision localization systems, including PLaTHEA, tracks agents by employing a simulated ceiling view of the environment [66]. Figure 2.6 shows how an acquired stereo frame is turned into a ceiling view map.

Given the depth map m_t, the model of the stereo camera obtained during the stereo calibration procedure and the foreground mask f_t extracted by the foreground extraction algorithm, we can project all the pixels belonging to the foreground to the ground in the coordinate space having its origin in the stereo-camera.

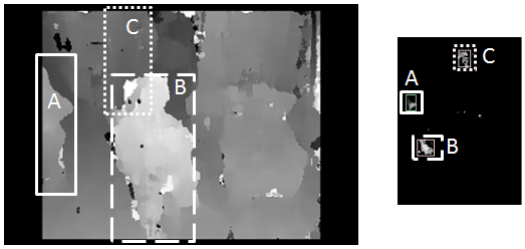


Figure 2.6. An acquired frame with the correspondent ceiling view.

As client applications rely on coordinates relative to the world, we need to transform all these points into a well known coordinate system. In order to do this, a translation vector and a rotation matrix are needed. These structures are obtained as output of the so called *external calibration* procedure. Now all the points can be projected onto the ground by simply ignoring the dimension orthogonal to the ground. As coordinates on the floor are continuous, it is needed to make them discrete by dividing the floor space in minimal square units of side d called *texels*, where d represents the *discretization step*[2]. Each texel corresponds to a pixel in the plan view map. As different foreground pixels are correspondent to the same texel, the value of a pixel in the plan view map actually depends on the number of foreground pixels that map onto it. As farther agents will have less foreground pixels, they will account more in the final count of a texel.

Once the plan view map is ready, we can easily separate blobs (corresponding to agents) in it. Noteworthy these blobs might be easily merged in the foreground. PLaTHEA, as in [95], produces two different ceiling views, namely the *occupancy map*, which reflects how much a certain texel is significant in terms of quantity of detected material, and the *height map*, which indicates, for each texel, the maximum detected height from the floor. Mobile agents are obtained by applying connected components analysis that allows the identification of blobs, each of them potentially representing a mobile agent. The height map is employed to filter out objects (e.g., pets) that are irrelevant to the final application.

The blobs are identified by applying a contour scanner to the plan view map. All the contours whose perimeter exceeds a certain threshold are inserted in the set $\mathcal{D}_t$ of the detected objects at time t. At this point, the set of tracked objects $\mathcal{T}_{t-1}$ is updated by comparing each element with all those contained in $\mathcal{D}_t$. PLaTHEA employs a novel technique for tracking based on Support Vector Machines (SVM) that is thoroughly explained in Appendix 2.8.2.

2.2.3 Identity Recognition

The face recognition method employed in PLaTHEA can be classified as feature-based (structure). Typically, this kind of methods exploits local features such as the eyes, nose, and mouth. These features are first extracted and their locations and local statistics (geometric and/or appearance) are fed into a structural classifier.

At startup, PLaTHEA extracts Scale-Invariant Feature Transform (SIFT) [128] features from each face pose stored, for each person, into the biometric database of

[2]Discretization step d is set, in our system, to 30 mm.

the smart home (see Section 2.3.2). The features of each specific face pose are then sorted into a kd-tree [29] data structure that allows for a fast similarity computation during the on-line recognition phase.

At runtime, a Haar classifier [194] is applied in order to detect faces into the scene and, for each detected face, SIFT features are extracted. At this point, a similarity score is assigned between detected faces and persons in the database. A face is associated to a person in the database if and only if the similarity score is higher than a specific threshold. The similarity score between a detected face and a specific person is defined as the sum of the similarity measures calculated between the former and each face stored into the biometric database for the latter. Such a similarity measure is obtained by counting the number of features in the stored face for which a correspondent feature is found into the detected face. For each feature in the stored face, a *best bin first* search is performed on its associated kd-tree (calculated at system startup) aimed at finding the more similar feature into the detected face. These two features are then said to be correspondent if *(i)* the distance between them is under a selected threshold, and *(ii)* the ratio between this distance and the distance between the detected face feature and the second best matching is under a specific threshold.

Whenever a face is recognized, we use the disparity map to reproject the face onto the floor and then associate the identity to the tracked object present in that position. As disparity data about faces pixels are frequently not available, the ground position is obtained by considering the disparity value of a chest pixel.

2.3 Software and System Architecture

PLaTHEA provides real time information about position and occupancy of agents detected inside a specific room and, possibly, their identities. In this way, basic information is always provided to the upper applications about moving objects, while identities are provided as soon as they become available.

2.3.1 Single-Instance Software Architecture

Algorithms and techniques introduced so far involve a set of computation intensive processing steps that are performed on a per-frame (stereo frame) basis. In order to obtain an effective tracking process, following persons moving at a typical speed for an indoor environment, the system needs to analyze a number of stereo frames per second equal or greater than 10. This means that a time slot not larger than 100ms is available to process each stereo frame. In addition, the software process in charge of analyzing the stream is supposed to run on off-the-shelf (possibly low cost) hardware equipped with a generic (not real-time) operating system. These two considerations make the software design crucial to the performance of the final system.

Figure 2.7 depicts a modular description of a single instance of PLaTHEA where the single software modules, represented by mean of gearwheels, are execution threads. The diagram can be roughly divided into an *acquisition* section containing the two acquisition cameras and the software threads devoted to the acquisition,

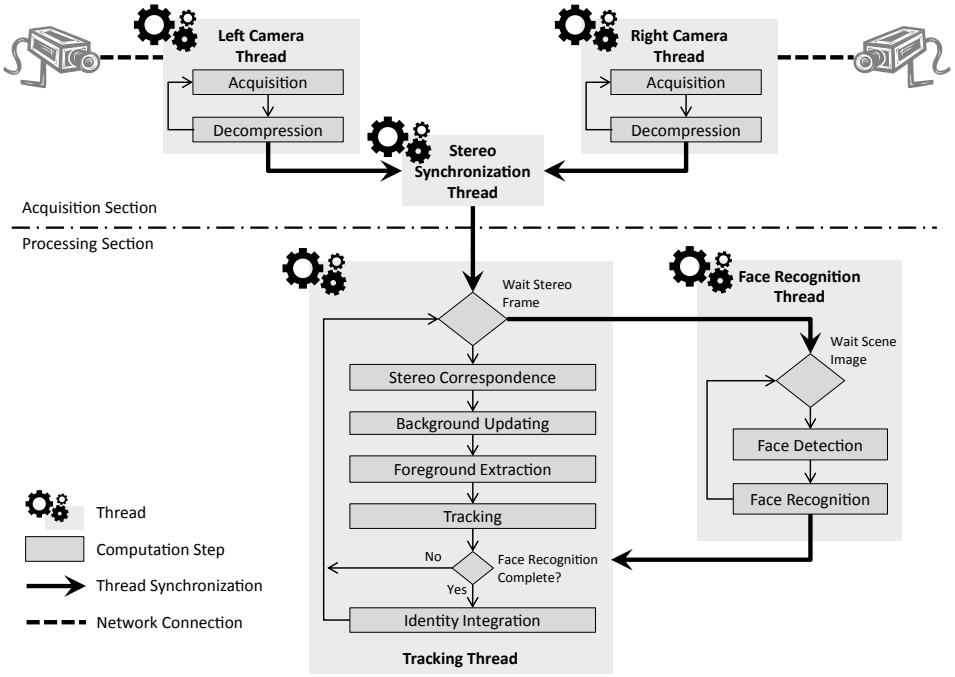


Figure 2.7. A detailed description of the modules composing a single PLaTHEA instance.

and a *processing* section made up of two software threads in charge of tracking and face recognition respectively.

PLaTHEA has been developed under Microsoft Windows as a native (Win32) C++ application. Synchronization among threads is mainly performed by employing software events for signaling and mutex for data integrity. Events are depicted in Figure 2.7 by means of thick arrows connecting threads.

Video Acquisition and Synchronization

One of the goals achieved by PLaTHEA is making the system able to acquire stereo streams from different kind of devices including not only stereo-cameras but also couples of independent cameras (of the same kind) working together. The main reason for this prerequisite is the relatively high cost of stereo cameras with respect to separate, not originally rectified and aligned, cameras. Recently Kinect (a stereo camera based on structured light) replaced vision based stereo cameras in a wide range of applications due to its very competitive performance/cost ratio. Unfortunately, the distance it can operate is fixed in the range 25cm-6m whereas, as we will see in Section 2.4.1, we need to customize it depending on the monitored indoor space.

Moreover, while stereo-cameras are only available as directly connected devices (using USB or FireWire interfaces), single cameras are also available as network attached devices (exploiting TCP/IP for transmitting) allowing an easier deployment into a domestic environment (both USB and FireWire allow for a maximum distance, without repeaters, of about 5 meters).

Supporting *compound* cameras (we use this term to denote a couple of independent

cameras, not originally intended to cooperate, used to acquire a stereo stream) involves including the support for a *calibration phase*, which has to be simple and effective in order to avoid an excessively increased complexity of the deployment phase.

Network attached cameras usually provide access to the video stream using Motion JPEG (M-JPEG, also known as Multipart JPEG). An M-JPEG video stream is requested using a standard HTTP GET message, which the camera responds to with a HTTP response whose payload is a sequence of frames each preceded by a HTTP-like header reporting the features (including the size in bytes) of the following frame. Hence extracting frames from an M-JPEG is thus equivalent to a parsing operation interleaved by JPEG decompressions. This job is performed independently for the two cameras by two separate threads, as shown in Figure 2.7. The extracted JPEG frames are usually lossy, so the quality of the stream extracted from network attached cameras is usually worse than the one obtained with directly attached cameras, and this has to be taken into account during the processing phase.

As an additional drawback, the lack of *synchronization* makes it difficult to uniquely associate a frame acquired from one camera to the corresponding frame acquired from the other one. Asynchrony of video sequences is mainly due to the fact that two entities continuously transmitting over the same medium concur for the transmission. Contention is usually a fair but not synchronized process that inevitably leads to packet losses. In the case of compound network cameras, both grabbers acquire a new frame as soon as the previous one has been delivered, trying to respect the *required* frame rate, and transmit it concurrently using a TCP connection, which provides reliability (through retransmission) and congestion control (through a sliding variable-length window mechanism) at the cost of uncertainty and irregularity of the time needed for the complete transmission of a single frame. This phenomenon is depicted in Figure 2.8, where the activity of the sockets in charge of receiving the two M-JPEG streams is reported. The upper part of the figure shows for each frame the time of arrival (in processor ms) of the first byte on the x axis, whereas the y axis reports the time needed for the reception of the entire frame; the bottom part of the figure reprojects the duration on the x axis showing the duty cycle of each camera [3].

If we consider negligible the shooting time, the transmission time of a single TCP packet and the number of retransmission needed for the first packet of a single frame, we can use the time the socket receives the first byte of a frame as representative of the frame shooting timestamp (*firstByteReceived* event relaxation), so Figure 2.8 is representative of the work performed by each camera for the transmission of the frames.

Let us now consider a trivial algorithm that simply stores the last frame received from each camera and returns a stereo frame as soon as a frame is available for both. This algorithm easily copes with progressive misalignment of video streams due to relative slowdowns. The communication channel is usually in the condition of keeping the transmission time of the frames from both cameras bounded under the inter-frame time interval, thus keeping the misalignment between the two streams almost

[3] The data has been acquired with a complex but static background in order to maintain bounded the difference in size between each frame composing the streams.

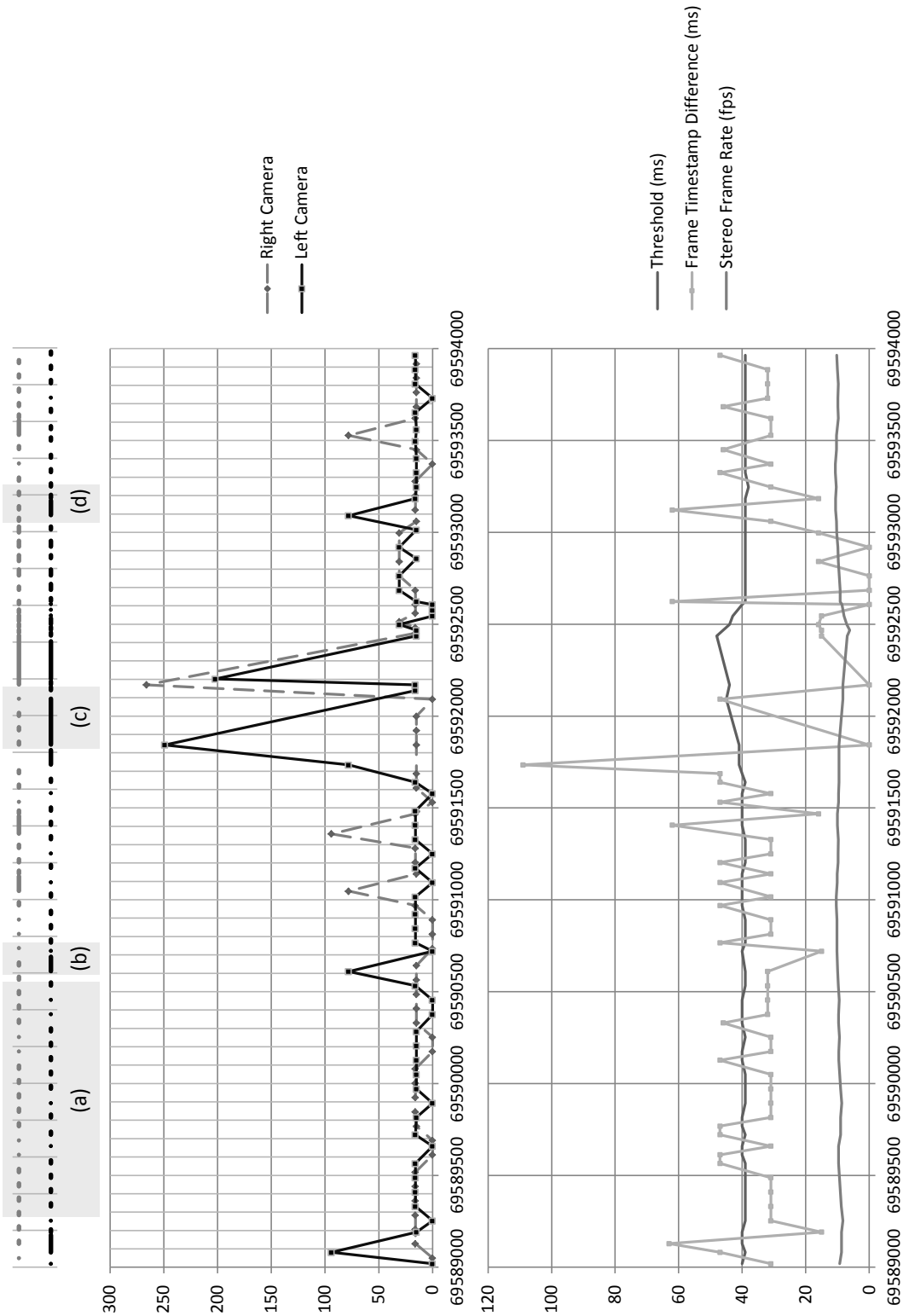


Figure 2.8. The problem of video sequences' synchronization. 5s Profiling.

constant. In this condition (denoted with (a) in Figure 2.8) the trivial algorithm always produces likely stereo frames. The same algorithm may behave well also when a single camera slows down as in the case denoted with (b) in Figure 2.8. Let's now consider condition (c); here the trivial algorithm produces an unlikely stereo frame while two straightforward stereo frames are easily recognizable. Similarly, in condition (d) the algorithm would generate a stereo frame whose validity cannot be easily assessed.

The limit of the trivial algorithm is the application of an eager strategy about frame coupling, which, in addition, does not take into account generation timestamps. In order to face the synchronization issue, we have to clearly define what we mean by a valid or *well-formed* stereo frame.

The frame rate followed by a camera x (being x equal to either LEFT or RIGHT), in general, changes over time due to network load and camera related occasional slow-downs, following a function $f_x(t)$ representing the number of frames received in the last second preceding t whose average value may differ from the requested frame rate f_r. Supposing the two cameras are of the same model and taking into account the fairness of the transmission channel, we can define the *maximum theoretical stereo frame rate* as $f_s(t) = \frac{f_{\text{LEFT}}(t)+f_{\text{RIGHT}}(t)}{2}$. As a consequence, the time between two stereo frames is roughly $\frac{1}{f_s(t)}$, and, considering the misalignment between the two sequences of frames, the best synchronization result will be obtained by considering two frames as composing a *"good" stereo frame* if and only if they are generated within $\frac{1}{2f_s(t)}$ one from each other.

Definition 2.1. *Given a stereo camera with a maximum theoretical frame rate of $f_s(t)$, a couple of frames, one per camera, composes a* well-formed stereo frame *if, considering a common global clock, the difference in time between the generation timestamps of the frames is less or equal to $\frac{1}{2f_s(t)}$.*

In order to cope with stream misalignment, a novel multi-thread synchronization algorithm (Algorithm 2.1) has been devised that returns, with a minimum amount of processing resources, all and only well formed stereo frames according to Definition 2.1.

The algorithm is mainly based on the usage of three frame buffers, two of them containing the frames candidate to become a well-formed stereo frame (represented using the vector of cardinality 2 denoted as `stereoFrame`) and an auxiliary frame buffer (denoted as `auxFrameBuffer`) filled with the last useful frame received from the temporarily *fastest* camera. The evolution of the algorithm is driven by a set of three events. Two of them advise the algorithm, respectively, of the reception of the first byte (with associated timestamp) of a new frame from a camera (the *firstByteReceived* event) and of the completion of the reception of a frame (the *lastByteReceived* event). The last event is generated by the algorithm whenever a new well-formed stereo frame is available. The timestamp of this stereo frame can be obtained by the average between the timestamps of the composing frames.

Theorem 2.1. *Algorithm 2.1 is correct with respect to Definition 2.1, i.e., it returns all and only the well-formed stereo frames.*

Proof. The *only* part ($\Rightarrow$) of the proposition trivially holds because of the timestamp condition imposed to a potential well-formed stereo frame.

Algorithm 2.1: Stereo Synchronization Algorithm

Events Description:

⟨ *firstByteReceived* | c, startTimestamp ⟩ is generated by camera c when, at time startTimestamp, the first byte of a new frame is received.

⟨ *lastByteReceived* | c, img ⟩ is generated by camera c when the reception of the frame img is completed.

⟨ *stereoFrameReady* | leftFrame, rightFrame ⟩ is generated by the synchronizer whenever a stereo couple is valid and ready to be analyzed.

```
upon event ⟨ Init ⟩ do
    stereoFrame[2] := {nullFrame, nullFrame};
    auxFrameBuffer := nullFrame;
    newFrameStarted[2] := {nullTS, nullTS};
end
upon event ⟨ firstByteReceived | c, startTimestamp ⟩ do
    newFrameStarted[c] := startTimestamp;
end
upon event ⟨ lastByteReceived | c, acquiredFrame ⟩ do
    newFrameStarted[c] := nullTS;
    othCam := (c = left ? right : left);
    loadNewImage := stereoFrame[c] = nullFrame or newFrameStarted[othCam] = nullTS or
            |stereoFrame[c].startTimestamp − newFrameStarted[othCam]| >
            |acquiredFrame.startTimestamp − newFrameStarted[othCam]|;
    if loadNewImage then
        stereoFrame[c] := acquiredFrame;
    else
        auxFrameBuffer := acquiredFrame;
        auxFrameSrc := c;
    end
    while stereoFrame[left] ≠ nullFrame and stereoFrame[right] ≠ nullFrame do
        if |stereoFrame[left].startTimestamp − stereoFrame[right].startTimestamp| < 1/(2 f_s(t))
        then
            trigger ⟨ stereoFrameReady | stereoFrame[left], stereoFrame[right] ⟩;
            stereoFrame[left] := stereoFrame[right] := nullFrame;
        else if stereoFrame[left].startTimestamp < stereoFrame[right].startTimestamp then
            stereoFrame[left] := nullFrame;
        else if stereoFrame[left].startTimestamp > stereoFrame[right].startTimestamp then
            stereoFrame[right] := nullFrame;
        end
        if auxFrameBuffer ≠ nullFrame and stereoFrame[auxFrameSrc] = nullFrame then
            stereoFrame[auxFrameSrc] := auxFrameBuffer;
            auxFrameBuffer := nullFrame;
        end
    end
end
```

Before demonstrating the rest of the theorem, we want to point out that in the case that after a bunch of three frames, two from one camera and one from the other, both the possible couples satisfy Definition 2.1, we can consider as valid indifferently either of the two couples as this will not change the precision of the outcome.

Suppose now the *all* part ($\Leftarrow$) of the proposition does not hold. In this case, there would be some well-formed stereo frame which would not be subject to the timestamp comparison. The algorithm applies the timestamp comparison as soon as both main buffers contain a frame (the 'while' cycle ensures this property) and this implies that one of the cameras has received exactly one frame while the other transmitted at least a frame (this is the reason why a single auxiliary buffer is provided). It is ensured that the frames composing the checked stereo frame have the nearest timestamps among those received from the last proposed stereo frame because we can rely on the *firstByteReceived* event relaxation. In conclusion, if a well-formed stereo frame would have been discarded, exactly one of the composing frames should have been discarded by a timestamp rule.

A frame is discarded for one of the following reasons:

- the timestamp comparison fails, so the oldest frame of a candidate stereo frame is discarded. If this happens, this frame would not, in any case, be associated to another frame from the other camera because in that case the difference in terms of timestamps would be greater than the previous;
- a frame, previously put into the main buffer, is replaced by another one. This can happen only if the previous frame was spurious and the supervening frame has a better timestamp with respect to the one that we are still receiving from the other camera;
- a frame, previously inserted into the auxiliary buffer, is replaced by another one (it is surely from the same camera, for the previously mentioned reason). This means that the other camera is suffering from a big slow-down, so if the discarded frame has been previously put into the auxiliary buffer, then it cannot be associated to the frame the other camera is currently receiving. However, if another frame has arrived, in the meanwhile, from the very same camera, it will be more suitable for the association to the next frame arriving from the other camera.

As none of these condition can actually hold, it is not possible that a well-formed stereo frame has been discarded by the algorithm.

As a final remark, please note how the last frame stored into the auxiliary buffer is not immediately discarded. This is because even though it cannot be temporarily associated to another frame, it may be compatible with an incoming frame following the one we are currently receiving from the other camera. □

The bottom part of Figure 2.8 shows the log of the execution of the proposed synchronization algorithm over the time period shown in the upper part of the same figure. Here each point represents a candidate stereo frame (that is a stereo frame which is subject to timestamp comparison). The picture shows that the threshold applied at a certain time and a stereo frame is well-formed if and only if the difference between the composing timestamp is under this threshold. It is also noticeable that,

Algorithm 2.2: TRACKING THREAD

Events Description:

⟨ *stereoFrameReady* | leftFrame, rightFrame ⟩ see Algorithm 2.1.

⟨ *faceRecognitionComplete* | ids ⟩ is generated by the Face Recognition Thread which put found identities into ids.

⟨ *newFaceRecognitionJob* | track, frame ⟩ is generated by the Tracking Thread after the last identities have been integrated into the track object and a new face recognition job has to be performed over frame.

```
upon event ⟨ Init ⟩ do
    track := EMPTYTRACK;
    newIds := NULL;
end
upon event ⟨ stereoFrameReady | leftFrame, rightFrame ⟩ do
    smallLeft := RESIZE (leftFrame, 320 · 240);
    smallRight := RESIZE (rightFrame, 320 · 240);
    disp := DISPARITY (leftFrame, rightFrame);
    track := PROCESS (track, smallLeft, smallRight, disp);
    if identities ≠ NULL then
        track := INTEGRATEIDENTITIES (track, newIds, frd);
        newIds := NULL;
        frd := disp;
        trigger ⟨ newFaceRecognitionJob | track, leftFrame ⟩;
    end
end
upon event ⟨ faceRecognitionComplete | ids ⟩ do
    newIds := ids;
end
```

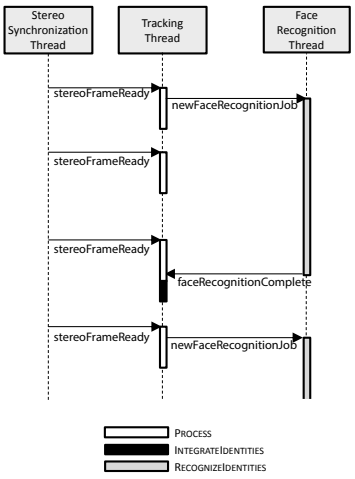


Figure 2.9. Tracking thread and face recognition integration

when the two cameras slow-down, the threshold increases in order to contrast the stereo frame rate reduction. In our tests, Algorithm 2.1 synchronizing two network cameras with a frame rate of 13 fps each allows to keep stereo frame rate between 13 and 10 fps for 95% of time.

A set of tests have also been conducted in order to assess the effective quality of the achieved synchronization. In particular, the test consisted in observing whether or not a stereo frame captured the overcoming of a threshold with respect to a very fast phenomenon (the firing of an incandescence bulb) on both the composing frames. The experiment showed 99% of success over a test bed of 200 firing events proving the quality of the results.

It is worth to note here, how the rate at which the processing section acquires well-formed stereo frames from the synchronizer may in general be different from the rate the synchronizer produces them, due to an excessively needed processing time. In this case, the event-based architecture of the system does not enqueue *stereoFrameReady* events discarding not consumed stereo frames.

The Tracking and the Face Recognition Threads

The sequence of well-formed stereo frames is fed directly into the processing module, which performs two tasks over the set of mobile objects detected into the scene, namely tracking them and trying to assign them an identity. Two different software functions, respectively PROCESS and RECOGNIZEIDENTITIES, are associated with these tasks. At each execution of these functions, the state of the tracked objects is updated following the techniques introduced in Section 2.2.

The PROCESS function takes as input *(i)* a resized, smaller size, version of the stereo frame (tracking is performed in PLaTHEA over frames of resolution 320 by

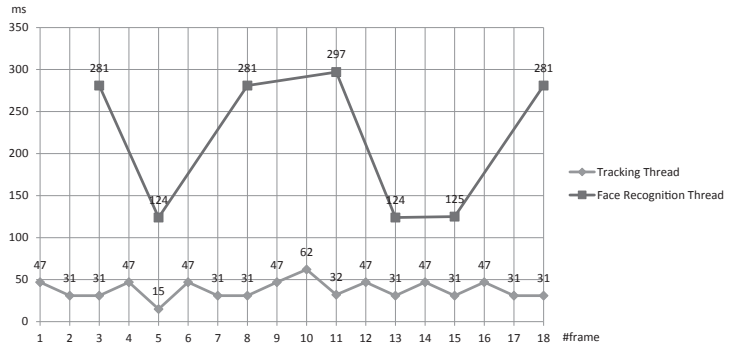


Figure 2.10. Comparison between the computation times of tracking and face recognition threads.

240 pixels, as in most of the works in the literature), *(ii)* the status of each tracked object, and *(iii)* a depth map, and returns as output the updated status of the tracked objects.

The RecognizeIdentities function takes as input the full-resolution left frame and the current status of tracked objects. As described in Section 2.2.3, the function executes the following tasks: *(i)* a face *detection* sub-task (applying a Haar classifier to a gray scale version of the input frame), and *(ii)* a *recognition* sub-task (performed using SIFT). These operations make it hard for the function to be executed on each stereo frame. Moreover, the execution time strongly depends on the number of subjects detected in the environment and on the size of the biometric database.

The computation needed for both functions is very intensive but, whereas a single iteration of Process fits in the time slot dedicated to a stereo frame, the time needed for an execution of RecognizeIdentities does not allow to execute both functions in the same execution thread, thus, two separate threads are employed. The left side of Figure 2.9 shows the algorithm followed by the tracking thread, while the right side shows an execution example (sequence diagram), i.e., the synchronization between the tracking thread and the face recognition thread where the arrows represent event signaling.

The tracking thread is in charge of *(i)* receiving the stereo stream from the synchronizer, *(ii)* executing the Process function, and *(iii)* dispatching identity recognition jobs to the face recognition thread that executes the RecognizeIdentities function. In particular, the tracking thread checks, immediately after the Process function returns, the reception of a *faceRecognitionComplete* from the face recognition thread:

- if the last call to RecognizeIdentities has successfully returned, and has generated a *faceRecognitionComplete* event, the tracking thread integrates the results into the tracked objects. We can assume this operation as performed by the IntegrateIdentities function whose computation cost is negligible. At this point a new face recognition job is assigned to the face recognition thread;
- if the last execution of RecognizeIdentities has not yet returned (no *faceRecognitionComplete* event is received), the check is postponed to the next time slot.

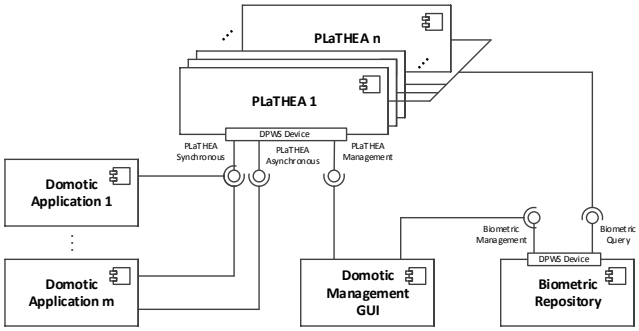


Figure 2.11. The proposed architecture of context aware system.

Figure 2.10 shows a sequence of 18 frames with the time employed by both threads for computation. It can be noticed that, by separating the computation in two different threads, it is possible to process up to 20 frames per second for tracking purposes, whereas face recognition is performed, roughly, every 3 frames depending on the number of faces detected in the scene. Results have been gathered with the software running on an off-the-shelf laptop equipped with a 2.4 GHz Intel Core 2 Duo processor.

2.3.2 System Architecture

The aim of PLaTHEA is to appear to a context-aware indoor system as an additional "high level" sensor. We can imagine a smart home/building featuring a set of these advanced location sensors installed one for each room. In this sense, a single instance of PLaTHEA can be considered as a standalone component providing a set of interfaces to other components of the context-aware system. If we imagine that the whole home/building is covered by a computer network (wired or wireless), all these components may communicate using a *service based* architecture.

Figure 2.11 depicts a set of PLaTHEA instances (one for each room), which are managed by a centralized management graphical user interface. These instances can be used by a set of application components, which potentially receive as input the measurements provided by a huge number of sensors, through specific interfaces. Finally a biometric repository, shared by all the instances, provides information needed by each instance to associate identities to tracked objects. At the actual stage of development, this component gives access to a database of faces needed for face recognition, but it might, in principle, contain many other types of biometric features appropriate for identity recognition.

The current implementation of PLaTHEA supports the DPWS – Devices Profile for Web Services [150] standard. The interfaces have been developed by employing the Java Multi Edition DPWS Stack (in short JMEDS, available at `http://ws4d.e-technik.uni-rostock.de/jmeds/`).

Due to its service-based architecture, PLaTHEA has been initially employed as a localization system in the SM4All European project [48] final demo (October 2011) and since then in some other EU and Italian projects.

Table 2.1. The synchronous and asynchronous DPWS interfaces used in PLaTHEA.

PLaTHEA Synchronous Interface	
Operation	Description
`GetRoomInfo`	returns all the information available about the controlled room
`GetPositionFromPersonID`	takes as input a person id and returns the position of the correspondent user if he is detected in the room
`GetPositionFromObjectID`	takes as input a tracked object id and returns the position of this object
`GetAllPositions`	returns the details, including positions and identities, about all the objects currently tracked
PLaTHEA Asynchronous Interface	
Subscription	Description
`notifyNewObject`	such an event is triggered whenever the system detects a new object inside the scene
`notifyNewIdentification`	such an event is generated whenever a tracked object identity is recognized
`notifyPositions`	a new event of this type, containing the details about all the tracked objects, is generated at each frame
PLaTHEA Management Interface	
Operation	Description
`Start/StopInstance`	asks the PLaTHEA instance to start/stop monitoring its competence area
`GetParameterList`	returns the list of available parameters
`Get/SetParameter`	retrieves/modifies the current value of a configuration parameter

PLaTHEA interfaces

Table 2.1 summarizes the DPWS interfaces provided by each single PLaTHEA instance for tracking purposes. We suppose, without losing generality, that an instance monitors a single room; information about this room, including size and current occupancy, can be obtained invoking the `GetRoomInfo` method.

An application component could be, at any moment, interested in the position of a tracked object (`GetPositionFromObjectID`), of a specific agent (`GetPositionFromPersonID`) or even of the whole set of tracked objects (`GetAllPositions`). A unique identifier is associated to tracked objects and agents, being the former assigned by the PLaTHEA instance (they are unique in the context of the instance), and the latter provided by the biometric repository component, which assigns to each registered user a unique identification number.

Beside these synchronous methods (cf. PLaTHEA Synchronous Interface), each PLaTHEA instance also offers asynchronous calls in the form of subscriptions (cf. PLaTHEA Asynchronous Interface). In the most basic form, these subscriptions provide updates about the detection of new tracked objects (`notifyNewObject`) or the successful recognition of some of the identities of already tracked objects (`notifyNewIdentification`). Application components that need continuous monitoring of the room can subscribe to periodic `notifyPositions` instead of recurring to periodic synchronous calls to `GetAllPositions`.

Finally, each PLaTHEA instance provides an additional synchronous interface for runtime management (PLaTHEA Management Interface), which is employed by the Domotic Management GUI to start/stop it and to fine tune the parameters that

influence its behavior.

Biometric Repository

The architecture depicted in Figure 2.11 models the biometric repository as a distinct component with respect to PLaTHEA. The reason for this is that many components of a context-aware system (including different instances of PLaTHEA) need biometric data of persons who inhabit the house/building.

The biometric repository is supposed to store data for each registered person (e.g., height, build, faces taken in different poses, voice features). This data is available to each component of the home automation system using a common interface called Biometric Query interface. The biometric repository additionally associates to each person a unique identification number in the context of the house, which can be used from multiple contexts to coherently refer to the same individual.

The Domotic Management GUI allows the users to modify the repository by exploiting the Biometric Management Interface; in particular it is possible to add and delete users, and for each of them adding new face poses to the database.

Connecting to the universAAL Architecture

PLaTHEA is intended to be installed as a component of a larger smart home automation system, therefore here we briefly analyze how it can be integrated into universAAL [189].

According to the universAAL architecture, a single PLaTHEA instance would be a context provider, with the PLaTHEA Asynchronous Interface exposed on the context bus. As PLaTHEA also exposes synchronous interfaces for querying (cf. PLaTHEA Synchronous Interface) and management (cf. PLaTHEA Management Interface), each of its instances might be also considered a service callee. Additionally, each PLaTHEA instance would provide user forms aimed at configuration purposes, acting as a user interaction caller. Specific applications, as the Domotic Applications represented in Figure 2.11, might act, with respect to a specific PLaTHEA instance, as either context subscribers or service callers, depending on the specific interface they connect to (i.e., if using the Asynchronous Interface, then they would be subscribers). The Domotic Management GUI would be an user interaction handler. Finally, the Biometric Repository would be an example of a manager node. Finally, it should be noted how the functionalities of DPWS, as a middleware technology, are basically equivalent to the ones provided by the universAAL buses.

2.4 Deployment

Being PLaTHEA a stereo-vision based system, its effectiveness strongly depends on the field of view of the cameras, which limits the covered area even inside a single room. We define an instance as a single installation of PLaTHEA covering a specific area of the house. While all the instances are independent of each other, it is possible to exploit combined information coming from different instances in order to cover the whole house. As an example, if a recognized person moves from one room to another, a new tracked object detected in the latter room could be directly

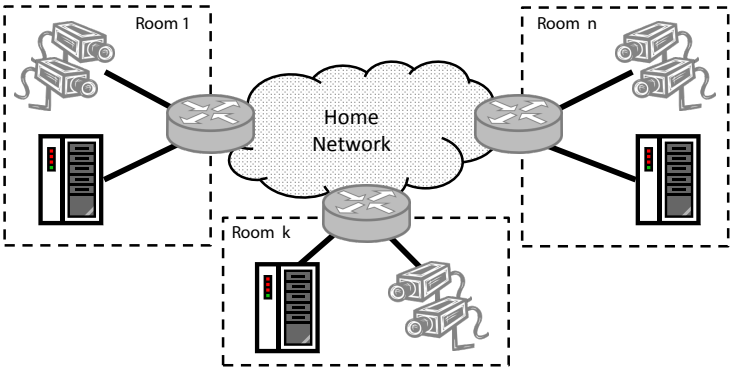


Figure 2.12. Deploying multiple instances of PLaTHEA

associated to the same person. This kind of reasoning, which is an example of *data fusion* [35], is out of the scope of a single instance, and has to be carried out by an external component of the context-aware system.

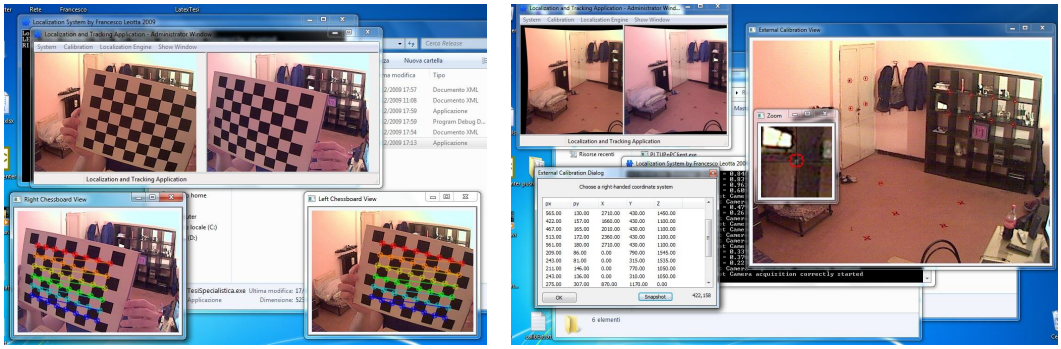
Figure 2.12 depicts the hardware deployment of a full PLaTHEA installation into a single house. A specific instance consists of a stereo rig, a processing unit and a wired network connected to a router. This approach allows to isolate each network segment traffic, ensuring that only packets generated by the DPWS interface of PLaTHEA are injected to the rest of the network (which can be wired or not).

2.4.1 Deployment of a single instance

The first installation step is represented by the physical installation of a stereo-camera. Mounting it in the corner of the room opposite to the entering door, close to the ceiling, allows to obtain the most effective *field of view*. This position maximizes the monitored area providing, at the same time, a privileged position for face recognition purposes.

In case a compound stereo camera is employed, composing cameras have to be mounted as close to a *frontal parallel* setting as possible in order to simplify the calibration. While performing this operation, the choice of the distance between the two cameras, or *baseline*, and of the focal length of the cameras (if modifiable) is fundamental for optimizing the system to work in a certain range of distances [37]. In particular, by decreasing both baseline and focal length we make the *horopter*, which is the space where depth information can be extracted, larger by paying in terms of depth resolution (the minimum distance variation that can be perceived) as distance from cameras increases. Moreover, depth resolution and disparity extraction algorithm performance are influenced by the camera resolution and definition. In our experimental setup, we have chosen a baseline of 20cm with the minimum focal length allowed by the employed cameras, in order to enlarge the horopter as much as possible, thus obtaining a good depth resolution in the distance range 2m-8m.

As the quality of the image decreases when a camera is pointed towards direct sources of light (e.g., sun, hotspots), pointing the compound stereo camera with them on the back is strongly suggested. Moreover, in order to maximize the performance of the face recognition algorithm, the compound stereo camera should be pointed towards the entrance door.



(a) Stereo Calibration (b) External Calibration

Figure 2.13. Camera calibration procedures

Once the stereo rig has been physically installed, the two cameras need to be calibrated, in order to have the generated streams suitable for the stereo correspondence algorithm [37]. Calibration consists of two separated steps, namely *stereo calibration* and *external calibration*, which can be executed using the PLaTHEA graphical installation interface. PLaTHEA uses the OpenCV library (cf. `http://opencv.org/`) in order to perform the processing steps necessary for calibration.

Stereo calibration is performed to discover *intrinsic* parameters (e.g., focal lengths, center of projection) and *distortion* coefficients of cameras, and the position of the cameras one to each other (this includes the exact baseline as well as information about the respective orientation). Stereo calibration can be performed providing a sequence of a fixed number of poses (usually 15) of a rigid chessboard pattern (see Figure 2.13a). The calibration process returns an estimated error which allows to evaluate the quality of the obtained result; the calibration is repeated until a satisfying result is obtained.

Stereo calibration is employed at runtime to *undistort* and *rectify* the two video streams and to obtain the position of every single pixel of the image with respect to the stereo rig coordinate system. Coefficients obtained through stereo calibration remain valid as long as the relative position between the two cameras does not change. In this sense, it is however possible to move the stereo rig as a whole with respect to the environment.

Discovering the position of the stereo rig with respect to the world coordinate system is obtained through external calibration (see Figure. 2.13b). This is fundamental to provide to client components, through the DPWS interface, meaningful coordinates. The result of external calibration is a translation vector together with a rotation matrix. Performing external calibration requires the installer to place a set of markers in the room and for each of them to calculate the exact position in world coordinates. Then, the external calibration window tool can be used to select, using a viewfinder, a marker in the scene and insert its world coordinates.

The results of both stereo and external calibration are stored into an XML configuration file to be used at runtime from PLaTHEA.

As a final step, the installer should refine the behavior of the system at runtime tuning the parameters exposed by each single module of the localization system. These parameters can be tuned while the system is running, by using the graphical

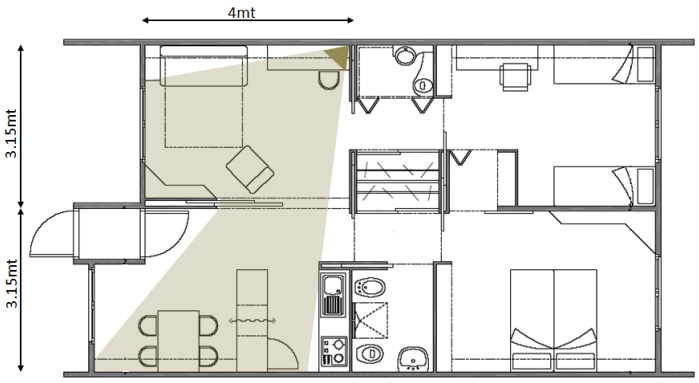


Figure 2.14. The map of "Casa Agevole".

administration interface.

2.5 Experimental Evaluation

Depending on the final application we are aiming at, a home automation system may require a specific degree of precision about the locations occupied by agents in the environment. Whereas in the simplest case a very coarse grain resolution is enough (e.g., what rooms contains agents at a give time) to achieve the needed results, more sophisticated applications may require the knowledge of exact positions, thus the system has to be aware of the uncertainty about measures provided by a localization system.

In order to validate PLaTHEA, a set of experiments have been performed inside a highly automated domestic environment, called "Casa Agevole", intended as a show case for assistive technologies to be used by persons with disabilities (cf. http://www.progettarepertutti.org/progettazione/casa-agevole-fondazione/index.html). "Casa Agevole" is located in Rome, Italy, in IRCCS Fondazione Santa Lucia (a research medical institution), and has been widely used for validation in some EU and Italian projects. In particular, the experiments have been set up in the living room (see Figure 2.14), the size being roughly 6.30 mt x 4 mt x 3 mt. The stereo-camera employed during the evaluation consisted of two independent Vivotek IP7161 network-attached cameras. The maximum resolution supported by the cameras is 1600 by 1200 at 15 frames per second.

The reader interested in repeating the experiments is invited to visit http://www.dis.uniroma1.it/~leotta/demos/plathea/plathea.html, where a link to the source code is also provided. A test package is also available consisting of the PLaTHEA executable and of a small dataset.

2.5.1 Position Test

The performance of a localization system is usually expressed in terms of *accuracy* and *precision*. Accuracy (or location error) usually corresponds to the mean distance error, which is the average Euclidean distance between the estimated location and the true location obtained for an agent. Accuracy can be considered to be a potential

bias, or systematic effect/offset of a positioning system. Precision, instead, reflects how consistently the system works, i.e., it is a measure of the robustness of the positioning technique, as it reveals the variation in its performance over many trials. It is usually expressed in terms of standard deviation or confidence limits expressed, for example, using percentiles.

The accuracy and precision of a localization system are assessed, in the literature, by gathering the measures returned by the system about agents occupying different fixed, known, postures and positions. In these conditions PLaTHEA shows an accuracy of 15cm and a precision of 25cm (90th percentile). On the other hand, as we are interested in proposing PLaTHEA as a building block for a home automation system, we assessed its performance in real life conditions, that is by considering how it is influenced by *(i)* a variable number of agents, *(ii)* different kinds of interactions between them (this includes different degrees of occlusion), and *(iii)* the velocity at which these interactions happen.

Evaluation Platform

Performance evaluation of localization systems is generally an expensive task in terms of human resources, involving the comparison between the measurements provided by the system with the ground truth of movements of mobile agents, using variable conditions in terms of number of mobile agents, interactions between them and environmental conditions (e.g., enlightenment, radio interference).

Whereas a large number of localization systems have been published or commercialized, very little effort have been put in defining common test methodologies useful at comparing different systems. In order to perform our evaluation, we employed the PERICLES test platform [119]. PERICLES works as it follows: during a single evaluation trial, agents move following a set of known paths defined as polylines whose vertices are called *landmarks*. Whenever an agent hits a landmark, it notifies this to the system by generating a *marker event*, by clicking a wireless mouse (a different mouse is provided to each agent). PERICLES matches localization system measurements of positions of mobile agents during the test with the approximated ground truth provided by the agents themselves while generating marker events.

Let us suppose the agent a_i moves inside the environment following a polygonal path $P_i = \left\langle \vec{l}_{i,1}, \vec{l}_{i,2}, \ldots, \vec{l}_{i,n_i} \right\rangle$, where $\vec{l}_{i,j}$ with $j = 1, \ldots, n_i$ are landmarks with known positions on the floor. Moreover, let us suppose a_i able to generate a marker event, with an associated timestamp, every time it is in correspondence of a landmark on the path it is following. Thus a mobile agent a_i that is following a path P_i generates a marker sequence $E_i = \langle e_{i,1}, e_{i,2}, \ldots, e_{i,n_i} \rangle$. The availability of these events, together with the straight movement assumption imposed by the use of a polygonal test path, represents our knowledge of the ground truth path followed by the mobile agent. This ground truth is affected by an error, mainly due to the human reaction time needed for generating a marker event in correspondence of a landmark. In addition, the real movement of the mobile agent might not be perfectly straight. Anyway, for the vast majority of the applications, the obtained approximation represents a good reference to compare the measurements produced by the localization system.

Let $\widehat{P}_i = \langle \vec{p}_{i,1}, \vec{p}_{i,2}, \ldots, \vec{p}_{i,m_i} \rangle$ be the positions measured by the localization system for the agent a_i during a single experiment; as the system generates

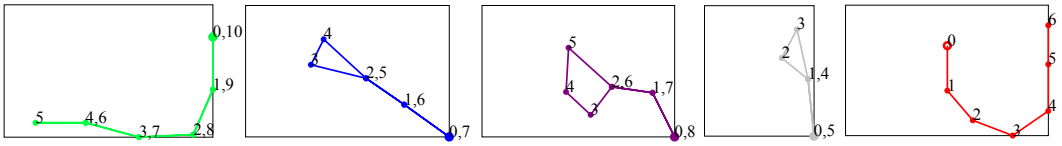


Figure 2.15. The test paths employed during tests.

many measurements per second, we have $m_i \gg n_i$; we denote with $\widehat{P}_i^E = \left\langle \vec{p}_{i,j_{i,1}}, \vec{p}_{i,j_{i,2}}, \ldots, \vec{p}_{i,j_{i,n_i}} \right\rangle \subset \widehat{P}_i$, with $j_{i,1} < \cdots < j_{i,n_i} \leq m_i$, the subset of measurements corresponding to the marker events in E_i. Performance measures provided by PERICLES report both accuracy and precision with respect to the information provided by the system (we denote with $|\vec{x}|$ the euclidean norm of the $\vec{x}$ vector):

- The *position over landmark* (POL) error aggregates the differences measured between a certain landmark in the path and the estimated position of the agent in the instant it generates the marker event correspondent to that landmark; it is measured as:

$$\frac{\sum_{k=1}^{n_i} \left| \vec{l}_{i,k} - \vec{p}_{i,j_{i,k}} \right|}{n_i} \tag{2.1}$$

- The *straight line drift* (SLD) represents the ratio between the distance between two landmarks and the estimated length covered by the agent; it is measured as:

$$\frac{\sum_{k=2}^{n_i} \left[\frac{\sum_{z=j_{i,k-1}+1}^{j_{i,k}} |\vec{p}_{i,z-1} - \vec{p}_{i,z}|}{\left| \vec{l}_{i,k-1} - \vec{l}_{i,k} \right|} \right]}{n_i - 1} \tag{2.2}$$

- The *measured height* (MH) and the *measured box base* X (MBBX) and Y (MBBY) errors show how the bounding box estimated by the system differs from the ground truth one. The ground truth bounding box is estimated by combinining the build of a_i, which is contained in the biometric repository, with the direction it is following on the path P_i.

During the evaluation of PLaTHEA, we added another performance measure that denotes the number of different tracked objects (NT) assigned to a single person during an experiment. This measure is useful to inspect the performance of the tracking algorithm.

Results

Five different test paths (see Figure 2.15) have been placed into the living room of "Casa Agevole", which span the entire room intersecting one with each other. This setup allowed to assess the performance of PLaTHEA with up to five persons at the same time. In particular, in order to perform a stress test of the system, experiments have been performed with a growing number of individuals moving at the same time in the monitored environment. For each occupancy condition, persons were asked to perform the experiment at three different velocities (slow - about 0.5m/s, moderate

Table 2.2. Test results

Num. Users	Speed	POL (cm)		SLD		MH (cm)		MBBX (cm)		MBBY (cm)		NT
		Avg.	90%	Avg.	90%	Avg.	90%	Avg.	90%	Avg.	90%	Avg.
1	Slow	15.0	30.0	1.7	2.0	3.1	8.3	3.9	8.0	5.2	8.7	1.0
	Moderate	16.7	32.0	1.7	2.2	3.3	8.0	4.0	7.7	4.7	7.5	1.0
	Fast	26.0	46.0	1.9	2.6	3.2	9.0	4.0	7.4	4.4	7.0	1.2
2	Slow	22.7	40.0	2.1	3.3	3.3	9.0	4.5	9.0	3.2	6.9	1.3
	Moderate	34.0	44.0	2.2	3.0	2.9	8.0	5.4	11.0	3.4	7.5	1.4
	Fast	38.0	50.0	2.3	2.9	3.9	10.6	5.4	11.7	3.9	8.0	1.3
3	Slow	21.0	31.0	1.8	3.0	3.0	6.6	4.0	8.0	3.1	6.9	1.4
	Moderate	25.0	40.0	1.9	2.9	2.9	7.4	4.7	9.8	3.2	6.5	1.3
	Fast	31.0	46.7	2.0	2.9	3.0	7.8	5.0	11.0	3.4	7.5	1.6
4	Slow	23.8	34.0	1.7	2.1	3.3	7.6	4.0	8.7	3.7	7.8	1.8
	Moderate	27.0	38.8	1.8	2.2	2.9	7.0	5.1	10.8	3.5	7.3	2.0
	Fast	36.0	56.0	1.9	2.6	3.5	9.0	4.3	9.7	4.3	7.9	2.0
5	Slow	41.0	63.0	2.4	3.4	4.2	10.5	4.9	9.0	4.6	8.9	2.2
	Moderate	36.0	65.0	2.3	3.4	3.6	9.4	5.2	10.2	4.0	8.5	2.2
	Fast	48.0	82.0	2.7	3.8	4.7	11.8	5.5	11.3	4.0	8.5	1.8

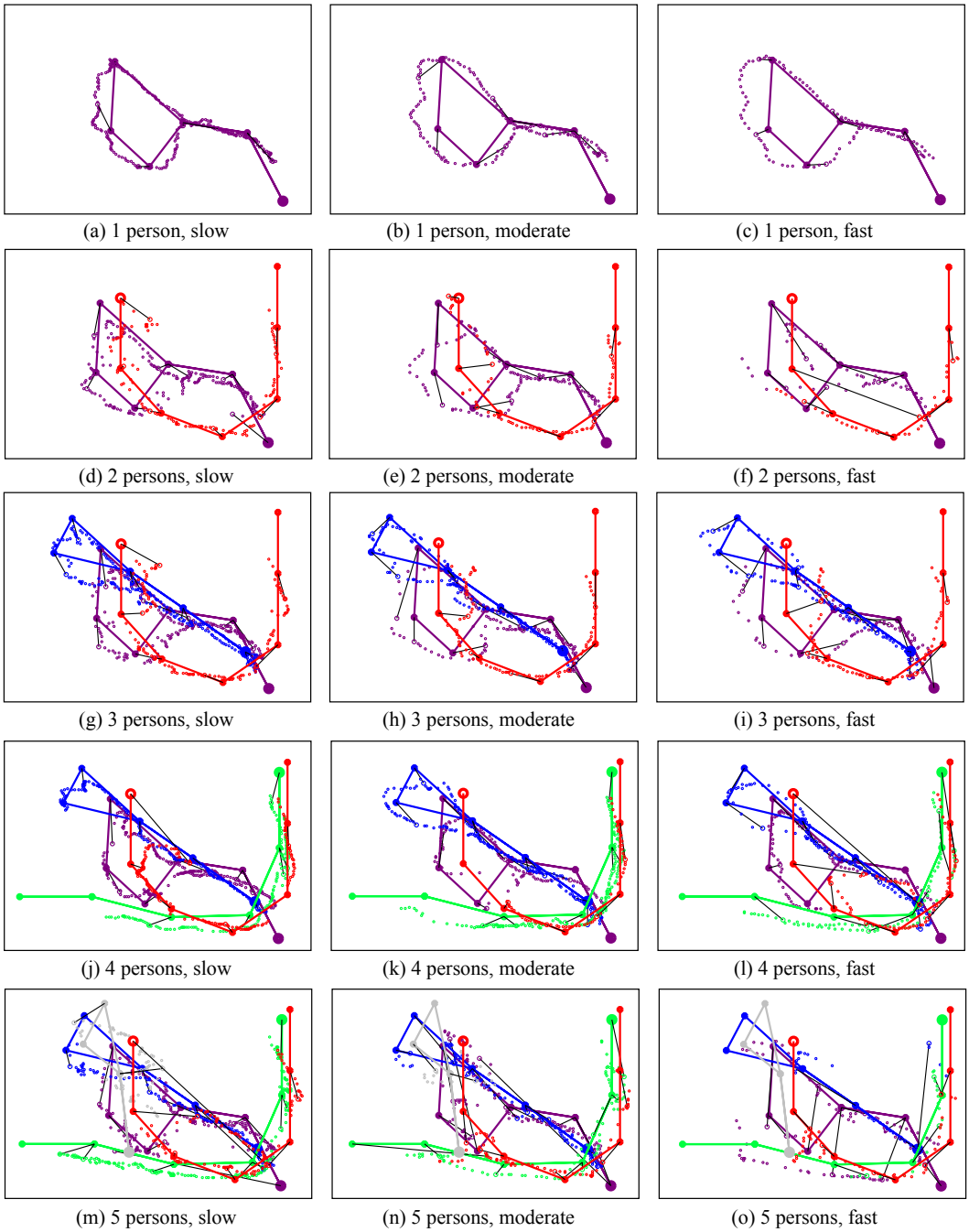
- about 1.0m/s, and fast - about 1.5m/s) in order to understand how the system works when people move faster. Persons were also asked to interact with each other without following specific instructions in order to assess the robustness of the system to such kind of events. For each of the different 15 conditions, 10 repetitions have been performed for a total of 150 experiments. The results are shown in Table 2.2.

Most errors measures are almost invariant with respect to both the number and the velocity of the agents. On the other hand, the POL error measure seems to be influenced by the velocity at which people were moving. While an explanation to this could be found in a possible synchronization error between the cameras, by thoroughly analyzing trajectories (see Figure 2.16), it can be noticed that this variation is introduced by the agents itself while signaling their positions over a landmark, as this task becomes harder and harder when walking is faster.

From the table it can be also noticed how performance worsens with 5 persons in the room. As the occlusions between agents necessarily increase, performance gets worse; additionally the number of tracked objects needed to follow a subject increases. This has to be taken into account when designing home automation services built on top of PLaTHEA. Precision requirements for some services could be held only with a limited number of subjects. Anyway, remarkably, PLaTHEA is aimed at an indoor domestic usage and, as the 2011 Italian census [104] reported, a family is composed, in the average, of 2.4 persons[4]; as a consequence, in the vast majority of cases, the accuracy and precision of the system are suitable for most final applications. Figure 2.16 depicts some executions representing the different conditions reported in Table 2.2; here trajectory points provided by PLaTHEA, and represented as small circles, are compared to the ground truth paths introduced in Figure 2.15; solid lines connect landmarks with the corresponding measurements.

The ground truth trajectories are shown using a bold line whereas the measurements are represented using small circles. Among the measures, the ones whose timestamps are correspondent to marker-events triggered by agents are connected

[4]This is quite similar in most of Western countries (US and Europe).



(a) 1 person, slow (b) 1 person, moderate (c) 1 person, fast

(d) 2 persons, slow (e) 2 persons, moderate (f) 2 persons, fast

(g) 3 persons, slow (h) 3 persons, moderate (i) 3 persons, fast

(j) 4 persons, slow (k) 4 persons, moderate (l) 4 persons, fast

(m) 5 persons, slow (n) 5 persons, moderate (o) 5 persons, fast

Figure 2.16. PERICLES output for different test conditions.

to the corresponding markers.

2.5.2 Robustness to Sudden Illumination Changes and Background Changes

Sudden illumination changes and background changes represent the most frequent class of events that can negatively influence the behaviour of PLaTHEA in real usage conditions. In both cases, the background model is not valid anymore and it needs to be updated as it does not allow to perfectly extract the foreground.

As the foreground extraction algorithm described in Appendix 2.8.1 is robust to sudden illumination changes, the extracted foreground is only partially influenced by this kind of events. Nevertheless, as an illumination change hits the whole background, updating it can be a slow process if agents are concurrently moving in the scene.

The impact of background changes on the performance of the system is, instead, more complex to analyze. Moving an object from one point to another creates two blobs in correspondence of the source and target positions, respectively; as long as the moved object is small with respect to a threshold parameter, the tracking algorithm will not consider these two blobs as denoting a new tracked object. If the moved object is big (e.g., a cabinet), two tracked objects will be erroneously detected by the system until the background update algorithm will absorb them. The time needed to update the background is, in this case, strictly dependent on the agent activity around the source and the destination blobs.

2.5.3 Performance of Identity Recognition

As described in Section 2.2.3, PLaTHEA features identity recognition capabilities based on SIFT features. Associating identities to tracked objects makes it possible to provide inhabitants with a customized domestic life experience.

In order to assess the performance of the identity recognition subsystem, it is necessary to distinguish between face detection, performed by using a Haar classifier, and face recognition, performed by matching SIFT features from the faces stored in the biometric database with those extracted by the detected faces.

A Haar classifier is very well suited for face detection and behaves very well in good lighting conditions. It is even possible to boost performance by applying more than one classifier in sequence. As a consequence, only the face recognition algorithm has been validated.

During the test, a biometric database containing 50 face poses of 5 persons (10 poses for each inhabitant) has been employed. In order to cope with false positives in the recognition algorithm, an identity is associated to a tracked object if and only if the former has been returned by the recognition algorithm twice in a row for the latter.

The performance of the face recognition algorithm is given in terms of correct answers, identity mismatches and missed answers returned by the algorithm over a total of about 2500 decisions taken (see Table 2.3). For each class of results, the average similarity score between the detected face and the assigned identity is reported.

Table 2.3. Identity recognition performance.

	Result	Avg. Score
Correct	1780 (71.5%)	67
Wrong	198 (8%)	32
Missed	513 (20.5%)	10

These results have been collected using the same sequences employed for assessing position precision and accuracy. As a consequence, the results are influenced by the distance between the tracking object and the camera, which, in turn, influences detection rate and successful identification.

2.6 Related Work

Performing localization over video streams is a quite expensive task from the computational point of view. While first solutions to the problem employed multiple hypothesis trackers, due to the limited computational power available, recent approaches rely on *continuous detection* by leveraging the capacity of analyzing video streams at a reasonable frame rate (around 10 frames per second) provided by modern CPUs [33].

The idea of using plan-view maps simulating a ceiling view was first introduced in [66]. While this approach is employed by many recent localization and tracking systems, a major limitation consists in the fact that foreground extraction is performed as a logical difference operation between the current frame and an immutable background model. A system designed for a practical application must take into account short and long term variation of the background, as introduced in Section 2.1.1. An initial solution to the background problem is given in [95], where TAPPMOGs are employed. One limitation of this approach is that it considers pixels as independent entities without considering the relationship with the underlaying object. A solution to this issue is given in [24], where the concept of row and column activity, based on edge detection, is used to define a correspondence between pixels belonging to the same object/agent. This approach is quite similar to the one adopted by PLaTHEA.

Authors in [24] also include disparity map into the foreground segmentation process. This has the advantage of easily coping with shadows and illumination changes, which are indeed absent into disparity maps. While this approach shows excellent performance, it strongly depends on the quality and stability of the disparity map which, in our experiment, could be a strong assumption in the case of cameras that employ lossy compressed video standards (e.g., network attached cameras). Thus PLaTHEA uses a different approach to deal with illumination artifacts, as described in Appendix 2.8.1.

Tracking in people localization systems is usually based on the comparison between a set of tracked object templates maintained by the system and a set of potential objects in the current frame. For each couple, a similarity score is assigned, obtained as a linear combination of different features whose coefficients are chosen experimentally. PLaTHEA follows a slight different approach, as explained in Appendix 2.8.2, based on SVM which saves the installer from manually selecting coefficients.

Table 2.4. Classification of presented indoor localization systems.

System	Marker Less	Sensing Technology	Identity Recognition	Pose Estimation
PLaTHEA	✓	Stereo-Vision	✓	Bounding Box
Harville [95]	✓	Stereo-Vision		Bounding Box
Iocchi [24, 35, 36]	✓	Stereo-Vision		Bounding Box
Marker-based RADAR [25]		Radio	✓	None
Marker-less RADAR [25]	✓	Radio		None
EKAHAU [175]		Radio	✓	None
WSN Based [193]	✓	Radio		None
ENSCO [80]		INS	✓	None

The system initially introduced in [24], has been recently extended in [35] and [36]. Here the tracking method is adaptive with respect to the level of crowding of the monitored area; the proposed approach performs tracking using optical flow over heads. Whenever the room become too crowded, a multiple hypothesis tracker is employed. These two works additionally introduce the concept of localization system as a building block of a context-aware system (a hospital in this case).

Differently from other proposed solutions, PLaTHEA is designed to be integrated into a specific application: a context-aware home automation system. Following this approach, it is seen as an additional sensor providing information about agents moving into the home. From the point of view of the rest of the system, it is a black box exposing a standard interface. For the same reason it is conceived to work with network attached cameras; this choice, while making the problem more difficult (see Section 2.3.1), allows the system to be easily installed with respect to traditional buses (e.g., USB, FireWire).

PLaTHEA is a vision-based indoor localization system, nevertheless it is meaningful to compare its performance also with those systems that are based on different sensing technologies. Selection criteria for the comparison include the significance of the work and the need of maximizing the coverage of different technologies. First works in the field of indoor localization were based on radio transmission, due to the possibility of using the techniques already employed outdoor. Among these early radio-based systems, RADAR [25] represents a seminal work including both a marker-based and a marker-less profile exploiting the standard wireless LAN (IEEE 802.11) infrastructure. RADAR marker-based profile inspired a set of commercial systems including EKAHAU [175]; RADAR marker-less profile, while more precise, is very sensitive to significant changes in the environment, such as moving metal furnitures or large groups of people congregating.

Recently a marker-less radio based approach based on a generic WSN infrastrucure has been proposed in [193]. Here the authors divide the floor into cells and then the association of an agent is transformed into a classification problem solved using the Support Vector Machine (SVM) algorithm. As this is a preliminary work, evaluation has been conducted with a single person.

ENSCO [80] developed an inertial navigation system for disaster management and war operations.

Table 2.4 shows how the systems introduced throughout this section can be classified following the dimensions defined in Section 2.1.

Table 2.5. Performance of presented indoor localization systems. NA: Not Applicable

System	Accuracy Avg. (cm)	Precision (cm)		Pervasive System Integration	Crowded Environment	Background Updating
PLaTHEA[5]	15	25	90%	+++	+	++
Harville [95]	18	32	84.1%	-	-	+
Iocchi [24, 35, 36]	10	High		+	+++	++
Marker-based RADAR [25]	290	470	75%	-	++	NA
Marker-less RADAR [25]	430	NA		-	-	-
EKAHAU [175]	300	500	90%	++	+	NA
WSN Based [193]	10	NA		-	-	-
ENSCO [80]	NA	300	95%	-	+	NA

Table 2.5 briefly compares the performance of these systems, with respect to the measures introduced in Section 2.5.1 and their ability to cope with requirements typical of indoor environments. The values of accuracy and precision reported in the table are directly taken from the corresponding papers. The symbols + and − denote how much a specific aspect is taken into account by the authors of the cited articles; a − indicates that no indication has been provided for a certain aspect:

- for the pervasive system integration aspect, possible positive scores are: + (support for integration available but not detailed), ++ (support for integration available and documented, but not service-based), and + + + (service-based support for integration is documented);
- for the crowded environment aspect, possible scores are: + (a maximum number of users is specified), ++ (a solution is provided but exact positions are lost), and + + + (a solution is provided that preserves positions while losing occupancy information);
- for the background updating aspect, possible scores are: + (supported, but updating is performed on a per-pixel basis), and ++ (supported, updating is performed on a per-area basis allowing to not incorporate still persons).

As can be easily noted, vision marker-less systems generally have better performance with respect to radio marker-based systems; on the other hand, they present a major cost in terms of computational resources. Radio marker-less systems seems to over-perform radio marker-based systems, but the ones here presented (namely [25] and [193]) do not cope with background changes.

Performance strongly depends on both the localization infrastructure and the crowding level of the environment. For example, EKAHAU performance is evaluated with two agents in the environment and with a number of 802.11 access points varying from 3 (500cm of accuracy) to 10 (300cm of accuracy).

2.7 Discussion and Conclusions

This chapter has presented PLaTHEA, a people localization and tracking system to be used as a component in larger context-aware systems managing smart spaces.

[5] PLaTHEA accuracy and precision refers to the static tests in order to have a fair comparison.

In particular, a single instance/deployment of PLaTHEA can be considered, by the upper layers of a complex system, as a sensor, to be complemented by other traditional sensors employed for home automation, to fully define the *context* of the home. Information collected by each single instance is provided as a DPWS service, thus making it transparent to the client application components.

The contribution of this work is to integrate several techniques in order to target them to smart homes. This has been obtained by applying both existing techniques and novel ones for video acquisition employing network-attached cameras and for tracking (see Section 2.3.1 and Appendix 2.8.2 respectively). The system is exposed to a smart home service-based architecture as a sensor component with a well-defined DPWS interface (see Section 2.3.2).

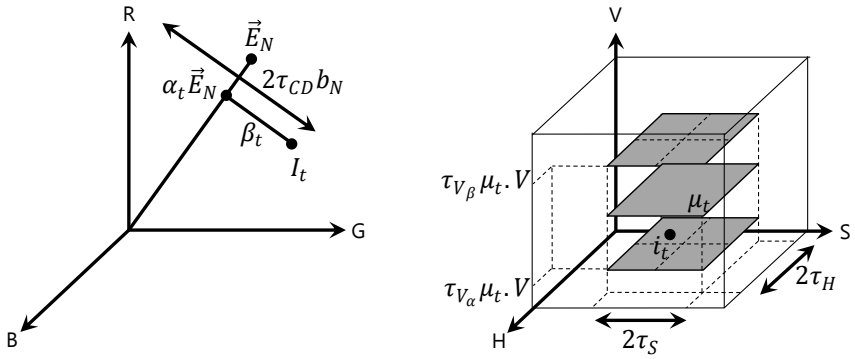
An important issue is the overall acceptability by end-users, i.e., inhabitants of the smart space, who may have concerns about their privacy. Whereas people are getting more and more used to be video-filmed in public spaces (e.g., banks and airports) and accept this for security reasons, they might raise objections about being video-filmed in their homes. Indeed, as described in this chapter, there is no diffusion of the videos outside a single PLaTHEA instance, and it is quite straightforward to make the network on which the video cameras exchange frames really secure; indeed, the usual WiFi router each of us has in its own home is a greater concern for security and privacy than PLaTHEA. But we are well aware that the *perception of security* is different from the actual security (cf. the recent news about the supposed Prism programme from US NSA); therefore we plan, in future work, to conduct an extensive user study, with normal bodied, elderly and disabled persons, about the acceptability of PLaTHEA and the trade-off between offered services and privacy concerns. Preliminary, not systematic tests have shown that, especially in the case of elderly and disabled persons, the advantages of being tracked in the smart space outperform the privacy concerns, as this gives them a sense of "taking care of them" which is their major desire.

On the more technical side, the system has to be made more robust to crowded environments; this can be done by relaxing the amount of information provided to the client applications, e.g., in condition of crowding it could be difficult to provide information about occupancy of subjects. In this sense, a future improvement might be to allow selecting a specific agent inside a crowd and track it for security monitoring.

The issue of supporting very large rooms is connected to that of managing crowded environments. Even though it is possible to change the horopter by fine-tuning the available parameters (i.e., focal length and baseline), it is not plausible to monitor a room featuring one of the sides longer than 10m, without employing costly acquisition hardware [6]. A solution to this problem is to perform the sensor fusion of multiple localization systems into a single space. This might also represents a way to improve the performance of tracking and identity recognition by resolving inconsistencies among the measures coming from multiple PLaTHEA installations about overlapping areas.

Similar conclusions can be argued about identity recognition performances, as

[6] The reader should remind that during our tests, cf. Section 2.5, we adopted commercial cameras and monitored a room of size 6.30 mt by 4 mt.



(a) α and β distortions [25] (b) Shadows in HSV color space [26]

Figure 2.17. Foreground extraction features for a specific pixel (x, y).

they are strongly influenced by the test environment and the employed hardware. The latter, in particular, plays here an even more fundamental role.

Light conditions represent another fundamental source of concern for vision-based localization systems. Although, in principle, color cameras might be replaced by infrared ones, PLaTHEA employs color histograms as tracking features. As a consequence, tracking performance decreases in case grayscale frames, as those provided by an infrared camera, are fed into the system.

A typical drawback of vision processing is the computational cost. Processing a stereo vision stream for localization purposes is a very CPU intensive task. Anyway, modern off-the-shelf CPUs are powerful enough to support, without wearing off, a stereo stream of 10 frames per second that is considered as a minimum requirement in order to achieve satisfying tracking performances.

Further improvements can be introduced about dynamic background; in particular, the current system shows bad performances in presence of mirrors. Even though the current approach performs pretty good in presence of still persons, correspondent pixels are eventually absorbed into the background if a big amount of time elapses.

Finally, PLaTHEA only supports bounding-box based pose estimation. Future work will include support for skeleton detection in constrained conditions. As an example, it would be possible, by using Microsoft Kinect as input device, to perform skeleton detection in a short range for a limited number of subjects.

2.8 Appendix: Details on Techniques

2.8.1 Foreground Extraction

The background model employed by PLaTHEA basically consists of two sub-models: the *color model* and the *edge intensity model*.

Let us suppose, initially, that our background modeling technique does not support time adaptivity. In this case the background model would be acquired during a bootstrap phase of N frames and not updated anymore. The color model associates to each pixel (x, y) three gaussian distributions $\mathcal{N}(\mu_N(x, y).c; \sigma_N(x, y).c)$, with $c = \{R, G, B\}$, each one corresponding to a component in the RGB color space.

Following [101], at time t, two measures $\alpha_t(x,y)$ and $\beta_t(x,y)$ representing, respectively, the *brightness* and *chromaticity* distortion of the pixel $l_t(x,y)$, can be derived from the normalized background model $\vec{E}_N(x,y) = \left[\frac{\mu_N(x,y).R}{\sigma_N(x,y).R}, \frac{\mu_N(x,y).G}{\sigma_N(x,y).G}, \frac{\mu_N(x,y).B}{\sigma_N(x,y).B}\right]$ and from the normalized intensity $\vec{I}_t(x,y) = \left[\frac{i_t(x,y).R}{\sigma_N(x,y).R}, \frac{i_t(x,y).G}{\sigma_N(x,y).G}, \frac{i_t(x,y).B}{\sigma_N(x,y).B}\right]$ where $\vec{i}_t(x,y)$ is the color of the pixel (x,y) in the RGB color space. As shown in Figure 2.17a, the brightness distortion $\alpha_t(x,y)$ is the value that minimizes the euclidean norm $\left|\left|\vec{I}_t(x,y) - \alpha_t(x,y)\vec{E}_N(x,y)\right|\right|$, whereas the chromaticity distortion $\beta_t(x,y)$ is the value of the norm with the selected (i.e., calculated) value of $\alpha_t(x,y)$. Intuitively, brightness and chromaticity distortion represent sudden variations of, respectively, the brightness and the chromaticity components of a pixel due to temporary light artifacts and distortion introduced by the camera. Let us now calculate brightness and chromaticity distortion and average them as follows:

$$\mu_N(x,y).BD = \sqrt{\frac{\sum_{t=1}^{N}(\alpha_t(x,y)-1)^2}{N}} \qquad \mu_N(x,y).CD = \sqrt{\frac{\sum_{t=1}^{N}(\beta_t(x,y))^2}{N}} \tag{2.3}$$

As a consequence, $\mu_N.BD$ and $\mu_N.CD$ are part of the color model as well as the color gaussians previously introduced. If we introduce a constant τ_{CD}, we can classify as part of the foreground those pixels such that:

$$\frac{\beta_t(x,y)}{\mu_N(x,y).CD} \leq \tau_{CD} \tag{2.4}$$

This rule allows also to cope with sudden illumination changes. While a similar reasoning with respect to the brightness distortion value can be done to cope with shadows, this method showed poor performance in our experimental setting.

Authors in [65] show how the properties of Hue-Saturation-Value (HSV) color space can be exploited to easily recognize a shadow. The HSV color space clearly separates chromaticity and luminosity. A pixel belonging to a shadow shows a substantial variation only in the value component of the pixel while hue an saturation values slightly change with respect to the color model. We can then define four constants (valid for each pixel) representing *(i)* the maximum variation of the saturation τ_S and of the hue τ_H with respect to the pixel color model, and *(ii)* the minimum τ_{V_α} and maximum τ_{V_β} scaling factor for the value component of the pixel color model. Figure 2.17b shows how these constants define a parallelogram, containing the pixel background $\mu_N(x,y)$, that defines the HSV subspace containing pixel color value corresponding to shadows. In order to be classified as a shadow pixel, a pixel (x,y) has to fulfill the following constraint that defines the above mentioned parallelogram:

$$|\mu_N(x,y).S - i_t(x,y).S| \leq \tau_S \wedge |\mu_N(x,y).H - i_t(x,y).H| \leq \tau_H \wedge \tau_{V_\alpha} \leq \frac{i_t(x,y).V}{\mu_N(x,y).V} \leq \tau_{V_\beta} \tag{2.5}$$

The output of the foreground recognition algorithm consists of a foreground mask f_t such that $f_t(x,y) = 1$ if and only if the pixel is not a shadow one (by first applying Equation 2.5) and its chromaticity distortion is not excessive (by finally applying Equation 2.4). As a final step of foreground extraction, a median filter is applied to the extracted foreground mask f_t aiming at eliminating little impurities

Figure 2.18. The foreground extraction algorithm.

of the foreground. An example of execution of such a procedure is shown in Figure 2.18.

An alternative solution to the foreground extraction problem, presented in [24], exploits depth map in order to detect shadows and sudden illumination changes. The ratio behind this approach lays in the fact that the disparity map does not change under these conditions. Anyway, our experimental setting showed a disparity map not stable enough (especially in the case of network-attached cameras) to be employed for this task.

In order to add time adaptivity to our background modeling technique, we have to continuously update the background model. As the background model is basically made up by averages and variances, we need a way to calculate them in such a way that oldest measurements slowly become non influential. Let w be a component of the background model (i.e., the color channels R,G,B,H,S,V of the left frame, the brightness and chromaticity distortions BD and CD), we define the concept of *running* average $\mu_t.w$ and variance $\sigma_t^2.w$ as follows:

$$\mu_t(x,y).w = \eta_t(x,y)w_t(x,y) + (1-\eta_t(x,y))\mu_{t-1}(x,y).w \tag{2.6}$$

$$\sigma_t^2(x,y).w = \eta_t(x,y)\left(w_t(x,y) - \mu_t(x,y).w\right)^2 + (1-\eta_t(x,y))\sigma_{t-1}^2(x,y).w \tag{2.7}$$

The novelty factor $\eta_t(x,y)$, common to all the statistical measures and calculated frame by frame, has to be tuned following the probability that a given pixel is not belonging to a mobile agent. In particular, it will be high for pixel belonging to objects to be acquired in the scene and zero for those belonging to mobile agents.

Time adaptivity has been reached by applying the method introduced in [24] as a tuning mechanism for η_t. The approach requires the maintenance of a second part of the background model, namely the *edge intensity model*, which stores the average edge intensity. Given the vertical V_t and the horizontal H_t border matrices computed by applying a Sobel filter to l_t, the value of the current frame edge intensity matrix E_t for pixel (x,y) is obtained as $E_t(x,y) = \sqrt{V_t^2(x,y) + H_t^2(x,y)}$. The edge intensity model consists of the running average $\mu_t.E$ over E_t where $\eta_t(x,y) = \lambda \ \forall(x,y)$ with the constant λ usually set to 0.2 in order to quickly react to changes. An example of edge model $\mu_t.E$ is shown in Figure 2.19.

The update term for the color model is obtained by the product of vertical and horizontal activities for a given pixel, $A_{t,vert}$ and $A_{t,horz}$ respectively, defined as follows:

$$A_{t,horz}(y) = \sum_X |E_t(X,y) - \mu_t(X,y).E| \quad A_{t,vert}(x) = \sum_Y |E_t(x,Y) - \mu_t(x,Y).E| \tag{2.8}$$

Figure 2.19. The edge model relative to the situation depicted in Figure 2.18.

The value of $\eta_t(x, y)$ for the color model is calculated on a per-pixel basis and is inversely proportional to the pixel activity. If a pixel shows a high activity value, then its background model will not be updated; conversely, if the pixel activity is low, then the background model will be updated with a learning factor dependent from the activity of the pixel. The reason why this definition of pixel activity is efficient is that, even in a still position, a person always shows moving edges due to breathing. A pixel with a moving edge is propagated by Equation 2.8 to the whole row and column that the pixel belongs to. In this way, not-animated objects are acquired into the background while persons and moving objects are not.

2.8.2 Tracking and Pose Estimation

Tracking consists of associating, at time t, a detected object $d_b^t \in \mathcal{D}_t$ to a tracked object $o_a^{t-1} \in \mathcal{T}_{t-1}$. The state $\mathcal{S}(o_a^{t-1})$ of the tracked object is made up, in PLaTHEA, by its *foreseen* position, velocity and color template at time t.

For a given tracked/detected object, the color template projects each of the pixels that map into its blob to a *chrominance* couple $\langle c_1, c_2 \rangle$, obtained from an *opponent color space* [161], where c_1 and c_2 serve as coordinates (through normalization) of a k by k matrix (k is set to 64 in PLaTHEA) storing in each position the counter of pixels mapping there. Supposing that o_a^{t-1} and d_b^t represent the very same agent, their difference is defined as follows:

$$\mathcal{S}(o_a^{t-1}) - d_b^t = \langle d_p, d_a, c \rangle \tag{2.9}$$

where *(i)* $d_p \in \mathbb{R}$ is the euclidean distance between the detected position of d_b^t and that foreseen by the Kalman filter for o_a^{t-1} (thus taking into account the velocity the tracked agent is moving at), *(ii)* $d_a \in \mathbb{R}$ is the angle (in radiants) between the velocity previously assigned to o_a^{t-1} and that of d_b^t as if it represents the very same agent of o_a^{t-1}, and *(iii)* $c \in \mathbb{R}$ is a measure of the difference between the color templates of d_b^t and o_a^{t-1} calculated by summing up over the differences between each of the k^2 positions.

At each time step, Equation 2.9 is calculated for every possible couple $\langle o_a^{t-1}, d_b^t \rangle$. The most likely couples are selected and the states in $\mathcal{T}_{t-1}$ are updated accordingly and added to $\mathcal{T}_t$. Brand new tracked objects are created in $\mathcal{T}_t$ for those detected objects for which no association has been found.

A classic approach [95, 24] makes use of a score function $\mathcal{F}$ in order to calculate for each couple $\langle o_a^{t-1}, d_b^t \rangle$ an affinity score. As couples correspondent to the very same agent will show the best scores, $\mathcal{F}$ can be employed to solve the association

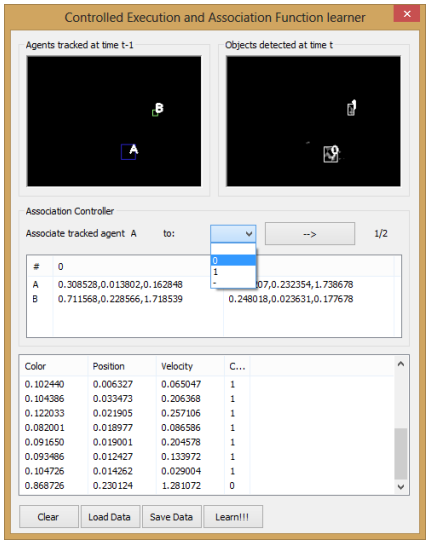


Figure 2.20. The facility used to learn the association function $\tilde{\mathcal{F}}$.

problem. Function $\mathcal{F}$ is usually a linear combination of distance measures whose weights are empirically set.

PLaTHEA implements a novel approach based on support vector machines (SVM). The data set, or experience, $\mathcal{E}$ of the learning task consists of a number of tracking task executions where associations are manually done. The operation in PLaTHEA has been made simple by using a functionality integrated into the graphic front-end (see Figure 2.20). At each frame the user is asked to select for each tracked agent the corresponding detected agent or nothing if the tracked agent has not been detected in the current frame. Remarkably, as we have chosen a set of features that are independent from both the indoor space size and the light, the learning task has to be performed only once without the need of repeating it for every installation of the system.

For each couple $\langle o_a^{t-1}, d_b^t \rangle$ and for each time stamp t, we define a training example $\langle \langle d_p, d_a, c \rangle, t \rangle$, where $r \in \{-1, 1\}$ is equal to 1 if and only if o_a^{t-1} and d_b^t represents the same mobile agent. So our data set $\mathcal{E}$ can be defined as follows:

$$\mathcal{E} = \{\langle \langle d_{p0}, d_{a0}, c_0 \rangle, r_0 \rangle, \ldots, \langle \langle d_{pN}, d_{aN}, c_N \rangle, r_N \rangle\} = \{\langle f_0, r_0 \rangle, \ldots, \langle f_N, r_N \rangle\} \tag{2.10}$$

where the components of $f_i \, \forall i$ are normalized in order to be all comprised between 0 and 1, in order to ensure that none of them excessively influences the machine learning result.

The function to be learned is the *association function* $\tilde{\mathcal{F}}_t : \mathcal{D}_t \times \mathcal{T}_{t-1} \to \{-1, 1\}$ that approximates $\mathcal{F}_t$. Given the nature of the input, $\tilde{\mathcal{F}}_t$ is a *continuous binary classification* function, and thus representable as a separation hyperplane $\tilde{\mathcal{F}}_t = sgn\left(\vec{w}\phi\left(f_t\right) + b\right)$ where the weight vector $\vec{w}$ and the coefficient b are obtained by applying the SVM algorithm for *soft margin hyperplane*, which is resistant to training data noise, whereas ϕ is a basis function allowing to transform the training examples' space into a more effective data space with respect to separability.

We chose a radial basis function (RBF) as ϕ in order to map the input space (which is probably not separable) into a very high dimensional space (which is very

likely separable). In addition, the real training is preceded by a *grid search* with cross-validation (obtained by splitting the data set into separated training set and validation set) in order to find the best parameters for SVM learning. It should be noted here that a single agent can be associated to many tracked objects while time passes, but can not be associated to multiple objects at a specific time instant.

This system allows to reach a very good performance (measured on an entire different data set) with respect to both false positives (0.008%) and false negatives (0.048%). False negatives are more dangerous than false positive, as to each false negative corresponds the creation of a new tracked object that makes it impossible to seamlessly follow a subject. In addition, the percentage of false negatives becomes higher in presence of partial occlusions, which are very frequent in every day life.

A tracked object o_a^{t-1} is associated to a detected object d_b^t if and only if the following condition holds (ts denote the creation time of a tracked object):

$$\begin{gathered} \tilde{\mathcal{F}}(d_b^t, o_a^{t-1}) = 1 \\ \wedge \\ \nexists\ d_k^t : \tilde{\mathcal{F}}(d_k^t, o_a^{t-1}) = 1 \\ \wedge \\ \nexists\ o_j^{t-1} : \left(\tilde{\mathcal{F}}(d_b^t, o_j^{t-1}) = 1 \wedge ts(o_j^{t-1}) < ts(o_a^{t-1})\right) \end{gathered} \tag{2.11}$$

In the vast majority of the cases, these trajectories have only to be smoothed by detecting outliers and noise in measurements. Authors in [35] employ different trackers depending on how much crowded the scene is. For lowly crowded scenes, a Single-Hypothesis Tracker (SHT) together with color features is used for tracking an agent. For highly crowded scenes, a Multiple-Hypothesis Tracker (MHT) is used for each agent.

PLaTHEA employs MHT to improve tracking in case a subject is lost due to occlusions or vision artifacts. While a tracked object is in the tracked stated, a single Kalman filter is employed to follow the subject. This filter supposes the agent moving at a constant vectorial velocity. Whenever PLaTHEA loses a subject, it continues trying to associate it to detected objects following two hypothesis. The first one supposes constant velocity while the second one supposes zero velocity. The first hypothesis copes with the possibility that a subject hidden behind another one is continuing to walk in the same direction. The second hypothesis supposes the subject is still behind a fixed object (an agent or a part of the environment). Whenever an association is found, the original filter is updated.

Chapter 3

The Quest for Datasets

Smart houses typically generate datasets as sequences of measurements generated by installed sensors. The main disadvantage of this is that gathering a critical mass of data requires a big amount of time; additionally the dataset cannot be augmented adding new sensors without a physical installation and a new recording session.

In this chapter we will first introduce a technique for generating a sensor log starting from a set of hand made models of human habits. This technique merge declarative languages for process modeling with planning. This technique represents the first step towards a platform for gathering datasets of human activities inside a smart space through crowdsourcing and gamification.

In the second part of the chapter first a serious game is introduced aimed at gathering from real users sensor log of their daily activities by providing a player character inside a virtual environment. Finally a strategy to mix the two approaches is provided.

3.1 Dataset Generation

In this section we propose a method for generating synthetic datasets for experimentation by *simulating* (virtual) smart spaces. As an example application of such a dataset, we will focus on the problem of recognition of *activities of daily life* (ADLs). Habit recognition can be performed over *logs* resulting from a set of installed sensors (door state detectors, light switches, movement sensors, item presence sensors, etc.). Given a set of ADL models (previously discovered, either automatically or manually) and the log resulting from the installed sensors, a habit recognition algorithm is able to recognize the ADLs users are performing into the environment. Nonetheless, as we explained earlier, only a limited number of such datasets are available.

To this aim, our tool starts from a set of *habit templates* and periodically assigns some of them to be performed by a given agent. From this assignment a random *trace* is generated as a list of actions to be performed by the agent, possibly interleaving the different habits he is pursuing. These are essentially high-level actions, called *h-actions*, each of which corresponds to a goal to be realized according to a lower-level underlying action theory. In the next step, a planner is employed that produces a sequence of low-level atomic *p-actions* for each *h*-action in the trace, generating a more detailed *planner log*. All these actions are finally executed by a virtual agent

into a 3D environment populated with a customizable set of sensors that trigger while the agent is acting. In such a way, from the planner log it is possible to produce a *sensor log*, which associates to each *p-action* a set of sensor measurements. The proposed tool can be customized in order to produce logs of a user-defined length.

3.1.1 Preliminaries

People's behavior is typically studied in terms of *habits* or *activities of daily life (ADLs)*. A habit is essentially a loosely specified sequence of high-level actions that are related to each other and aim at pursuing a goal (e.g. cleaning the house). The way a habit is performed may portray a high degree of variability between different users or even between the same user in different time frames. Due to this variability, tasks belonging to a habit can be modeled using precedence relations rather than strong sequential assumptions. This characteristic of habits makes them suitable to be described using *declarative* process modeling formalisms (such as DECLARE [156]) rather than *procedural* ones. A distinctive feature of declarative models is that there exist a lot of different execution logs which satisfy a certain template, as they are based on the principle that "all that is not explicitly forbidden is allowed". This is different point of view with respect to procedural models, which assume that "all that is not explicitly allowed is forbidden".

In order to generate a dataset that represents habit realizations by people, one cannot rely solely on the enactment of a habit model, as a habit does not provide information about how the tasks are actually realized. Habit templates essentially refer to high-level events that may be executed in different ways depending on the circumstances. As far as low-level execution is concerned, an action theory can be used as a fine-grain representation of the environment where tasks composing habits may be pursued using planning.

Declarative Modeling Languages and DECLARE

Instead of rigidly defining the flow of interaction, DECLARE [156] focuses on the (minimal) set of rules which must be satisfied in order to correctly carry out a process (flow of tasks). A DECLARE model is defined as a triple $\mathcal{CM} = \langle T, \mathcal{C}_m, \mathcal{C}_o \rangle$, where *(i)* T is a set of tasks, represented as boxes containing their name, *(ii)* $\mathcal{C}_m$ is a set of mandatory constraints, and *(iii)* $\mathcal{C}_o$ is a set of optional constraints..

DECLARE constraints, represented using arrows which connect task boxes and annotations, are grouped into four families: *(i)* existence constraints are unary cardinality constraints expressing how many times an activity can/should be executed; *(ii)* choice constraints are n-ary constraints expressing the necessity to execute some activities between a set of possible choices, independently from the rest of the model; *(iii)* relation constraints are binary constraints which impose the presence of a certain activity when some other activity is performed, possibly imposing also temporal constraints on such a presence; *(iv)* negation constraints are the negative version of relation constraints, and are employed to explicitly forbid the execution of a certain activity when some other activity is performed.

The semantics of DECLARE constraints has been originally grounded into LTL (Linear Temporal Logic) formulas. Temporal logics are a special class of modal

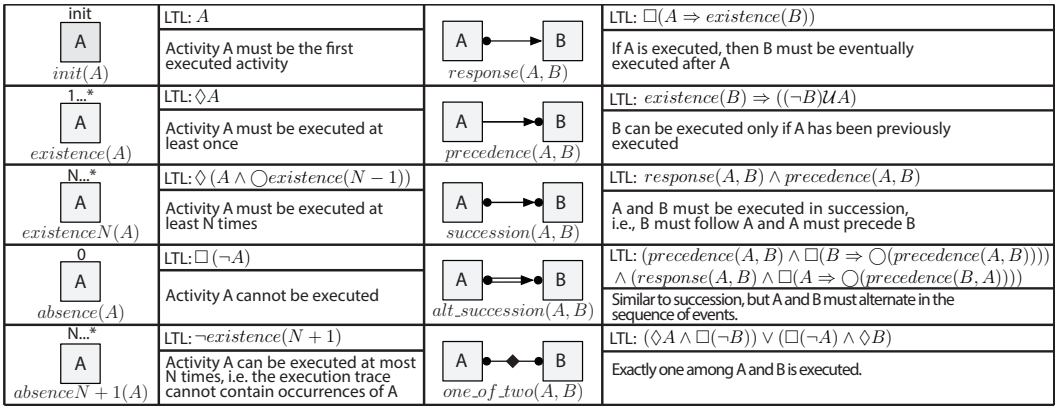


Figure 3.1. A selection of DECLARE constraints

logics where modalities are interpreted as temporal operators, used to describe and reason about how the truth values of assertions vary over time. In this class, LTL can be considered as being: *(i)* propositional, as formulas are defined from atomic propositions, whose truth values change over time; *(ii)* linear, as temporal operators predicate on the truth of propositions along a single timeline and each moment has a single next future moment; *(iii)* qualitative, as temporal operators are used to express qualitative time relations between propositions; *(iv)* point-based, as temporal operators and propositions are evaluated over points in time; *(v)* time discrete, as the next moment corresponds to the immediate successor state induced by the occurrence of an event; *(vi)* future-tense, as temporal operators predicate on the occurrence of events in the future. LTL formulae are defined using atomic propositions (with true and false constants), propositional connectives ($\neg$, $\wedge$, $\vee$, $\Rightarrow$), and temporal operators ($\bigcirc$ next time, $\Box$ globally, $\Diamond$ eventually, $\mathcal{U}$ until).

The interpretation of LTL is defined over infinite traces, which is inconsistent with the execution of a process that always generates a finite trace of tasks. In order to enact DECLARE process models, the same authors propose to employ a interpretation over finite traces of LTL called LTL_f [159].

Figure 3.1 shows all the graphical DECLARE elements which are used throughout this chapter together with their correspondent LTL formulas. We can add to the DECLARE language new constraints, as well as we can associate a graphical symbol to it, and we can also provide an LTL semantics for it. Starting from the single constraints of the model, the LTL formula representing the DECLARE model is obtained by the logical AND of the formulas associated to the single constraints.

Recently [74] proposed to ground the semantics of declarative constraints into regular expressions because *(i)* they are natively built to express finite sequences, *(ii)* LTL has been proved to have the same expressive power as star-free regular expressions and First-Order Logic (FOL) over finite ordered traces [76, 199, 135, 83], and *(iii)* regular expressions have the same expressive power as finite state automata (that are equivalent to monadic second order logic over finite ordered traces) [41, 190]. As a consequence, regular expressions are more powerful than LTL and allow to express more powerful declarative constraints (e.g., a certain task always occurs in couples). For this reason, in the following we will directly refer to regular expressions

Constraint	Regular expression	Example
Existence constraints		
$Existence(n, \mathtt{a})$	`[^a]*(a[^a]*){n,}+[^a]*`	
$Participation(\mathtt{a}) \equiv Existence(1, \mathtt{a})$	`[^a]*(a[^a]*)+[^a]*`	bcaac
$Absence(m+1, \mathtt{a})$	`[^a]*(a[^a]*){0,m}+[^a]*`	
$Uniqueness(\mathtt{a}) \equiv Absence(2, \mathtt{a})$	`[^a]*(a)?[^a]*`	bcac
$Init(\mathtt{a})$	`a.*`	accbbbaba
$End(\mathtt{a})$	`.*a`	bcaaccbbbaba
Relation constraints		
$RespondedExistence(\mathtt{a}, \mathtt{b})$	`[^a]*((a.*b.*)|(b.*a.*))*[^a]*`	bcaaccbbbaba
$Response(\mathtt{a}, \mathtt{b})$	`[^a]*(a.*b)*[^a]*`	bcaaccbbbab
$AlternateResponse(\mathtt{a}, \mathtt{b})$	`[^a]*(a[^a]*b[^a]*)*[^a]*`	bcaccbbbab
$ChainResponse(\mathtt{a}, \mathtt{b})$	`[^a]*(ab[^a]*)*[^a]*`	bcabbbab
$Precedence(\mathtt{a}, \mathtt{b})$	`[^b]*(a.*b)*[^b]*`	caaccbbbaba
$AlternatePrecedence(\mathtt{a}, \mathtt{b})$	`[^b]*(a[^b]*b[^b]*)*[^b]*`	caaccbaba
$ChainPrecedence(\mathtt{a}, \mathtt{b})$	`[^b]*(ab[^b]*)*[^b]*`	cababa
$CoExistence(\mathtt{a}, \mathtt{b})$	`[^ab]*((a.*b.*)|(b.*a.*))*[^ab]*`	bcaccbbbaba
$Succession(\mathtt{a}, \mathtt{b})$	`[^ab]*(a.*b)*[^ab]*`	caaccbbbab
$AlternateSuccession(\mathtt{a}, \mathtt{b})$	`[^ab]*(a[^ab]*b[^ab]*)*[^ab]*`	caccbab
$ChainSuccession(\mathtt{a}, \mathtt{b})$	`[^ab]*(ab[^ab]*)*[^ab]*`	cabab
Negative relation constraints		
$NotChainSuccession(\mathtt{a}, \mathtt{b})$	`[^a]*(aa*[^ab][^a]*)*([^a]*|a)`	bcaaccbbbba
$NotSuccession(\mathtt{a}, \mathtt{b})$	`[^a]*(a[^b]*)*[^ab]*`	bcaacca
$NotCoExistence(\mathtt{a}, \mathtt{b})$	`[^ab]*((a[^b]*)|(b[^a]*))?`	caacca

Table 3.1. Semantics of Declare constraints expressed using regular expressions

instead of LTL_f; the translation of declarative constraints into regular expression is reported in Table 3.1 taken from [74].

STRIPS Planning

In STRIPS planning [81] one is faced with the following task. Given *(i)* a complete specification of the initial state of the world, *(ii)* a set of actions that describe how the world may change, and *(iii)* a goal condition, one has to find a sequence of actions such that if they are executed sequentially starting from the initial state, checking for corresponding preconditions, and applying the effects of each action one after the other, they lead to a state that satisfies the goal condition.

In STRIPS, the representation of the initial state, the actions, and the goal condition is based on first-order logic literals.

The initial state is specified as a set of positive ground literals which give a complete specification of the world based on a closed-world assumption.

An action is specified in term of its preconditions and effects, also expressed as sets of literals. In the case of preconditions a set of positive literals specify what conditions need to be true in order for the action to be executable. Similarly for the effects of an action, a set of positive and negative literals specify how the state should be transformed when the action is performed: positive literals in the set of effects are added in the set describing the state and negative literals are removed.

A goal condition is also a set of positive ground literals and it is satisfied in a state if all the literals listed in the goal condition are included in the set that describes the state.

In this work we solve STRIPS planning problems using a Planning Domain Definition Language (PDDL) [82] compliant planner. In particular, our tool makes use

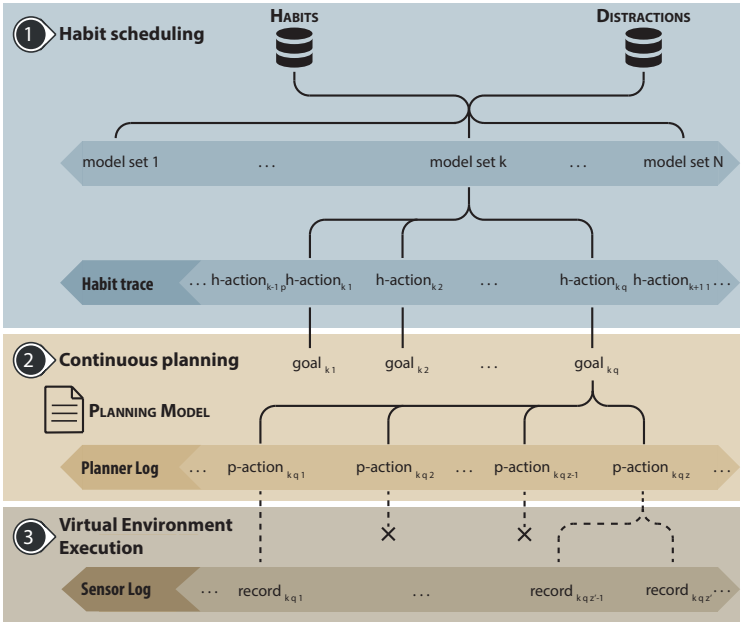


Figure 3.2. The sensor log generation strategy.

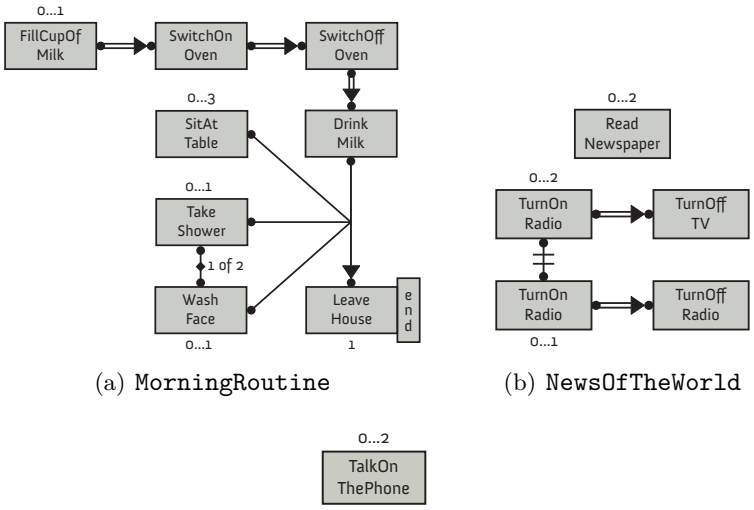
of the JavaFF [57] planning engine, which is an implementation of the FastForward (FF) heuristic search planning algorithm. In PDDL, the specification of a planning task (planning model) is separated into two parts, the *domain description* that lists the available predicates and actions, and the *problem description* that specifies the initial and goal state.

3.1.2 Dataset Generation Strategy

We propose a layered dataset generation strategy which employs techniques from business process modeling and from classical planning in order to drive characters to perform realistic sequences of operations inside a virtual environment and collect synthetic sensor measurements.

Figure 3.2 summarizes the strategy employed for dataset generation. Our tool takes as input two repositories describing habit and distraction templates in the DECLARE formalism, and a PDDL file that models the smart home as a planning domain. A designer, through an appropriate interface (textual or graphical) is in charge of defining them. Habit templates relate to achieving an articulate goal in the smart space, e.g. cook dinner, while distraction templates model casual behaviors, e.g., turn-on TV. The planning domain provides lower level actions that may be used to realize the high-level actions of the DECLARE templates.

In the first stage an *interleaved habit trace* is produced. At each step, a number of habit and distraction templates are chosen, and a random instance of their interleaving is extracted. The number of habits and distractions to be combined in each iteration as well as the number of repetitions (that influence the length of the generated dataset) can be set as parameters of our tool. The resulting log specifies an interleaved sequence of high-level actions that we call *h-actions* based on the given templates.



(a) `MorningRoutine` (b) `NewsOfTheWorld`

(c) `TalkingOnThePhone`

Figure 3.3. Example habit and distraction models.

In the second stage, our tool takes as additional input a PDDL planning model that specifies the virtual environment, and produces a *planner log* in the following way. For each h-action in the interleaved habit trace, a planning problem is instantiated and solved, having as a goal to realize the h-action. The solution generates a sequence of atomic *p-actions*, which are executed updating the planning environment.

In the final stage all the *p-actions* contained in the planner log are executed into a 3D virtual simulation environment. This environment is populated with a customizable set of sensors that monitor the environment. Each *p-actions* may trigger (none or multiple) sensor measurements in the sensor log.

3.1.3 Habit Modeling and Scheduling

Figures 3.3(a), 3.3(b) and 3.3(c) depict DECLARE templates describing two daily habits and one distraction. For example, the `MorningRoutine` habit template specifies that the agent either takes a shower or wash his face (modeled using a branching `succession` and a `not_coexistence` DECLARE constraints).

DECLARE templates representing habits and distractions impose constraints about the execution order of composing h-actions. Each DECLARE template can be translated into a set of regular expressions to be respected by compliant traces. As a consequence, it is also possible to combine different templates by the union of their representative regular expressions, hence allowing for the execution of more habits/distractions in an interleaved way.

At each step, our tool selects a configurable number of habit and distraction templates following a probability function. This function associates to each habit template the probability of being executed at a certain time of the day. In essence, the trace generation is performed by our tool by first translating the set of selected templates into a set of regular expressions and then by randomly executing the corresponding finite state automata over the subset of h-actions (considered as

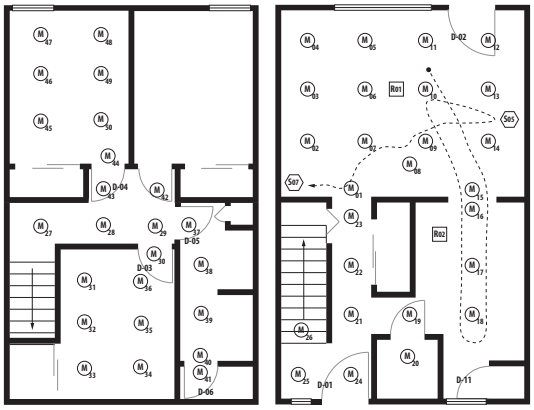


Figure 3.4. The virtual smart space with deployed sensors.

an alphabet) contained in the original DECLARE template. If no traces can be generated we can infer that habits and/or distractions are incompatible and a new set of templates is randomly selected.

As an example, an execution trace compliant to the combination of the templates previously introduced is the following:

Example 3.1.
`(1) LeaveBed (2) FillCupOfMilk (3) TurnOnRadio (4) ReadNewspaper (5) TurnOffRadio (6) TalkOnThePhone (7) SittingKitchen (8) StartOven (9) ReadNewspaper (10) StopOven (11) DrinkMilk (12) TakeShower (13) LeaveHouse`

This sequence of h-actions represents a high level strategy employed by the virtual agent to execute his habits.

3.1.4 Continuous Planning

Each of the h-actions in the habit trace is treated as a goal that the agent needs to achieve in the context of a planning model reproducing a smart home.

The PDDL domain specifies four different types of objects: `room`, `device`, `state`. The `device` type abstracts objects that the agent can interact with in the smart home such as *doors*, *windows*, electronic devices and household objects. Some devices are *stateful*, having a current `state` object associated (e.g., either `closed` or `open` for a door device) stored into the `deviceState` predicate.

The arrangement of the smart home is fixed and specified using the predicates `(adjacent ?r1 - room ?r2 - room ?d - device)` and `(deviceAt ?d - device ?r - room)` with the intuitive meaning. Devices used in the context of the `adjacent` predicate as passages between rooms are specified by a `waypoint` predicate. The topology used in our examples and during the evaluation, depicted in Figure 3.4, is inspired by the Kyoto setup of CASAS project; additionally the planning problem defines a wide set of devices that, for the sake of readability, are not shown.

The basic predicates affected by actions are the following:

- `(characterAt ?r - room ?d - device)` specifies the device the agent is next to;

- `(deviceState ?d - device ?st - state)` specifies, for each stateful device, the current state;
- `(usedDevice ?d - device)` is intended to denote that a device has been used by the agent in the scope of a *h-action* execution.

The available actions model how the agent moves in the environment (`moveToRoom` and `moveToDevice`) and how it interacts with devices (`changeDeviceState` and `useDevice`). The specification of the `moveToRoom` follows:

```
(:action moveToRoom
:parameters (?r1 - room ?d1 - device ?r2 - room ?w - device)
:precondition (and (characterAt ?r1 ?d1) (waypoint ?w)
 (adjacent ?r1 ?r2 ?w) (deviceState ?w open)
:effect (and (not (characterAt ?r1 ?d1)) (characterAt ?r2 null)))
```

Similarly, `moveToDevice` allows the character to move from one device to another belonging to the same room.

The interaction with devices can be performed either by changing the state of a device by means of a `changeDeviceState` action, or by a `useDevice` action that represents the execution of an atomic complex task over the device. The effect of the latter is to change the `usedDevice` predicate, which is reset after each planner execution to allow reuse of devices.

From h-actions to planning problems and p-actions

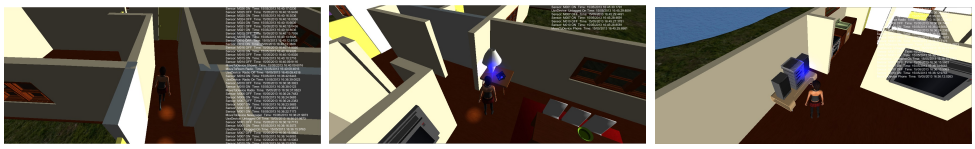
In order to produce a planner log as part of our data generation methodology, our tool continuously tries to realize new goals following the h-actions in the interleaved habit tracer. For each h-action, a new planning problem is instantiated and solved. The resulting action sequence is then reported and *executed* in the domain so that the current state is *updated* according to it.

The goals for the planning problems need to be specified using predicates available in the planning domain, so a correspondence between h-actions and goals is required. An extended description of the DECLARE input is assumed by our tool, in which for each h-action one or more goal conditions are also specified using the predicates of the PDDL domain. For example the template in Figure 3.3(a) is annotated to specify that h-action `LeaveHouse` may be realized by pursuing the goal `(and (deviceState entranceDoor closed) (characterAt outside null))`.

The level of detail by which a h-action will be transformed into a goal and eventually to a sequence of p-actions, depends on the degree of detail of the virtual environment expressed as a planning domain. Note also that the PDDL representation provides a principled way for specifying a wide range of different configurations and allows for automated methods for re-configuring and exploring the space of design options for smart spaces.

3.1.5 Example

Suppose that the character, currently in the living room close to the newspaper, wants to fill a cup of milk; the planner may produce the following plan:



(a) Movement (b) Phone (c) Radio

Figure 3.5. Example sensors

Example 3.2.

```
GOAL deviceState cupOfMilk filled
(1) (moveToDevice livingroom newspaper kitchenDoor)
(2) (changeDeviceState livingroom kitchenDoor closed open)
(3) (moveToRoom livingroom kitchenDoor kitchen kitchenDoor)
(4) (moveToDevice kitchen null cupOfMilk)
(5) (changeDeviceState kitchen cupOfMilk empty filled)
```

After executing the previous plan, the state of the environment is changed: the character is not in the living room anymore, the door of the kitchen is open and the cup is full of milk. Suppose now the character after turning on the radio, answers the phone. The planner will start from the current state of the world and will compute:

Example 3.3.

```
GOAL deviceState radio on
(1) (moveToRoom kitchen cupOfMilk livingroom kitchenDoor)
(2) (moveToDevice livingroom null radio)
(3) (changeDeviceState livingroom radio off on)
GOAL usedDevice livingroom phone
(1) (moveToDevice livingroom radio phone)
(2) (useDevice livingroom phone)
```

The continuous execution of the planner produces a planner log which contains p-actions. Post-processing removes all the p-actions that do not come from sensors.

3.1.6 Virtual Environment Execution

The planning environment introduced in the previous section can be easily reproduced into a rapid development environment for video-games such as Unity[1]. As a consequence, the p-actions contained into the planning log can be executed by a *non-player* character (NPC), thus obtaining a 3D simulation environment.

Beside all the devices defined into the planning problem, the virtual 3D environment contains a customizable set of *sensors* monitoring the house and providing measurements. Each sensor is in charge of detecting and reporting the timestamp of a particular event. Figure 3.4 shows the set of sensors employed during our tests. As an example, PIR - Presence InfraRed sensors are frequently deployed into real smart houses; they trigger whenever a person is inside their detection cone and can be easily modeled into Unity as colliders. If these kind of sensors are mounted on the ceiling following a grid layout, the triggering sequence can capture the trajectory a person is following into the environment.

Sensors in our Unity virtual environment are completely decoupled from devices and rooms coming from the planning problem. Sets of sensors differing for number

[1] http://unity3d.com

```
20-05-13 08:06:24:5232 M010 ON
20-05-13 08:06:25:1766 M010 OFF
20-05-13 08:06:26:5278 M015 ON
20-05-13 08:06:30:5666 M015 OFF
20-05-13 08:06:30:6832 M016 ON
20-05-13 08:06:31:2484 M016 OFF
20-05-13 08:06:46:6807 M017 ON
20-05-13 08:06:47:2657 M017 OFF
20-05-13 08:06:47:6974 M018 ON
20-05-13 08:06:48:1249 M018 OFF
20-05-13 08:06:48:3299 M018 ON
20-05-13 08:06:52:0331 M018 OFF
20-05-13 08:06:52:5234 M017 ON
20-05-13 08:06:53:0781 M017 OFF
20-05-13 08:06:53:5392 M016 ON
20-05-13 08:06:54:0743 M016 OFF
20-05-13 08:06:54:2236 M015 ON
20-05-13 08:06:54:7582 M015 OFF
20-05-13 08:06:56:1208 M010 ON
20-05-13 08:06:56:6013 M010 OFF
20-05-13 08:06:57:9044 M009 ON
20-05-13 08:06:58:5216 M009 OFF
20-05-13 08:06:59:3668 M007 ON
20-05-13 08:06:59:7478 M007 OFF
20-05-13 08:07:00:4976 M001 ON
20-05-13 08:08:32:1452 M001 OFF
```

Figure 3.6. Example Sensor Log 1

```
20-05-13 08:06:25:4872 R001 OFF
20-05-13 08:06:30:4773 R002 ON
20-05-13 08:06:47:5623 R002 OFF
20-05-13 08:06:52:2981 R002 OFF
20-05-13 08:06:54:8979 R001 ON
20-05-13 08:06:57:1171 S005 ON
20-05-13 08:07:00:3328 S007 ON
20-05-13 08:08:32:3230 S007 OFF
```

Figure 3.7. Example Sensor Log 2

and types, as further discussed in the following, give different insights about ADLs performed into the environment. As a consequence, none or more sensor log entries may correspond to a single *p-action*. The virtual 3D environment allows to obtain customized datasets by adding and removing sensors. Additionally, sensors can be customized aiming at representing the real world at the desired level, for example by integrating physics and introducing white noise.

Moreover, the virtual environment allows us to leave timing issues outside the habit and planning layer. The time that executing a *p-action* takes to be executed depends on the way the agent and the devices are modeled into the virtual environment.

3.1.7 Evaluation

In order to illustrate how different sensor configurations influence the obtained dataset, we first look into two sensor logs generated using our tool. Both logs are generated starting from the very same planner log (shown in the previous example) but using different set of sensors.

The first log (see Figure 3.6) is obtained by placing into the virtual environment a grid of fine grain movement sensors (e.g., M001, M010) that trigger while the virtual agent is moving following a trajectory (shown in Figure 3.4). The second log (see Figure 3.7) is obtained by equipping the environment *(i)* presence sensors in every room (e.g., R001, R002) and *(ii)* switch sensors attached to both the radio and the phone (S005 for the radio and S007 for the phone). It is easy to see from these examples that different sensor configurations provide a completely different point of view w.r.t. habit recognition.

The use of the same habit templates and planning domain over different sets of sensors, enables the evaluation of techniques for smart homes under different assumptions. In order to illustrate the type of analysis, we use a number of datasets generated with our tool as input to the state-of-the-art habit recognition (HR)

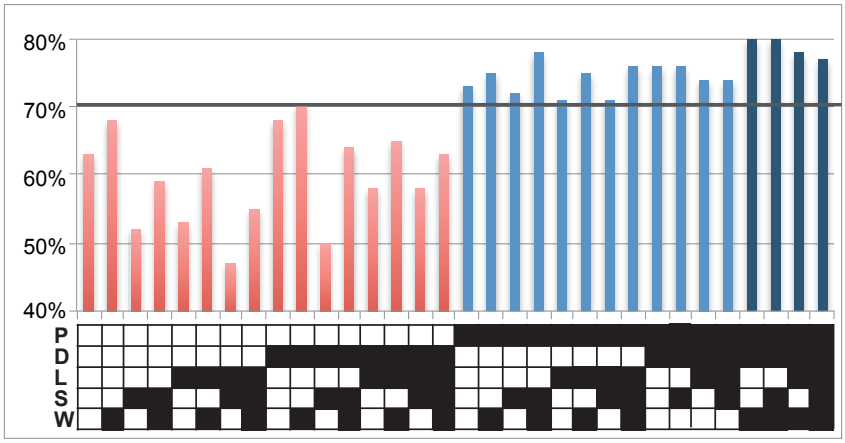


Figure 3.8. CASAS activity recognition performance.

tool developed by the CASAS project [59] (freely available at the project website). CASAS tool allows to perform activity recognition using various techniques. In our analysis we used a Naive Bayes Classifier (similar results are obtained by using the other provided methods). Also, the DECLARE templates used consisted of 10 habits and 5 distractions inspired by the work of [114] in ADLs.

In order to perform a comparison over sensor configurations, we kept the planner log fixed and varied the type of sensors deployed in the smart home through the planning environment. The configurations were obtained by generating all combinations of classes of generic sensors, denoted as follows: P - Presence InfraRed (PIRs), D - door, L - light, S - shutter and W - window. A small subset of specific sensors to appliances, such as the heating control, radio, TV, vacuum cleaner and oven, were included in all the configurations.

Figure 3.8 shows the performance of the the HR tool for each configuration (represented w.r.t. black checkboxes). The number reported for each category is the average recognition rate over all habits with respect to the ground truth that is provided by our tool:

- Red bars in the histogram refer to configurations without PIR sensors while blue bars refer to those including PIR sensors. By comparing the two categories we observe that PIR sensors definitely improve the performance of habit recognition producing more accurate results (minimum 71% of accuracy) and a consistently better performance for all categories that include them.
- It is clear that the presence of window sensors always improves performance.
- The last four bars lead to the best accuracy rates reached with the simultaneous presence of PIRs, door and window sensors, with the P/D/W configuration being a minimal set of sensors that achieves best results.

3.1.8 Related Work and Limitations

Generic tools for generating datasets of agents moving into pervasive environments are presented in [97] and [139]. The former tool generates a dataset based on a

transition matrix representing the probability-causal relations between all the sensors; as a consequence, the generated sensor log is tightly coupled to the model. The latter tool (based on Repast Symphony simulation environment[2]) aims at the massive generation of big datasets, but no evaluation over whatsoever technique for smart spaces has been – to the best of our knowledge – presented so far.

In [134] a large scale simulator for ubiquitous computing called Siafu is proposed. Here agents are modeled using simple state machines, the world using images and coordinates of points of interest, and statistical models are employed so simulate environmental variable behavior.

The DiaSim simulator in [40] allows a very fine grain specification of a smart environment. This description is used to generate both a programming framework to develop the simulation logic and an emulation layer to execute applications. Furthermore, a simulation renderer is coupled to DiaSim to allow a simulated pervasive system to be visually monitored and debugged.

A customized dataset generation strategy is presented in [138], which is employed to evaluate and optimize performance of a multimodal identification and tracking algorithm.

Recently the UbiREAL[3] [149] prototype has been released (end of September 2012); this is conceived as a framework for building virtual smart spaces, but it does not provide any way to generate datasets. Another example is DIVAS[4] [16], a plugin-based multi-agent 3D simulation framework focusing on large scale events.

Currently, the approach we described is used to generate sensing data related to a single character that moves and acts in the smart space. Real smart spaces, however, may feature multiple persons at the same time each one performing his own habits. This adds an additional complication to processes that analyze sensing data, as most of the widely used sensors generate measurements that are not correlated to a particular person at the time of sensing. As a consequence it is important that the system is able to generate a dataset depicting the activities of multiple agents. From the point of view of our methodology this represents a challenge because agents can execute habits concurrently and this has to be modeled in a way that a realistic dataset can be generated.

3.2 Gamification and Crowdsourcing

Previous section introduced a methodology to automatically generate datasets starting from hand-made models of human habits expressed using a declarative process modeling language. The methodology employs a virtual environment populated with non-player characters. The same kind of data can be gathered by populating the virtual environment with *player* characters controlled by human users. The idea here is to provide the environment to real people as a *serious game* [207, 10]. In such a way, a human user can "play" a portion of his daily life (possibly by giving incentives to reproduce his everyday activities or act in hypothetical scenarios) and our tool can record such traces. This way, *crowdsourcing* can be employed to obtain

[2] http://repast.sourceforge.net/
[3] http://ubireal.org/
[4] http://mavs.utdallas.edu/projects/divas/

(a) Exploration (b) Mission Menu (c) Interaction

Figure 3.9. Screenshots from the serious game.

a large amount of data that also exhibit a higher degree of variability as the set of involved humans is heterogeneous.

A direct comparison should be done with games like *The Sims*: a major difference here lies on the fact that The Sims does not intend to model extremely realistic situations and the agents act following their own fitness function instead of a structured schedule. The Sims game implements a series of rewards that are not related to the likelihood of characters behavior; instead they are related to (short and long term) wishes and goals that provide points to be used to obtain objects or to change the state of the character.

Even though rewards provided to the user can be kept, creating a serious game aimed at gathering real human habit traces requires to change the point accumulation mechanism in order ensure the *truthfulness* of the reported behaviors. In [107, 108] a rewarding mechanism based on active consensus, where users are asked to evaluate other users behavior or answers. Such kind of consensus mechanism is employed in many *task markets* (e.g., Mechanical Turk[5]) and other popular services such as Stack Overflow[6]. As active consensus may result tedious to users, automatic consensus mechanisms should be introduced.

Our approach consists of providing users with a smartphone game, which allows them to deal with different missions that require to perform a certain habit in the virtual environment. While performing a habit the user receive points for interacting with the environment. As an example, Figure 3.9(a) depicts the player character gathering points while it is exploring the environment.

Developing a smartphone game has the advantage of fostering sporadic use with users that open the application only for playing briefly. Depending on the time of the day, the user will be prompted with a menu asking to chose a specific habit to perform (see Figure 3.9(b)). During a mission, the user will interact through the player character with a set of appliances. Figure 3.9(c) shows the character approaching the fridge; at this point the character can take with him some of the

[5] http://www.mturk.com/
[6] http://stackoverflow.com/

objects contained in the fridge in order to move them to another recipient object (e.g., the oven).

Once the user completed a habit (i.e., the chosen mission), it cannot be ensured that it completed the habit in a truthful way. Additionally, the interaction with the objects provides points independently from their utility in the considered habit as specifying it manually would be expensive in terms of human resources. An automatic consensus algorithm can be implemented at this point by employing a bonus/malus mechanism.

Periodically the system gathers all the repetitions of the same habit from multiple users and provide a bonus or a malus depending on how many times each specific action has been performed by the other users. Other bonuses can be given by the system to those users who decide to perform habits for which few repetitions have been gathered.

Future works on the system will include the possibility of performing habits involving multiple persons by supporting multiple users at the same time.

3.3 Future Work: Mixing through Mining

Part of our future work is to couple this type of automated behaviors (see Section 3.1) with user-generated behaviors (see Section 3.2) inside the simulated environment.

The main drawback of the dataset generation tool introduced in Section 3.1 is that declarative models of habits are hand-made. Once the tool described in 3.2 is available, and hence real usage data can be obtained, the habit models can be automatically mined as we will see in Chapter 4.

Once habit templates are mined, non player characters can be embedded into the serious game in order to give the environment an additional degree of variability. Real people may be, at this point, instructed to perform daily activities while virtual agents driven by our scheduling strategy interact with them executing daily activities of their own.

Chapter 4

Mining Smart Spaces

Systems for smart environments aim at recognizing needs and wishes of users, and at automatically controlling the cyber-physical space in order to satisfy them. However, in order to perform correct actuation actions, i.e., to change the environment in a desired way, the smart system must have sufficient information: the expected reactions to certain situations, the environmental constraints that must be satisfied, the expected goals to be accomplished, together with the services (i.e., available actions) to accomplish such goals, etc. We called knowledge such information and models its constituting components.

Models may be entered manually into the system, or be based on pre-generated templates and domain knowledge. The manual creation of models may be a viable choice for newly constructed smart systems; the manual addition of such models is implementable, through appropriate user interfaces [75], in smart homes, where such models inherently have high degree of legitimacy and correctness, as they by-creation represent the end-user desires. However, the efforts to create models manually are in most cases much larger than what users are willing to spend, even if pre-defined model templates are taken into account. In larger environments, e.g., offices, smart residences for elderly persons, etc., in which the amount of models is huge, such efforts are overwhelming and it is not guaranteed that manual insertion will not contain internal contradictions, leading to unsatisfying situations.

Therefore, a promising approach is to obtain the models through (fully- or semi-)automated learning, i.e., by collecting previous observations and inferring the rules from them.

In this chapter we first introduce a technique based on itemset mining to mine constraints in smart spaces. These constraints can be employed to produce automatic rules that triggers whenever a non conventional context is detected. In the second part of the chapter the topic of applying process mining to human habits is tackled.

4.1 Mining Constraints for Decision Making

In this section, we propose a novel technique to mine rules by using a variation of the Apriori algorithm [9]. We mine rules in the form of environmental constraints, i.e., we aim at finding restricted combinations of environmental values. In order to find them, we aim at detecting a sudden abnormal drop in frequency in logs of a certain

variable combination w.r.t. its subsets. Such an abnormal drop may represent the fact that this set of values is undesirable, and the respective constraint should be created. Such mined constraints can be then directly used in reasoning systems, e.g., similar to the one described in [69, 68], which are based on constraint satisfaction principles.

4.1.1 Preliminaries

Let $V = \{v_1, \ldots, v_n\}$ be a set of variables, each of them varying over a finite domain $D^{v_i} = \{d_1^{v_i}, \ldots, d_{n_i}^{v_i}\}$, describing the current state of the smart environment. Variables in V can be classified as either uncontrollable or controllable; uncontrollable variables are bound to *sensors* whereas controllable ones are bound to *actuators*. We assume values for both categories of variables to be accessible (at least during the mining phase); if for some actuators the assigned values cannot be directly acquired, we assume the availability of a set of sensors providing enough information to infer their current states (in such case a corresponding *virtual* sensor may be added).

Following [69], the expected behavior of the smart space automation system is defined as a set of rules, which must be satisfied at any particular instant of time. The rules are defined as predicate logic formulas where atoms take the form $v_i = d_k^{v_i}$ or $v_i \neq d_k^{v_i}$, with $d_k^{v_i} \in D^{v_i}$; well-formed formulas are obtained by composing atoms with logical connectives and quantifiers, such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, and $\Leftrightarrow$. Rules represent dependencies between variables and can be easily represented in the form of either allowed or forbidden assignments.

Example 4.1. *As an example, the reader should consider a rule* $Jack.location = room1 \Rightarrow room1.lamp = on$*; the assignments allowed by this rule for the variables* $(Jack.location, room1.lamp)$ *are the following:* $(\neg room1, on), (room1, on), (\neg room1, off)$*. The very same rule can be written in terms of its forbidden assignment* $\neg(Jack.location = room1 \wedge room1.lamp \neq on)$*.*

It is easy to notice that representing rules as restrictions (or *constraints*) requires much less explicit statements, than if the rules were represented as possibilities, as environments are usually much more *permissive* than *restrictive*. Therefore we opted to mine rules in the form of *constraints* over the environment.

Definition 1.3 and 1.4 can be easily adapted to the case of discrete, instead of binary, variables as follows:

Definition 4.1 (Item, Itemset, Snapshot)**.** *The assignment* $v_i = d_k^{v_i}$ *of a single variable to a single value of its domain will be referred to as an* item*. An* itemset C *is a combination of items such that no variable is seen more than once in it, i.e.,* $C = \{v_i = d_k^{v_i} \mid \forall i \neq j : v_i \neq v_j\}$*. A* snapshot *(we will use this term instead of transaction as it is more appropriate to the contained information)* S *is an itemset assigning a value to all the variables in* V*.*

A snapshot describes the state the smart space in a given instant of time $t(S)$. Constraints, i.e., forbidden variable assignments, are returned by the technique presented in Section 4.1.3 in the form of itemsets.

The input to our algorithm is a weighted snapshot dataset defined as follows:

Definition 4.2 (Weighted Snapshot Dataset)**.** *A weighted snapshot dataset* $\mathcal{T} = \{< S_1, w_1 > \cdots < S_{|\mathcal{T}|}, w_{|\mathcal{T}|} >\}$[1] *is obtained by generating, every time the state of a single variable in* V *changes, a new couple* $< S, w >$*, where* S *is the current snapshot and* w *is a weight assigned to it. The weight* w *is a real number, which defines how much* S *is significant inside the dataset.*

When new snapshots are not added to the dataset at a regular pace, being them added after the arrival of a new measurement, they should be accounted proportionally to the time $\Delta(S)$ they remain valid as current state of the system. If two snapshots S_j and S_{j+1} are added one after the other to $\mathcal{T}$, then $\Delta(S_j) = t(S_{j+1}) - t(S_j)$. Following this intuition, the weight is calculated as follows:

$$w = \frac{\Delta(S)}{\alpha} \text{ with } \alpha = \frac{t(S_{|\mathcal{T}|}) - t(S_1)}{|\mathcal{T}| - 1} \tag{4.1}$$

Conversely, if new snapshots are added at a fixed pace, w can be set to an arbitrary real value β (e.g., during our evaluation over the real dataset we used $\beta = 1$); in this case two consecutive snapshots may be equal in terms of variable values as they represent the very same environment state.

Given an itemset C and a dataset $\mathcal{T}$, we define $\mathcal{T}^C = \{< S_i, w_i > \in \mathcal{T} \mid C \subseteq S_i, i \in [1, |\mathcal{T}|]\}$ as the set of couples in $\mathcal{T}$ whose snapshots contain C. The weight of C inside $\mathcal{T}$ is defined as follows:

$$w(\mathcal{T}^C) = \sum_{<S_i, w_i> \in \mathcal{T}^C} w_i \tag{4.2}$$

If $w_i = \beta$ for every couple in $\mathcal{T}$, the definition reduces to $w(\mathcal{T}^C) = \beta|\mathcal{T}^C|$.

Frequent Itemsets Mining

The mining technique proposed here is a variant of the seminal Apriori algorithm [9] employed for mining *frequent* itemsets. Even though the original formulation mines itemsets of binary variables as those in Definition 1.4, the Apriori algorithm can be straightforwardly extended to the case of itemsets assigning values to discrete variables as those in Definition 4.1. We define a frequent itemset as follows:

Definition 4.3 (Frequent Itemset)**.** *An itemset* C *assigning values to a set of discrete variables is told to be frequent if its support is above a minimum threshold value.*

Algorithm 4.1 contains a listing of the Apriori algorithm. At the k-th step the algorithm constructs a set L_k containing all the k-itemsets (i.e., itemsets containing exactly k items) with enough support by adding a single item to the itemsets non discarded at step $k - 1$.

The APriori algorithm can be easily adapted to a weighted snapshot dataset by simply replacing the Definition 4.3 with the following one:

[1] Given a set A, we denote with $|A|$ the cardinality of A.

Algorithm 4.1: Apriori algorithm

```
1: function findFrequentItemsets (T - dataset; r - minimum support)
2: L_1 ← {1-itemsets S s.t. Supp_T(S) > r}
3: for (k ← 2; L_{k-1} ≠ ∅; k++) do
4:    C_k ← {a ∪ {b} | a ∈ L_{k-1} ∧ b ∈ ⋃L_{k-1} ∧ b ∉ a}
5:    for all S ∈ T do
6:       C_t ← {c ∈ C_k | c ⊆ S}
7:       for all c ∈ C_t do
8:          c.count++
9:       end for
10:   end for
11:   L_k = {c ∈ C_k | c.count ≥ minsup}
12: end for
13: return ⋃_k L_k
```

Definition 4.4 (Weighted Frequent Itemset)**.** *Given an itemset C, its* weighted support *in $\mathcal{T}$ is defined as the fraction $WSupp_{\mathcal{T}}(C) = \frac{w(\mathcal{T}^C)}{W}$, where $W = \sum_{i=1}^{|\mathcal{T}|} w_i$ is the sum over the entire dataset $\mathcal{T}$ of the weights w_i. An itemset C is told to be frequent if its weighted support is above a minimum threshold value.*

Definition 4.4 reduces to Definition 4.3 when $w_i = 1\ \forall i$, i.e., when the transactions are not weighted.

4.1.2 Restricted Itemsets

Many variations to the original Apriori algorithm have been proposed in the literature in order to meet the requirements of particular applications. In some cases, the discovery of *rare itemsets*, instead of frequent ones, may be particularly useful. If, in the ideal case, an itemset correspondent to a forbidden assignment should never happen, in practice, this is not the case, and forbidden combinations still occur due to erroneous sensor readings, non-immediate reaction of users and deviations from the standard behavior. An itemset is usually told to be rare if it has a very low support. Unfortunately, fixing a very low support threshold for the Apriori algorithm is not enough to find rare itemsets, as it would generate a huge number of redundant and unnecessary itemsets; this is known as the *rare itemset problem*. Additionally the definition of rare itemset is application specific. In [204], a rare itemset is defined as "one which its frequency in the database does not satisfy the minimum support but appears associated with the specific data in high proportion of its frequency". In [188], a rare itemset is somewhat *interesting* only if it represents an exception (e.g., there are many vegetarians and many persons with cardiovascular diseases, but very few people are both); thus, the final goal of the proposed Apriori-Rare algorithm is to compute *minimal rare itemsets* (mRI), which are those itemsets that are rare but whose proper subsets are frequent.

Other works, e.g., [53, 112], point out that it may be more interesting to find combinations of items that are not necessarily frequent, i.e., they may have low *support* value, but a high *confidence* rate, i.e., high probability of being together

in the same transaction. This consideration is applicable in our scenario. As an example, the reader should consider a situation where a person goes to the bathroom, opens a bathroom cabinet, takes a toothbrush, turns on the water tap, etc. The amount of minutes people spend doing this activity is very low, comparing to the full log of the smart space, thus the itemset describing this situation has low support and cannot be regarded as "frequent". However, it can be said with very high confidence that if a person is in the bathroom, and the toothbrush is taken, then the water tap is always turned on. A constraint to describe the above case could be $\neg(Jack.location = bathroom \wedge Jack.toothbrush = taken \wedge bathroom.watertap = off)$.

Our definition of "interesting" is somewhat orthogonal to the above mentioned ones. Instead of fixing a threshold in order to distinguish frequent from rare itemsets, a dynamic threshold is computed for each itemset aimed at detecting a significant drop in its weight with respect to its subsets; we will refer to such itemsets as *restricted*.

It can be trivially shown that the support of an itemset is always smaller or equal than the support of each of its proper subsets. We formalize this drop by using the concept of *decline*:

Definition 4.5 (Decline)**.** *Given two itemsets C and C_p, where C is obtained by adding a single item to C_p, i.e., by assigning a value to a variable v_i not appearing in C_p (C_p is told to be* parent *of C), we define the* decline *of C with respect to C_p as* $\frac{w(\mathcal{T}^C)}{w(\mathcal{T}^{C_p})} \leq 1$.

Decline is a natural phenomenon as we are constraining the variable v_i to assume a specific value $d_j^{v_i} \in D^{v_i}$. In order to detect the amount of decline that is abnormal, we can define the *normalized decline* as follows:

$$\frac{w(\mathcal{T}^C)}{p(d_j^{v_i})w(\mathcal{T}^{C_p})} \leq 1 \tag{4.3}$$

Here $p(d_j^{v_i})$ represents the probability of a variable v_i to be in a state $d_j^{v_i}$. We need this probability, as many variables in smart environments have non-uniform distribution. For example, during a working day, a PC has much bigger chance to be turned on, than being off.

Specifically, for variables with uniform distribution $p(d_j^{v_i}) = \frac{1}{|D^{v_i}|}$. For a variable showing a non-uniform distribution, instead, if $\mathcal{T}^{d_j^{v_i}} = \{< S_i, w_i > \in \mathcal{T} \mid (v_i = d_j^{v_i}) \in S_i, i \in [1, |\mathcal{T}|]\}$, we have:

$$p(d_j^{v_i}) = \frac{w(\mathcal{T}^{d_j^{v_i}})}{W} \text{ where } w(\mathcal{T}^{d_j^{v_i}}) = \sum_{<S_i,w_i> \in \mathcal{T}^{d_j^{v_i}}} w_i \tag{4.4}$$

If the normalized decline of an itemset C is close to 1, this means that the combination of variable assignment it represents is perfectly admissible for the final users; an *abnormal* decline (with much smaller closeness values) may be an indicator that C is an undesired, restricted, combination. As, in general, the very same itemset C can be obtained by adding a variable assignment to a set $\mathcal{C}_p$ of different parent itemsets, we define a restricted itemset as follows:

Algorithm 4.2: Find restricted variable sets

function findConstraints ($\mathcal{T}$ - dataset; r - restriction rate)
for all $d_j^{v_i} \in D^{v_1} \cup \cdots \cup D^{v_n}$ **do**
  $p(d_j^{v_i}) \leftarrow \frac{|v_i = d_j^{v_i}|}{|\mathcal{T}|}$
end for
$\mathcal{R} \leftarrow \{\text{1-itemsets } S \text{ s.t. } \frac{w(\mathcal{T}^S)}{|D^{v_i}|*|\mathcal{T}|} < r\}$
$\mathcal{I} \leftarrow \{\text{all 1-itemsets}\} \backslash \mathcal{R}$
while $|\mathcal{I}| > 0$ **do**
  $\mathcal{C} \leftarrow getSetOfChildren(\mathcal{I})$
  $\mathcal{I} \leftarrow \emptyset$
  for all $C \in \mathcal{C}$ **do**
    $W_P \leftarrow \emptyset$
    for all $P \in parents(C)$ **do**
      $d_j^{v_i} \leftarrow addedValue(P, C)$
      $W_P \leftarrow W_P \cup \{w(\mathcal{T}^P)p(d_j^{v_i})\}$
    end for
    if $\frac{w(\mathcal{T}^C)}{min(W_p)} < r$
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{P\}$
    else if $w(\mathcal{T}^C) > 0$
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{P\}$
    end if
  end for
end while
return $\mathcal{R}$

Definition 4.6 (Restricted Itemset)**.** *An itemset C is told to be restricted if its normalized decline with respect to each of the parent itemsets in $\mathcal{C}_p$ is less than a predefined threshold $r \in [0, 1]$ called "restriction rate".*

Example 4.2. *As example, assume we have a small environment with three boolean variables: PC with weights $w(PC = 0) = 80$, $w(PC = 1) = 137$; LCD with weights $w(LCD = 0) = 145$, $w(LCD = 1) = 72$, and chair PR[essure] with weights $w(PR = 0) = 92$, $w(PR = 1) = 125$. The weights of combined itemsets $w(\{PC = 1; LCD = 0\}) = 65$ and $w(\{PC = 1; LCD = 1\}) = 72$ both represent normal decline rate, and therefore are considered permitted. On the other hand, the weight of itemset $w(\{PC = 0; LCD = 1\}) = 0$ clearly indicates that such itemset is restricted and will generate a constraint $\neg(PC = 0 \wedge LCD = 1)$, which represents physical constraint that LCD cannot be turned on if the PC is off. Another itemset $w(\{LCD = 1; PR = 0\}) = 2$ is also restricted, because the decline rate from both parents (with weights 72 and 92) is still large enough, even though the situation happened for a couple of minutes. This constraint shows that there is a big preference to turn off a monitor if no-one is sitting in front of the PC.*

4.1.3 Constraint Mining Algorithm Description

Our approach to mine constraints in a smart environment is listed in Algorithm 4.2. The algorithm takes as input a dataset $\mathcal{T}$ and the desired *restriction rate* r, and it returns as output a set $\mathcal{R}$ of restricted itemsets representing constraints. As a first step, the distribution p of domain values is calculated (lines 2-4).

Coherently to the original Apriori algorithm, at every iteration, the algorithm processes itemsets with increasing size stored in the set $\mathcal{C}$. At the k-th iteration, $\mathcal{C}$ contains only k-itemsets (itemsets containing exactly k items) obtained by combining (using the $getSetOfChildren$ function at line 8) those $(k-1)$-itemsets, contained in the set $\mathcal{I}$, that was not declared as restricted at the $(k-1)$-th iteration. Given a k-itemset C, its parents are those $(k-1)$-itemsets contained in $\mathcal{I}$ such that all their items are contained in C. Thus, every itemset has a number of parents equal to its size.

Example 4.3. *As an example, suppose* $\mathcal{I} = \{\{v_1 = d^{v_1}, v_2 = d^{v_2}\}, \{v_1 = d^{v_1}, v_3 = d^{v_3}\}, \{v_2 = d^{v_2}, v_3 = d^{v_3}\}\}$*, then* $getSetOfChildren(\mathcal{I}) = \{\{v_1 = d^{v_1}, v_2 = d^{v_2}, v_3 = d^{v_3}\}\}$.

Before iterations begin, $\mathcal{R}$ is initialized with all restricted 1-itemsets, and $\mathcal{I}$ is initialized with all the possible 1-itemsets, except those that are restricted. We regard 1-itemset as restricted if its closeness to the uniformly distributed values is less that the restriction rate.

During each iteration the set $\mathcal{I}$ is immediately emptied and will be filled with the itemsets to be expanded during the successive iteration; the algorithm stops as soon as, after the conclusion of an iteration, $\mathcal{I}$ is empty (see line 7).

During a specific iteration, the weight of each itemset $C \in \mathcal{C}$, is compared to a dynamically computed threshold dependent on the value of r. Restriction rate represents the maximum decline value for considering an itemset as restricted. As a C has, in general, more than a single parent, the abnormal decline should be registered for all its parents in order to declare it as restricted. To this aim, it is sufficient to compare r with the maximum value of decline calculated (see line 16).

At every iteration, all the itemsets in $\mathcal{C}$ that satisfy the condition at line 16 are put into the set $\mathcal{R}$ of constraints (see line 17). Itemsets in $\mathcal{C}$ that do not satisfy the same condition are put into the set $\mathcal{I}$ (see line 19), unless their weight is equal to 0, which means that the corresponding combination of values is never seen in the dataset, so there is no point in trying to expand it further. This happens when the decline difference of all itemset's parents never exceeded the restriction rate, which means decline was always within expected limits until finally reaching 0.

4.1.4 Validation

We assessed the performance of the proposed algorithm by running different experiments on both synthetic and real datasets. Tests on the synthetic dataset highlight how much the accuracy changes by tuning different parameters of the algorithm. The evaluation on the real dataset shows the effectiveness of the algorithm in a realistic application scenario.

During the evaluation, accuracy results represent the similarity of original rules and mined ones by constructing the full truth table for all n variables, and counting the combinations of sensor values in the table returning the same result ("allowed"

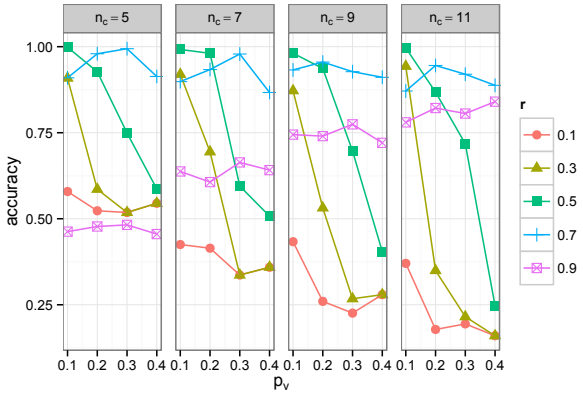


Figure 4.1. Accuracy of the algorithm in terms of correctly interpreted snapshots.

or "restricted") for both original and mined rules.

Validation on Synthetic Datasets

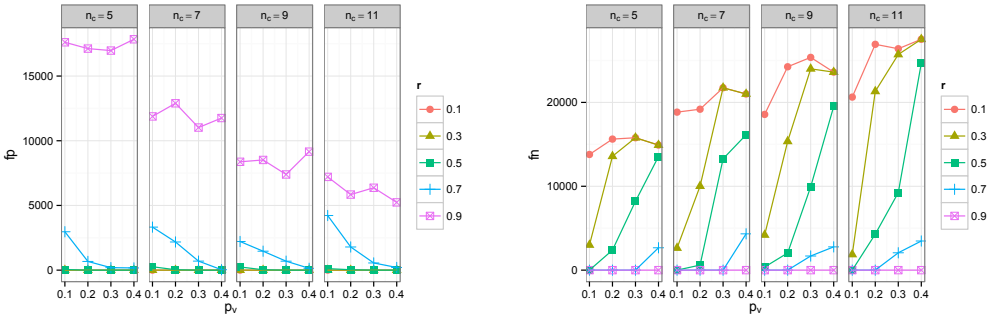
As it is very difficult to obtain datasets large enough to evaluate algorithms in a wide range of input configurations, a first quantitative evaluation has been conducted on a configurable synthetic dataset.

For these experiments we used a generation tool producing a dataset $\mathcal{T}$ consisting of $|\mathcal{T}|$ snapshots. A snapshot binds n boolean variables; however, it is always possible to transform a discrete variable into a set of boolean variables. As previously discussed, each of the produced snapshots is supposed to have an associated weight; as the generation tool generates snapshots at a constant pace, the weight is always equal to $\beta = 1$. A parameter p_c represents the probability that at each discrete time step the state of the environment changes; p_c reflects how much the simulated environment is dynamic.

The synthetic dataset is built by trying to respect a randomly generated set of n_c constraints; each constraint binds up to 4 variables, as this is a reasonable limitation for a real environment. Violations to rules are accepted with a probability p_v, which represents how much the inhabitants of the simulated environment are prone to deviate from their usual behavior (the one we aim to mine). Finally a parameter p_p represents the probability that a violation is maintained after a single step; after k steps this probability becomes p_p^k.

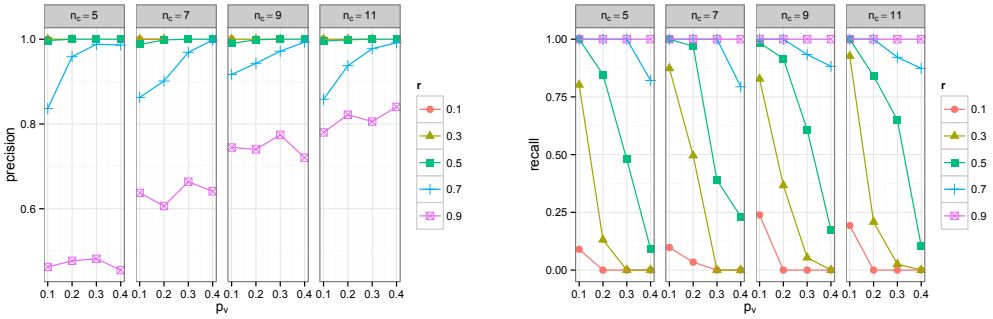
At every step with probability p_c a change is applied to a variable; supposing this change introduces n_v new violations and solves n_s previously introduced violations, the probability of adding this change to the dataset is equal to $p_v^{n_v} \cdot \prod_{i=1}^{n_s}(1 - p_p^{k_i})$, where k_i denotes the number of steps since i-th violation was introduced.

Figure 4.1 and Figure 4.2 represents the results obtained by applying our algorithm to different significant parameters of both the algorithm and the generation tool. Each facet represents the performance of the algorithm with a different number n_c of constraints. The performance of the algorithm, with respect to the probability p_v of adding a violation, is calculated for five different values of the r parameter, namely: 0.1, 0.3, 0.5, 0.7 and 0.9. Each single condition has been repeated 20 times



(a) False positives (fp)

(b) False negatives (fn)



(c) Precision = $\frac{tp}{tp+fp}$

(d) Recall = $\frac{tp}{tp+fn}$

Figure 4.2. Analysis of the errors introduced by the algorithm.

using different values of the p_p parameter[2]. All the conditions generate a number of snapshots $|\mathcal{T}| = 10000$ by keeping the probability p_c of changing the environment fixed to 0.8 and the number n of variables fixed to 15.

Parameter r tells the algorithm how much the actual decline should differ from the expected one in order to declare a constraint as restricted. Figure 4.1 shows how for low values of p_v, putting $r = 0.5$ provides the best results, with an accuracy greater than 75% when p_v is less than 0.2. On the other hand, the higher is the value of p_v, the higher will be the number of violations inside the dataset, thus, the higher will be the weight of the constraint inside the dataset; as a consequence, in order to declare an itemset as restricted we need r to have a high value.

Figure 4.2 thoroughly inspects how the errors committed by the algorithm are distributed. It can be noticed how, except for high values of the parameter r, the algorithm provides very good performances in terms of false positives, intended as snapshots allowed in the original dataset that are forbidden by the mined constraints; this means that the proposed algorithm does not introduce unwanted restrictions. From the point of view of false negatives, intended as snapshots forbidden in the original dataset that are allowed by the mined constraints, the performances of the algorithm quickly get worse for small values of r as either p_v or n_c increase. As a consequence of the previous considerations, small values of r increases the precision of the algorithm; conversely, high values of r increases the recall provided by the algorithm.

For small values of r, the mined rules are very close to the original set and this makes them useful if we aim at mining original constraints for enacting the smart environment. On the other hand, by increasing the value of r, the algorithm will overfit the original data by producing a set of rules with a high cardinality that is very different from the set of constraints employed to generate the dataset. Nevertheless rules extracted with high values of r can be still employed for detecting unusual conditions of the environment without automatically perform any modification to the environment.

Even though results get worse as either p_v or n_c increase, putting $r = 0.5$ provides, overall the best results. Moreover, it is likely to expect that rules that the users want to be respected are not violated more than 15% of the time.

Living Lab Case Study

To evaluate the algorithm on real dataset, we used the dataset obtained from previous experiments for a small living lab. Here we briefly explain the design of the experiment and the results.

The experiment was performed on the premises of the University of Groningen, where two identical working rooms were populated with sensors and sensor data was collected for three days. The original setup is described in details in [148]. Each room contains a desk with PC, LCD, two motion (PIRK, PIRM), two acoustic (AC, ACK), and two chair pressure (PR1, PR2) sensors.

The goal of the original experiment was to use the rules of the environment to find and resolve sensor errors to improve activity recognition (AR) rate. Therefore

[2]During our evaluation we noticed that p_p has a small influence on the achieved accuracy.

Table 4.1. Manual vs Mined Constraints (shown as itemsets)

Original	Mined
LCD=T PC=F	PC=F
LCD=T PR1=F PR2=F	AC=T PR2=T
LCD=T PIRK=F	ACK=F PR2=T
PIRK=T PR1=F PR2=F	LCD=T PIRK=F
AC=T ACK=F	PIRK=F PR1=F PR2=T
AC=T PIRM=F	ACK=T PIRK=T PIRM=F PR1=F PR2=F
AC=F PR1=T PR2=T	AC=T ACK=T PIRK=F PR1=T

the rules of environment were written manually. In this experiment we used the original sensor data (before sensor errors correction, as it was using manually written rules) and applied our algorithm to it to mine the rules automatically. We then used those mined rules and compared them with the original manually created ones. Note that original sensor data was quite noisy, with activity recognition rate from the original data being 80.9% in the first room and 76.3% in the second one. Sensor readings were collected once every minute during working hours. Three days of data from two different rooms with identical setup were combined to obtain a log of 2402 snapshots in total. Among those snapshots 199 (8.3%) minutes represented the full absence of people in a room; 344 (14.3%) snapshots represented presence of people in a room, but not in front of a PC or sitting at working desks; during 70 (2.9%) minutes in total meetings were held; 395 (16.4%) snapshots represented different types of paperwork, i.e. people working at a desk, but not with PCs; and 1394 (58.0%) minutes were dedicated to working with PCs. Note that several sensor value combinations can sometimes represent the same type of situation, for example working with PC may happen with background motion sensor showing motion or not.

The original and mined constraints are shown in Table 4.1. There are several interesting points to mention. For example, the algorithm found the constraint $\neg(PC = F)$. This constraint was not part of the manual rules. However, all data collection was done during working hours, and during all that time both PCs in both rooms were always turned on. That means that the found constraint represents the preference rule "During the working day PC should be turned on", which follows directly from actual users behavior, and therefore in some sense is more representative of the actual situation than manually written rules were. Some constraints, as $\neg(LCD = T \wedge PIRK = F)$ are contained in both the original and mined sets, while others, as $\neg(PIRK = T \wedge PR1 = F \wedge PR2 = F)$, are found as a part of more restrictive constraints, namely $\neg(ACK = T \wedge PIRK = T \wedge PIRM = F \wedge PR1 = F \wedge PR2 = F)$. The accuracy results were that for 256 combinations of the truth table, 185 were the same (72.3%), while 71 were different. Noteworthy, the obtained accuracy is comparable to that reported in [20], where Event-Condition-Action rules in smart environments are mined.

The less regular human behavior is the bigger the input dataset needs to be in order to not integrate spurious behavior into the results. As in our experimental setup users were instructed to execute routines in a predefined way, three days

of recordings are enough in order to obtain good results. In order to perform experiments over a not specifically designed real dataset, the size of the dataset needs to be selected in order to represent the average behavior of the users.

4.1.5 Related Work

Data mining techniques have been already employed in smart environments in order to infer human habits or unusual (e.g., dangerous) situations. Authors in [129] applies (point-wise) inter-transaction association rules (IAR) mining in order to detect *emerging* (i.e., unusual) behaviors; the first step consists of mining IAR rules from sensor logs, whereas emerging ones are identified, as a second step, during runtime by comparison. The real world data consisted of event logs from an array of state-change sensors. The SPUBS system [23] merges concepts from both data mining and workflow mining. The proposed technique is mainly supervised and supposes the availability of a huge amount of labeled data. The CASAS project employs a pattern mining technique [168, 61] in order to discover human activity patterns. This method allows to discover *discontinuous patterns* by iteratively trying to compress the dataset. A pattern is defined as a set of events where order is not specified and events can be repeated multiple times. The data mining method employed for activity discovery is completely unsupervised without the need of manually segmenting the dataset or choosing window sizes.

All the aforementioned works differ from our technique as they all are *event-based* and mainly focused on the runtime recognition of activities. Conversely, we concentrate on discovering unusual states of the environment in order to generate *reactive rules* that are triggered in the case mined constraints are violated. From this point of view, the final goal is similar to that of the APUBS [20] system. APUBS mines simple Event-Condition-Action (ECA) rules, where an ECA rule takes the form "ON *event* IF *condition* THEN *action*". Still, the approaches are different and complementary, as while APUBS inherits from SPUBS a workflow-based approach, we do not impose any order between state changes. This allows us to mine human habits from a different point of view.

Example 4.4. *Suppose, as an example, that users follow a safety rule that says that the oven and the hair-dryer are never on at the same time in order to avoid electrical blackouts; our algorithm will mine in this case the constraint* $\neg(OVEN = ON \wedge HAIRDRYER = ON)$. *The APUBS algorithm employs the Apriori algorithm for finding the event-action relationships, hence it will find ECA rules related to this constraints only if there is enough support for the event-action couples where sensor values are opposite (e.g.,* $\langle OVEN = ON, HAIRDRYER = OFF\rangle$ *and* $\langle OVEN = OFF, HAIRDRYER = ON\rangle$*). Anyway, these event-action couples will be found only if, during a window of events, it happens, often enough to exceed the required support, that the two events are both present. As the chosen window size is usually small in order to avoid to connect unrelated events, none of these event-action couples will be likely discovered.*

4.2 Extracting Habits Using Process Mining

The term *process* denotes the sequence of activities performed by human or non-living beings to yield a result. Business process management is based on the observation that each product that a company provides to the market is the outcome of a number of activities performed. Nowadays, business processes are the key instrument to organize these activities and improve the understanding of their interrelationships.

As shown [23], learning a habit is very similar to mine a business process. Yet, due to the nature of smart spaces, some considerations need to be done in order before applying process mining to this context.

As a first consideration, human habits are flexible in their nature. In the vast majority of cases, humans do not follow a precise workflow while performing their daily activities; they instead follow a series of best practices that are difficult to be described using a precise sequence of tasks. For this reason, we believe that declarative models, which specify a set of constraints among the instances of tasks inside a process, are more suitable than classical imperative models (e.g., workflows), which instead specify possible compliant traces in a very restrictive way, to model human habits (see Section 4.2.1).

As a second point, as noted in [26], despite the growing availability and maturity of process mining techniques, their applicability in some contexts still faces the problem of bridging the gap between events given as input and activities composing the models obtained as output. This problem particularly applies to the challenge of mining human habits using process mining techniques as here sensor/event log contains very fine grained examples whereas mined model should contain higher level activities.

In this section we outline a method to overcome these problems in order to apply process mining to the problem of habit mining. Solving this problem will by the way close the dataset generation loop we introduced in Chapter 3.

4.2.1 Preliminaries

Process Mining [3], also known as *Workflow Mining* [2], is the set of techniques that allow the extraction of structured process descriptions, stemming from a set of recorded real executions. Such executions are intended to be stored in so called *event log*s, i.e., textual representations of a temporarily ordered linear sequence of tasks. There, each recorded *event* reports the execution of a *task* (i.e., a well-defined step in the workflow) in a *case* (i.e., a workflow instance). Events are always recorded sequentially, even though tasks could be executed in parallel: it is up to the algorithm to infer the actual structure of the workflow that they are traces of, identifying the causal dependencies between tasks (*condition*s). ProM [4] is one of the most used plug-in based software environment for implementing workflow mining techniques.

Most of classical process management environments for the specification of processes are based on imperative languages and procedural models (such as that employed in [23]), such as YAWL nets, WS-BPEL and BPMN. Procedural process models explicitly define the control-flow of activities, i.e., the schedule and ordering of tasks to be executed, and process executions are driven by the flow of control prescribed by the corresponding models. As a result, many possible execution options

and potentially allowed paths not explicitly included in the control-flow are not admitted, as they would violate the strict constraints imposed by the models. On the one hand, an imperative and procedural modeling approach is well suited for processes where tasks need to be repeatedly executed in a predefined and controllable manner, and where all possible interactions among involved entities (people, devices, services, etc.) can be captured and specified in detail by the modeling capabilities and semantics of procedural languages. On the other side, procedural models are not able to provide the degree of flexibility required in many other settings and they may unnecessarily limit possible execution behaviors. When trying to define a process using a procedural model, the designer is forced to either capture all possible allowed executions in an attempt to enhance flexibility, or to make some a priori assumptions and choose (and thus impose at design-time) a restricted subset of allowed executions. As a consequence, produced models may become either over-specified and thus complex to understand and manage, or over-constrained, as "all that is not explicitly modeled is forbidden".

The idea to apply process mining in the context of workflow management systems was introduced in [8]. There, processes were modeled as directed graphs where vertices represented individual activities and edges stood for dependencies between them. Cook and Wolf, at the same time, investigated similar issues in the context of software engineering processes. In [62] they described three methods for process discovery: *(i)* neural network-based, *(ii)* purely algorithmic, and *(iii)* adopting a Markovian approach. The authors considered the latter two as the most promising. The purely algorithmic approach built a finite state machine where states were fused if their *futures* (in terms of possible behaviors for the next k steps) were identical. The Markovian approach used a mixture of algorithmic and statistical methods and is able to deal with noise. Although, the results presented in [62] were limited to sequential behavior only.

From [8] onwards, many techniques have been proposed, in order to address specific issues: pure algorithmic (e.g., α algorithm, drawn in [6] and its evolution α^{++} [197]), heuristic (e.g., [196]), genetic (e.g., [136]), etc. Indeed, heuristic and genetic algorithms have been introduced to cope with noise, which the pure algorithmic techniques were not able to manage. Whereas algorithmic processes rely on footprints of traces (i.e., tables reporting whether events appeared before or afterwards, if decidable) to determine the workflow net that could have generated them, heuristic approaches build a representation similar to causal nets, taking frequencies of events and sequences into account when constructing the process model, in order to ignore infrequent paths. Genetic process mining adopts an evolutionary approach to the discovery and differs from the other two in that its computation evolves in a non-deterministic way: the final output, indeed, is the result of a simulation of a process of natural selection and evolutionary reproduction of the procedures used to determine the final outcome. [43] discusses in depth the user-tunable metrics adopted for the genetic algorithm, in order to make it return qualitatively better workflows: replay fitness, precision, generalization and simplicity. The accurate results are valuable, though such an algorithm suffers from unpredictability in terms of the process returned, which can change from run to run, due to the nature of evolutionary algorithms itself, and the time it might take, which is generally high.

A very smart extension to the previous research work has been recently achieved

by the two-steps algorithm proposed in [1].

Differently from previous works, which typically provide a single process mining step, it splits the computation in two phases: *(i)* the tunable mining of a Transition System (TS) representing the process behavior, and *(ii)* the automated construction of a Petri Net bisimilar to the TS [63, 72].

The first phase was made "tunable", so that it could be either more strictly adhering or more permissive to the analyzed log traces behavior, i.e., the expert could determine a balance between "overfitting" and "underfitting".

Indeed, past execution traces are not the whole universe of possible ones that may run: hence, the extracted process model should be valid for future unpredictable cases, on one hand, nevertheless checking whether the latter actually adhere to the common behavior, on the other hand. We also remark that a little percentage of the whole log might represent erroneous deviations from the natural flow of tasks. The second phase had no parameter to set, since its only aim was to synthesize the TS into an equivalent Workflow Net. Thus, it was fixed, while the former step could be realized exploiting one among many of the previously proposed "one-step" algorithms (for instance, [196] was claimed to integrate well).

The need for flexibility in the definition of some types of process, such as artful business processes, lead to an alternative to the classical "imperative" approach: the "declarative". Rather than using a procedural language for expressing the allowed sequences of activities, it is based on the description of workflows through the usage of constraints: the idea is that every task can be performed, except what does not respect them. Declarative models especially apply to artful processes, i.e., rapid informal processes that are typically carried out by those people whose work is mental rather than physical (managers, professors, researchers, engineers, etc.), the so called "knowledge workers". [5] showed how the declarative approach (such as the one of DECLARE modeling language [160]) can help in obtaining a fair trade-off between flexibility in managing collaborative processes and support in controlling and assisting the enactment of workflows. The definition of these languages was inspired by the observation that, in a process, possible execution alternatives can be classified as: *(i)* forbidden, i.e., executions that are not allowed and should never occur; *(ii)* optional, i.e., executions that, although should be avoided, are still possible and admitted; *(iii)* allowed, i.e., the intended normal executions. Instead of strictly and rigidly defining the control-flow of process tasks using a procedural language, such languages aims at supporting the identification and specification of a (minimal) set of policies and business rules (i.e., constraints) to be satisfied and followed in order to successfully perform a process and achieve the intended goals. This approach results in a shift to "all that is not explicitly forbidden is allowed".

[131] outlines an algorithm for mining Declare processes, implemented in ProM. The approach works as follows. The user is asked to specify a set of Declare constraint templates. Then, the system generates all of the possible constraints stemming from them, i.e., obtained by the application of those templates to all of the possible subsets of activities in the process. The user is also required to set a parameter named PoE (Percentage of Events), ranging from 0 to 100, so to avoid the generation of constraints involving those activities that appear, in percentage, less than PoE times in the log (i.e., rare activities, according to the user preferences). Then, every candidate constraint is translated into the related accepting finite automata,

according to the rules defined in [158]. For the optimization of this task, the tool is integrated with the technique described in [198]. Traces are thus replayed on the resulting automata. Each constraint among the candidates becomes part of the discovered process if and only if the percentage of traces accepted by the related automaton exceeds a user-defined threshold, named PoI (Percentage of Instances).

[130] proposes an evolution of [131]: there, a two-phase approach is adopted. The first phase is based on the Apriori algorithm, developed by Agrawal and Srikant for mining association rules [9]. During this preliminary step, the correlated activity sets are identified. The candidate constraints are computed on the basis of the correlated activity sets only. During the second phase, the candidate constraints are checked as in [131]. Therefore, the search space for the second phase is reduced.

In output, constraints constituting the discovered process are weighted according to their *Support*, i.e., the probability of such constraint to hold in the mined workflow. There, it is calculated as the proportion of traces where the constraint is satisfied. To filter out irrelevant constraints, more metrics are introduced, based on the appearances of the activities involved within the log: they are *Confidence*, *Interest Factor* and *CPIR* (Conditional-Probability Increment Ratio).

Authors in [74] introduces MINERful++ (see Section 4.2.1 for further details), a two-step algorithm for an efficient discovery of constraints that constitute declarative workflow models. As a first step, a knowledge base is built, with information about temporal statistics gathered from execution traces. Then, the statistical support of constraints is computed, by querying that knowledge base. Noticeably MINERful++ is independent of the specific formalism adopted for representing constraints, based on a probabilistic approach and capable of eliminating the redundancy of subsumed constraints. The algorithm has been employed to mine process models from email flows [73].

[115, 51] describes the usage of inductive logic programming techniques to mine models expressed as a $\mathcal{S}$CIFF [11] first-order logic theory, consisting in a set of implication rules named Social Integrity Constraints (IC's for short). [52] shows how ConDec/DecSerFlow can be translated into SCIFF and a subset of SCIFF can be translated into Condec/Decserflow. Finally, the learned theory is automatedly translated into the ConDec [157] notation.

[51] proposes the implementation of the framework, named DPML (Declarative Process Model Learner [116]) as a ProM plug-in. [31, 30] extends this technique by weighting in a second phase the constraints with a probabilistic estimation. The learned IC's are indeed translated from $\mathcal{S}$CIFF, discovered by DPML, into Markov Logic [174] formulae, so that their probabilistic-based weighting is computed by the Alchemy tool. They both rely on the availability of compliant and non-compliant traces of execution, with respect to the process to mine. For instance, [115] takes a real log from cervical cancer screening careflows. All the traces have been analyzed by a domain expert and labeled as compliant or non compliant with respect to the protocol, adopted in the screening center. [31] takes as a case study the records of the careers which belonged to their affiliated university's students. In that case, positive traces were represented by graduated students, whilst negative traces were related with students who did not finish their studies. For a comprehensive insight on the logic-based approach to declarative workflow mining, the reader can refer to [143].

A collateral problem to process mining (both imperative and declarative) is bridging the gap between events given as input and tasks composing the models obtained as output. Authors in [26], an abstraction approach is proposed able to deal with n:m relations between events and activities also supporting concurrency. The approach consists of four distinct phases. During the first phase, tasks that will concur to the extracted models are annotated in order to simplify the mapping between between events and corresponding tasks; the annotation consists of a text description. During the second phase, objects related to event types are automatically extracted from the text description of each task. After the second step, a potential association between events and tasks is available. At this point, during the third step, the potential mapping obtained during the second step is employed to perform contextual mapping of event instances to task types. To this end, authors consider the context of an event, either as defined over the event attributes or the surrounding event instances. During the last phase, event instances are clustered into activity instances.

MINERful++ [74]

Our proposed methodology makes use of the MINERful++ algorithm [74], which is targeted to mine declarative process models composed by the constraints shown in Table 3.1. Hence an insight of the algorithm is needed to continue the subject.

MINERful++ is based on the concept of MINERfulKB, which is the knowledge base that represents statistical data needed to discover the constraints of the declarative process represented by the log. In fact, MINERfulKB is queried during the execution of the technique. Therefore, MINERful++ is a two-step algorithm. The first step consists of the construction of MINERfulKB, whereas the second step infers the declarative model through the analysis of MINERfulKB.

Let Σ be a finite alphabet: the symbols in the alphabet are meant to correspond to *tasks* in a process. Therefore, the terms "activity", "character" and "symbol" are interchangeably used. A log is a collection of traces, i.e., a finite sequence of tasks. We consider $T \subset \Sigma^*$ as the log, where Σ^* is the set of traces of elements in Σ. The terms "trace" and "string" are interchangeably used for denoting every $t \in T$. The authors introduce six functions mapping to integers a log T and either one character $\rho \in \Sigma$ or two characters $\rho, \sigma \in \Sigma$. Such numbers are interpreted in MINERFUL++ as the result of specific quantitative analyses performed on collections of strings (logs). The six functions are grouped into the MINERful *ownplay* and the MINERful *interplay*. The former collects information referring to single characters. The latter describes the relation between couple of characters. The possible occurrences of ρ and σ in the string are referred to as either *pivot* or *searched* characters, depending on the role they play in the definition of the following functions. In the following definitions, the authors denote by ρ' (resp., σ') any occurrence of ρ (resp., σ) in a string other than the pivot or searched character. In the examples, we will assign $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ to the alphabet Σ. ρ will be assigned as a and σ as b. The assigned log $\mathsf{T} \subset \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}^*$ for T will change, case by case.

Definition 4.7 (MINERful interplay)**.** *A tuple* $\mathcal{I} = \langle \Sigma, \delta, \beta^{\rightarrow}, \beta^{\leftarrow} \rangle$*, where:*

$\delta(T, \rho, \sigma, d)$ $\delta : \Sigma^* \times \Sigma \times \Sigma \times \mathbb{Z} \rightarrow \mathbb{N}^+$ *is the* distance *function. It maps a distance*[3] $d \in \mathbb{Z}$ *between pivot* $\rho \in \Sigma$ *and searched* $\sigma \in \Sigma$ *to the number of cases when they appeared at distance* d*, in the traces of log* T *(e.g.,* $\delta(T, \mathsf{a}, \mathsf{b}, 2) = 4$ *means that there is the evidence of a searched* b *appearing* 2 *characters after the pivot* a *in* 4 *cases, given* $T = \{\mathsf{cacbcc}, \mathsf{acbcacba}, \mathsf{acbaaa}\}$*); we recall that* $\mathbb{N}^+$ *is the set of natural integers excluding zero*[4]*;*

$\beta^{\rightarrow}(T, \rho, \sigma)$ $\beta^{\rightarrow} : \Sigma^* \times \Sigma \times \Sigma \rightarrow \mathbb{N}$ *is the* in-between onwards appearance *function. Given a pivot* ρ *and the closest* following *occurrence of searched* σ*, it counts the number of* ρ'*s in-between, for every string of log* T *in which both* ρ *and* σ *occur. By closest following occurrence we mean that no other* σ' *is read after* ρ *and before* σ*.* $\beta^{\rightarrow}(T, \mathsf{a}, \mathsf{b}) = 2$ *means, e.g., that* a *appeared* 2 *times between the preceding occurrence of pivot* a *and the* closest *following occurrence of searched* b*, as in* $T = \{\mathsf{accaacb}\}$*,* $T = \{\mathsf{accabcaab}\}$*, or* $T = \{\mathsf{accab}, \mathsf{caab}\}$*;*

$\beta^{\leftarrow}(T, \rho, \sigma)$ $\beta^{\leftarrow} : \Sigma^* \times \Sigma \times \Sigma \rightarrow \mathbb{N}$ *is the* in-between onwards appearance *function. Given a pivot* σ *and the closest* preceding *occurrence of searched* ρ*, it counts the number of* σ'*s in-between, for every string of log* T *in which both* ρ *and* σ *occur. Here, closest preceding occurrence means that no other* ρ' *is read before* σ *and after* ρ*.* $\beta^{\leftarrow}(T, \mathsf{a}, \mathsf{b}) = 3$ *means, e.g., that* b *appeared* 3 *times between the following occurrence of pivot* b *and the* closest *preceding occurrence of searched* a*, as in* $T = \{\mathsf{acbcbbcb}\}$*,* $T = \{\mathsf{accbbcbcabb}\}$*, or* $T = \{\mathsf{accbbcb}, \mathsf{cabb}\}$*.*

Definition 4.8 (MINERful ownplay)**.** *A tuple* $\mathcal{O} = \langle \Sigma, \gamma, \alpha, \omega \rangle$*, where:*

$\gamma(T, \rho, n)$ $\gamma : \Sigma^* \times \Sigma \times \mathbb{N} \rightarrow \mathbb{N}$ *is the* global appearance *function. It maps the pivot* ρ *and a natural number* $n \in \mathbb{N}$ *to the number of traces of* T *in which* ρ *was read* n *times (e.g.,* $\gamma(T, \mathsf{a}, 4) = 2$ *means that the pivot* a *happens to be read exactly four times in only two strings in the log, as in* $T = \{\mathsf{aabbabcca}, \mathsf{babacaa}\}$*);*

$\alpha(T, \rho)$ $\alpha : \Sigma^* \times \Sigma \rightarrow \mathbb{N}$ *is the* initial appearance *function. It represents the number of strings where the pivot* ρ *appeared as the* initial *symbol (e.g., if* $\alpha(T, \mathsf{a}) = 5$*, five traces started with* a*, as in* $T = \{\mathsf{abc}, \mathsf{abbc}, \mathsf{aca}, \mathsf{aa}, \mathsf{a}\}$*);*

$\omega(T, \rho)$ $\omega : \Sigma^* \times \Sigma \rightarrow \mathbb{N}$ *is the* final appearance *function. It represents the number of strings where the pivot* ρ *appeared as the* last *symbol (e.g., if* $\omega(T, \mathsf{a}) = 0$*, no trace ended with* a*, as in* $T = \{\mathsf{abc}, \mathsf{abbc}\}$*).*

Definition 4.9 (MINERfulKB)**.** *A tuple* $\mathcal{KB} = \langle \mathcal{I}, \mathcal{O} \rangle$ *where* $\mathcal{I} = \langle \Sigma, \delta, \beta^{\rightarrow}, \beta^{\leftarrow} \rangle$ *is the MINERful interplay, and* $\mathcal{O} = \langle \Sigma, \gamma, \alpha, \omega \rangle$ *is the MINERful ownplay.*

The algorithm associates to each mined constraint a series of measures that help in understanding their likelihood and significance; this allows the algorithm

[3] The *distance* represents the number of characters between ρ and σ. It is a positive value if σ *follows* ρ, negative if σ *precedes* ρ.

[4] Thus, concurrency of events in a log is not considered, i.e., no pair of characters is read in the same position.

Constraint	Support function
$Existence(n, \rho)$	$1 - \frac{\sum_{i=0}^{n-1} \gamma_\rho(i)}{\lvert T \rvert}$
$Participation(\rho)$	$1 - \frac{\gamma_\rho(0)}{\lvert T \rvert}$
$Absence(m, \rho)$	$\frac{\sum_{i=0}^{m} \gamma_\rho(i)}{\lvert T \rvert}$
$AtMostOne(\rho)$	$\frac{\gamma_\rho(0)+\gamma_\rho(1)}{\lvert T \rvert}$
$Init(\rho)$	$\frac{\alpha_\rho}{\lvert T \rvert}$
$End(\rho)$	$\frac{\omega_\rho}{\lvert T \rvert}$
$RespondedExistence(\rho, \sigma)$	$1 - \frac{\delta_{\rho,\sigma}(\pm\infty)}{\Gamma_\rho}$
$Response(\rho, \sigma)$	$1 - \frac{\delta_{\rho,\sigma}(+\infty)}{\Gamma_\rho}$
$Precedence(\rho, \sigma)$	$1 - \frac{\delta_{\rho,\sigma}(-\infty)}{\Gamma_\sigma}$
$AlternateResponse(\rho, \sigma)$	$1 - \frac{\beta^{\rightarrow}_{\rho,\sigma}+\delta_{\rho,\sigma}(+\infty)}{\Gamma_\rho}$
$AlternatePrecedence(\rho, \sigma)$	$1 - \frac{\beta^{\leftarrow}_{\rho,\sigma}+\delta_{\sigma,\rho}(-\infty)}{\Gamma_\sigma}$
$ChainResponse(\rho, \sigma)$	$\frac{\delta_{\rho,\sigma}(1)}{\Gamma_\rho}$
$ChainPrecedence(\rho, \sigma)$	$\frac{\delta_{\sigma,\rho}(-1)}{\Gamma_\sigma}$
$CoExistence(\rho, \sigma)$	$1 - \frac{\delta_{\rho,\sigma}(\pm\infty)+\delta_{\sigma,\rho}(\pm\infty)}{\Gamma_\rho+\Gamma_\sigma}$
$NotCoExistence(\rho, \sigma)$	$\frac{\delta_{\rho,\sigma}(\pm\infty)+\delta_{\sigma,\rho}(\pm\infty)}{\Gamma_\rho+\Gamma_\sigma}$
$Succession(\rho, \sigma)$	$1 - \frac{\delta_{\rho,\sigma}(+\infty)+\delta_{\sigma,\rho}(-\infty)}{\Gamma_\rho+\Gamma_\sigma}$
$NotSuccession(\rho, \sigma)$	$\frac{\delta_{\rho,\sigma}(+\infty)+\delta_{\sigma,\rho}(-\infty)}{\Gamma_\rho+\Gamma_\sigma}$
$AlternateSuccession(\rho, \sigma)$	$1 - \frac{\beta^{\rightarrow}_{\rho,\sigma}+\delta_{\rho,\sigma}(+\infty)+\beta^{\leftarrow}_{\rho,\sigma}+\delta_{\sigma,\rho}(-\infty)}{\Gamma_\rho+\Gamma_\sigma}$
$ChainSuccession(\rho, \sigma)$	$\frac{\delta_{\rho,\sigma}(1)+\delta_{\sigma,\rho}(-1)}{\Gamma_\rho+\Gamma_\sigma}$
$NotChainSuccession(\rho, \sigma)$	$1 - \frac{\delta_{\rho,\sigma}(1)+\delta_{\sigma,\rho}(-1)}{\Gamma_\rho+\Gamma_\sigma}$

Table 4.2. Functions computing constraints' support

to be suitable for application areas, such as smart environments, where rules are not always satisfied. The main significance measure is the *support* of a constraint, that represents the percentage of events that confirm the constraint. The *confidence* is instead the product of the support with the percentage of traces where the so called *implying task* (the main task in the constraint) is present. Finally, the *interest factor*, is the product of the confidence with the percentage of traces where the so called *implied task* is present. With reference to Table 3.1 and the `Precedence(A, B)` constraint, the task `A` is the implying one whereas the task `B` is the implied one.

The actual mining algorithm consists of three steps. The first one populates the $\mathcal{B}^+$ bag. $\mathcal{B}^+$ is a collection of tuples $\langle b, s_b, c_b, i_b \rangle$, each associating to a constraint b the related *(i)* support s_b, *(ii)* confidence c_b, and *(iii)* interest factor i_b. Each kind of constraint has a different formula to calculate the support with respect to the MINERfulKB (see Table 4.2). The second step filters out of the output trivially deducible constraints, e.g., it is enough to say that `ChainPrecedence(a,`

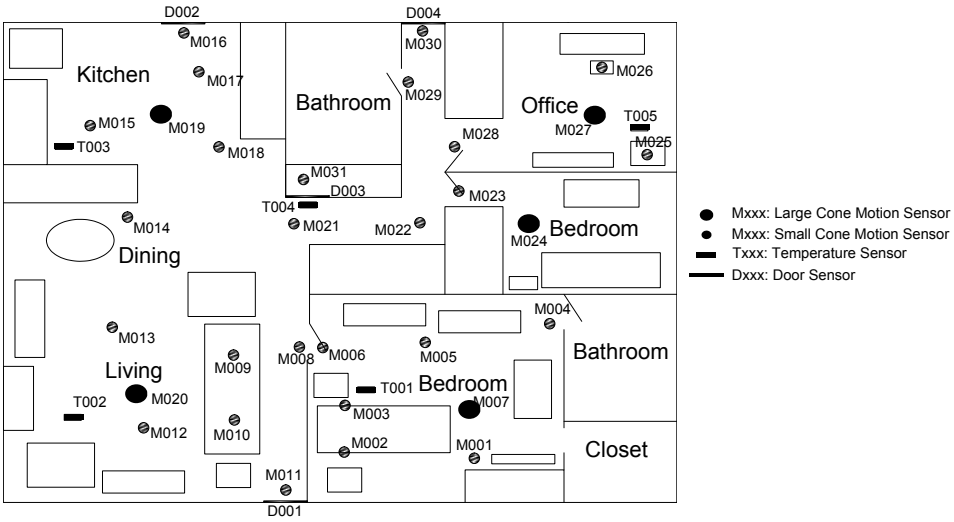


Figure 4.3. The Aruba setup of the CASAS project.

`b)` holds, rather than explicitly return as valid constraints `ChainPrecedence(a, b)`, `AlternatePrecedence(a, b)` and `Precedence(a, b)` – where the latter pair is directly implied by the first. In the last step, irrelevant constraints are filtered out looking at minimum thresholds of support, confidence and interest factor.

4.2.2 The Proposed Technique

In order to expose the proposed approach, we will take as an example a real installation whose datasets are freely available. Figure 4.3 depicts the Aruba setup provided by the CASAS project. The map shows the different rooms composing the environment, the most important furnitures and appliances, and the set of installed sensors. In particular, this setup contains four different categories of sensors, namely *(i)* large cone presence sensors, *(ii)* small cone presence sensors, *(iii)* temperature sensors, and *(iv)* door sensors.

Presence sensors trigger whenever something enters their field of action (also known as cone). As we are interested in modeling human habits, small cone presence sensors can provide a useful insights of what the user is doing at a certain time. Differently from large cone presence sensors, which cover large areas (e.g., an entire room), small cone ones allow to restrict the set of possible objects the user is interacting with. We will neglect temperature and door sensors in order to keep the discussion simple.

Unfortunately, small cone presence sensors do not give any direct hint of the reason why they trigger: is it because the user is moving from one location to another, or because the user is performing some action in the monitored location? In order to give an answer to this question, an important help is given by the timestamps that come with sensor measurements. Hence, the first step of our habit mining technique is to filter out those measurements generated by the user while moving from one interest location to another. This is the kind of specific bridging layer supposed in [26] to apply techniques from the process mining world to certain application areas where available logs have a finer granularity with respect to that of a task

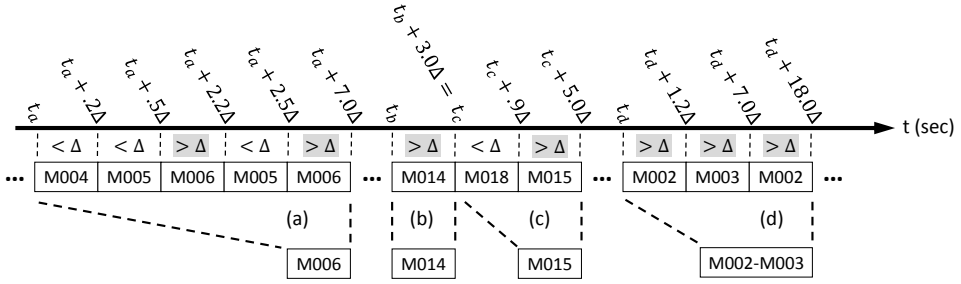


Figure 4.4. An example execution of the algorithm that transforms a sensor log into a task trace.

trace.

A trivial algorithm would simply keep a simple sensor measurement representative of a sequence of measurements coming from the very same presence sensor and lasting for a period at least long Δ, where Δ is parameter denoting the minimum duration of a task significant for the habit. Unfortunately, sensor measurements are not easily interpretable and the trivial algorithm can not be employed at our aim.

Figure 4.4 shows an example of execution of our solution that highlights the issues coming from the analysis of a stream of presence sensors readings. Keeping unchanged the behavior in the trivial case, the following conditions may occur:

(a) In this condition, the user is under the M006 sensors but spurious measurements comes from sensor M005 that are under the threshold Δ. The algorithm in this case outputs a single measurement M006 with duration 7.0Δ.

(b,c) In conditions labeled with (b) and (c), the user spends a period greater than Δ under the M014 sensor and then he/she moves under the M015 sensor after quickly transiting under sensor M018. In this case our algorithm returns a task under M014 with duration 3.0Δ and another one under M015 with duration 5.0Δ. With reference to Figure 4.3, this conditions represent an user who first prepare something to eat in the kitchen, and then moves to the table in the living room.

(d) In the last condition, we have two sensors (M002 and M003) alternated with durations greater than Δ. In this case the algorithm decides to aggregate the areas under the sensor under a single area. With reference to Figure 4.3, this condition represents the user moving in the bed while sleeping.

The pseudo-code of the filtering technique is shown in Algorithm 4.3. Despite the apparent complexity of the code, the algorithm simply maintain a dynamic size window on the last sensor measurements and employ the threshold Δ to decide which measurements have to be merged in order to obtain a task. The different choices reflect the conditions depicted in Figure 4.4.

We applied the described filtering strategy to the dataset taken from the Aruba setup and reporting two years of daily life of a single person [5]. The result dataset has been segmented in order to obtain a different task trace for each day (we omitted holidays) and for several time slots (two hours each). In order to show

[5]The dataset is freely available at http://ailab.wsu.edu/casas/datasets/aruba2.zip.

Algorithm 4.3: Filtering Algorithm

```
1: function sensorLog2taskLog (S - sensor log; Δ - task minimum duration)
2: S_M ← {sensor measurements s ∈ S s.t. s.sensor = Mxxx}
3: W ← ⟨⟩ //A linked list containing the current window of measurements
4: D ← ∅ //A set containing interesting sensors for the current window
5: S_R ← ⟨⟩ //A linked list containing the result task log
6: previous ← null; reference ← null; alternate ← null
7: for all s ∈ S do
8:    if previous ≠ null then
9:       if previous.sensor ≠ s.sensor then
10:         closeWindow ← false
11:         Δ_p ← s.ts - previous.ts
12:         if Δ_p ≥ Δ then
13:            if ‖D‖ = 0 then
14:               reference = previous
15:            else if ‖D‖ = 1 then
16:               if previous.sensor ∈ D then reference = previous else alternate = previous end if
17:            else if ‖D‖ = 2 then
18:               if previous.sensor ∈ D then
19:                  if alternate ≠ null then
20:                     if previous.sensor ≠ alternate.sensor then alternate = null else alternate =
                        previous end if
21:                  end if
22:                  if alternate = null then reference = previous end if
23:               else
24:                  closeWindow ← true
25:               end if
26:            end if
27:            D = D ∪ previous.sensor
28:         else
29:            if ‖D‖ = 2 ∧ s.sensor ∉ D then closeWindow ← true end if
30:         end if
31:         if closeWindow then
32:            first ← null
33:            while ‖W‖ > 0 do
34:               curr ← W.get(0)
35:               if first = null ∧‖W‖ > 1 then
36:                  Δ_a = W.get(1).ts − curr.ts
37:                  if Δ_a ≥ Δ then first = curr end if
38:               end if
39:               W.remove(0)
40:               if curr = reference then break end if
41:            end while
42:            sensorName ← description of the sensor/s representing the task
43:            S_R.append([first.ts, sensorName, (reference.ts − first.ts) + Δ_a])
44:            if alternate ≠ null then
45:               D = D/{previous.sensor}
46:               reference = alternate
47:               alternate = previous
48:            else
49:               D = ∅
50:               reference = previous
51:            end if
52:         end if
53:         W.append(s)
54:         previous ← s
55:      end if
56:   else
57:      W.append(s)
58:      previous ← s
59:   end if
60: end for
61: return S_R
```

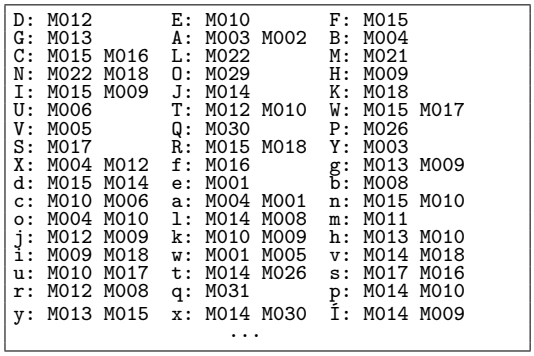


Figure 4.5. Sensor Mapping

Figure 4.6. Sensor Mapping

some preliminary results, we will concentrate on the time slot going from 6:00am to 8:00am that contains measurements related to the morning routing, which is the one that starts when the user leaves the bed. Figure 4.6 shows some of the traces extracted for this time slot; here, for the sake of compactness symbols are used in place of sensor names and the partial mapping is shown in Figure 4.5. The traces obtained for the morning routine are then fed as input to the MINERful++ algorithm.

As a preliminary result, we analyzed in a qualitative manner the results returned by the algorithm. As a simple example, the algorithm returns the constraints `Uniqueness(M015)` with support 0.74 and `Uniqueness(M015 M016)` with support 0.96, which imply that very often the user goes to the kitchen only once during his/her morning routine. The constraints `Precedence(M004, M015)` with support 0.83 and `Precedence(M004, M015 M016)` imply that in the vast majority of cases the user goes to the bathroom before going to the kitchen. As a final example, the algorithm return the constraint `AlternatePrecedence(M004, M012 M010)` with a support of 0.96 tells us that, almost every time, before sitting in the living room (likely in front of the TV) the user goes to the bathroom and then every time he/she leaves that place, he/she goes to the bathroom before returning back to sit there.

Future work will include of course a quantitative and massive evaluation of the algorithm. At the actual stage, the proposed algorithm shows room for improvements. In the proposed example, time slots have been chosen manually and arbitrarily; it would be nice if the time slots would be automatically decided by the algorithm also taking into account the calendar (e.g., holidays) and other environmental variables (e.g, weather). Additionally, supporting multiple users is a frequent issue of algorithms in the literature; in our case, heuristics could be employed to recognize different trajectories in the space. Finally, the contribution of task durations, at this stage, is only employed by the bridging algorithm whereas they may have the potential to distinguish among different kinds of interaction in the interested area, or to simply gather statistics of task durations for improving the informative content of the mined model.

Chapter 5

Conclusions and Future Work

In this thesis, we approached two major problems in smart spaces that we referred to as "instrumenting" and "mining" respectively. Instrumenting has been defined as the task of choosing and designing sensors and actuators aiming at maximizing the performance of AmI techniques employed in a specific smart space. Mining is instead the task of extracting knowledge from a smart space with a few intervention from the users in terms of labeling effort. Noticeably, in this thesis, these two tasks are not treated as separated aspects of the same research field but a strategy for exploiting them together is envisioned. In order to show this, let us analyze the thesis in the inverse order of the chapters.

Whereas many supervised approaches for learning models of a smart environment are known, we believe that unsupervised methods can provide many benefits, which include not only a lower burden for the final users during setup but also a straightforward advantage in adapting models at runtime. To this aim, Chapter 4 introduces a method for mining reactive rules and outlines a technique for mining declarative models of human habits. Sensor logs fed as input to these mining algorithms can be obtained from real installations or, alternatively, they can be gathered using crowdsourcing and gamification as explained in Section 3.2.

Habit mining algorithm have shown to be very promising and it can be useful to provide an AmI system with the models necessary to predict user needs and actuating the environment accordingly. Additionally, it can be useful for the generation of customized datasets by providing the input to the dataset generation tool proposed in Section 3.1. The dataset generation tool is intended as a facility for the all AmI community as, often, it is difficult to fairly evaluate algorithms with freely available datasets, and building a real setup can be expensive; the dataset generation tool allows instead to freely configure a virtual environment with the set of needed sensors and to execute high level models of human habits inside it.

One of the main flaws of mining algorithms for smart environment, is the support for multiple users as it is difficult to infer which users triggered a certain event. This problem affects the algorithm for habit mining proposed in Section 4.2.2 as well: the dataset employed for our qualitative evaluation contains measurements coming from a setup inhabited by a single subject. One solution to this issue would be to employ a localization system with identity recognition capabilities as the one introduced in Chapter 2.

Having said this, the proposed techniques have some limitations that may limit their applicability and that give us suggestions for future work.

A quantitative and massive evaluation of the habit mining algorithm is still needed. Such an evaluation would contribute to understand whether classical declarative constraints are the most appropriate to model human behavior. Hence, one of the future work will be to identify suitable constraints for modeling habits among those that are classically used in process mining, and to introduce new ones by specifying their semantics formally (e.g., using regular expressions). The choice of the constraints has also an impact on the dataset generation techniques we proposed in this thesis. Additionally, the bridging algorithm proposed to transform the sensor log into a set of task traces is still simple and novel heuristics could be employed to better recognize tasks from sensor measurements.

The algorithm proposed for mining reactive rules, at the current stage, only supports sensors with discrete output values. As many sensors in the smart environment output analog values (e.g., amp meters) a big contribution to the practical applicability of the proposed technique would be given by the introduction of discretization techniques.

The introduced dataset generation technique, at this moment, only generates sensor log coming from single subjects. To overcome this issue, techniques for modeling, both at the habit layer and at the planning layer, interactions between multiple subjects.

The crowdsourcing approach proposed for gathering datasets still needs a massive evaluation and a definitive technique to ensure truthfulness of users is still to be found. Additionally the integration of this platform with the dataset generation tool (see Section 3.3) is an ongoing work.

Our indoor localization system suffers in low light conditions. A solution to this first issue can come by employing infrared cameras, but this imply the necessity of devising techniques for background extraction and person tracking that do not rely on colors. Low light conditions also affect the identity recognition capabilities of the system. Additionally, the system suffers in crowded environment as in such situations, following the current approach, people become not separable on the simulated ceiling view.

Bibliography

[1] van der Aalst, W.M.P., , Rubin, V., Verbeek, E., van Dongen, B.F., Kindler, E., Günther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting. Software and Systems Modeling 9, 87–111 (2010), 10.1007/s10270-008-0106-z (Cited on page 91)

[2] van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998), http://dx.doi.org/10.1142/S0218126698000043 (Cited on page 89)

[3] van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011) (Cited on pages 13 and 89)

[4] van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Rozinat, A., Verbeek, E., Weijters, T.: ProM: The process mining toolkit. In: BPM (Demos) (2009) (Cited on page 89)

[5] van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - R&D 23(2), 99–113 (2009) (Cited on page 91)

[6] van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004) (Cited on page 90)

[7] Aggarwal, C.C.: Managing and Mining Sensor Data. Springer Publishing Company, Incorporated (2013) (Cited on page 5)

[8] Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.J., Alonso, G., Saltor, F., Ramos, I. (eds.) Advances in Database Technology – EDBT'98, Lecture Notes in Computer Science, vol. 1377, pp. 467–483. Springer Berlin / Heidelberg (1998) (Cited on page 90)

[9] Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th Int. Conf. Very Large Data Bases, VLDB. vol. 1215, pp. 487–499 (1994) (Cited on pages 10, 11, 13, 21, 77, 79, and 92)

[10] von Ahn, L., Dabbish, L.: Designing games with a purpose. Commun. ACM 51(8), 58–67 (Aug 2008), http://doi.acm.org/10.1145/1378704.1378719 (Cited on page 74)

[11] Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: The SCIFF framework. ACM Trans. Comput. Log. 9(4), 29:1–29:43 (August 2008), http://doi.acm.org/10.1145/1380572.1380578 (Cited on page 92)

[12] Aloise, F., Schettini, F., Aricò, P., Leotta, F., Salinari, S., Mattia, D., Babiloni, F., Cincotti, F.: P300-based brain–computer interface for environmental control: an asynchronous approach. Journal of neural engineering 8(2) (2011) (Cited on page xi)

[13] Aloise, F., Schettini, F., Aricò, P., Leotta, F., Salinari, S., Mattia, D., Babiloni, F., Cincotti, F.: Toward domotic appliances control through a self-paced p300-based bci. In: Proceedings of the 2011 International Joint Conference on Biomedical Engineering Systems and Technologies. pp. 239–244 (2011) (Cited on page xi)

[14] Anagnostopoulos, C., Ntarladimas, Y., Hadjiefthymiades, S.: Situational computing: An innovative architecture with imprecise reasoning. Journal of Systems and Software 80(12), 1993–2014 (2007) (Cited on page 16)

[15] Anwar, A.K., Ioannis, G., Pavlidou, F.: Evaluation of indoor location based on combination of agps/hsgps. In: In proceedings of 3rd International Symposium on Wireless Pervasive Computing (ISWPC). pp. 383–387 (2008) (Cited on page 26)

[16] Araujo, F., Al-Zinati, M., Valente, J., Kuiper, D., Zalila-Wenkstern, R.: Divas 4.0: A framework for the development of situated multi-agent based simulation systems. In: 12th Intl. Conf. on Autonomous Agents and Multiagent Systems (2013) (Cited on page 74)

[17] Augusto, J., Nugent, C.: Smart homes can be smarter. In: Augusto, J., Nugent, C. (eds.) Designing Smart Homes, Lecture Notes in Computer Science, vol. 4008, pp. 1–15. Springer Berlin / Heidelberg (2006), http://dx.doi.org/10.1007/11788485_1 (Cited on page 2)

[18] Augusto, J.C., Liu, J., McCullagh, P., Wang, H., Yang, J.B.: Management of uncertainty and spatio-temporal aspects for monitoring and diagnosis in a smart home. International Journal of Computational Intelligence Systems 1(4), 361–378 (2008), http://www.tandfonline.com/doi/abs/10.1080/18756891.2008.9727632 (Cited on page 11)

[19] Augusto, J.C., Nugent, C.D.: The use of temporal reasoning and management of complex events in smart homes. In: ECAI. vol. 16, p. 778. Citeseer (2004) (Cited on page 11)

[20] Aztiria, A., Augusto, J.C., Basagoiti, R., Izaguirre, A., Cook, D.J.: Discovering frequent user–environment interactions in intelligent environments. Personal and Ubiquitous Computing 16(1), 91–103 (2012) (Cited on pages 11, 87, and 88)

[21] Aztiria, A., Izaguirre, A., Augusto, J.C.: Learning patterns in ambient intelligence environments: a survey. Artificial Intelligence Review 34(1), 35–51 (2010) (Cited on pages v and 3)

[22] Aztiria, A., Izaguirre, A., Basagoiti, R., Augusto, J.C., Cook, D.J.: Discovering frequent sets of actions in intelligent environments. In: Intelligent Environments. pp. 153–160. Citeseer (2009) (Cited on page 13)

[23] Aztiria, A., Izaguirre, A., Basagoiti, R., Augusto, J.C., Cook, D.J.: Automatic modeling of frequent user behaviours in intelligent environments. In: Intelligent Environments (IE), 2010 Sixth International Conference on. pp. 7–12. IEEE (2010) (Cited on pages 13, 88, and 89)

[24] Bahadori, S., Iocchi, L., Leone, G., Nardi, D., Scozzafava, L.: Real-time people localization and tracking through fixed stereo vision. Applied Intelligence 26, 83–97 (2007) (Cited on pages 27, 31, 53, 54, 55, 59, and 60)

[25] Bahl, P., Padmanabhan, V.: Radar: an in-building rf-based user location and tracking system. In: Proceedings of 19th IEEE Annual Joint Conference (INFOCOM). vol. 2, pp. 775 –784 vol.2 (2000) (Cited on pages 54 and 55)

[26] Baier, T., Mendling, J.: Bridging abstraction layers in process mining by automated matching of events and activities. In: Business Process Management, pp. 17–32. Springer (2013) (Cited on pages 89, 93, and 96)

[27] Bao, L., Intille, S.: Activity recognition from user-annotated acceleration data. Pervasive Computing pp. 1–17 (2004) (Cited on page 19)

[28] Barnes, J., Rizos, C., Wang, J., Small, D., Voigt, G., Gambale, N.: Locata: A new positioning technology for high precision indoor and outdoor positioning. In: Proceedings of the 2003 International Symposium on GPS/GNSS. pp. 9–18 (2003) (Cited on page 26)

[29] Beis, J.S., Lowe, D.G.: Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In: In Proceedings of 1997 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1000–1006 (1997) (Cited on page 33)

[30] Bellodi, E., Riguzzi, F., Lamma, E.: Probabilistic declarative process mining. In: KSEM. pp. 292–303 (2010) (Cited on page 92)

[31] Bellodi, E., Riguzzi, F., Lamma, E.: Probabilistic logic-based process mining. In: CILC (2010) (Cited on page 92)

[32] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6(2), 161–180 (2010) (Cited on page 14)

[33] Beymer, D., Konolige, K.: Real-time tracking of multiple people using continuous detection. In: IEEE Frame Rate Workshop (1999) (Cited on page 53)

[34] Bishop, C., en ligne), S.S.: Pattern recognition and machine learning, vol. 4. springer New York (2006) (Cited on pages viii and 7)

[35] Bloisi, D.D., Iocchi, L., Marchetti, L., Monekosso, D.N., Remagnino, P.: An adaptive tracker for assisted living. In: Proceedings of the 6th IEEE Conference on Advanced Video and Signal Based Surveillance. pp. 164–169 (2009) (Cited on pages 45, 54, 55, and 62)

[36] Bloisi, D.D., Iocchi, L., Monekosso, D.N., Remagnino, P.: A novel segmentation method for crowded scenes. In: Proceedings of 4th International Conference on Computer Vision Theory and Applications (VISAPP). pp. 484–489 (2009) (Cited on pages 54 and 55)

[37] Bradski, G., Kaehler, A.: Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Incorporated (2008) (Cited on pages 29, 45, and 46)

[38] Brdiczka, O., Crowley, J., Reignier, P.: Learning situation models in a smart home. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 39(1), 56–63 (2009) (Cited on page 19)

[39] Broy, M., Cengarle, M.V., Geisberger, E.: Cyber-physical systems: Imminent challenges. In: Large-Scale Complex IT Systems. Development, Operation and Management, pp. 1–28. Springer (2012) (Cited on pages iv and 1)

[40] Bruneau, J., Jouve, W., Consel, C.: Diasim: A parameterized simulator for pervasive computing applications. In: Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, 2009. MobiQuitous' 09. 6th Annual International. pp. 1–10. IEEE (2009) (Cited on page 74)

[41] Büchi, J.R.: Weak second-order arithmetic and finite automata. Mathematical Logic Quarterly 6(1-6), 66–92 (1960) (Cited on page 65)

[42] Bui, H., Venkatesh, S., West, G.: Policy recognition in the abstract hidden markov model. Journal of Artificial Intelligence Research 17, 451–499 (2002) (Cited on page 18)

[43] Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: On the role of fitness, precision, generalization and simplicity in process discovery. In: CoopIS (2012) (Cited on page 90)

[44] Calisi, D., Iocchi, L., Nardi, D., Randelli, G., Ziparo, V.: Improving search and rescue using contextual information. Advanced Robotics 23(9), 1199–1216 (2009) (Cited on page 29)

[45] Caruso, M., Cincotti, F., Leotta, F., Mecella, M., Riccio, A., Schettini, F., Simione, L., Catarci, T.: My-world-in-my-tablet: An architecture for people with physical impairment. In: Proceedings of the 15th International Conference on Human-Computer Interaction. Interaction Modalities and Techniques. pp. 637–647 (2013) (Cited on page xi)

[46] Caruso, M., Ilban, c., Leotta, F., Mecella, M., Vassos, S.: Synthesizing daily life logs through gaming and simulation. In: Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication. pp. 451–460. UbiComp '13 Adjunct, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2494091.2495977 (Cited on pages vii, viii, and ix)

[47] Caruso, M., Leotta, F., Mecella, M., Vassos, S.: Benchmarking smart spaces through autonomous virtual agents. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems. pp. 1229–1230. AAMAS '13, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2013), http://dl.acm.org/citation.cfm?id=2484920.2485156 (Cited on pages vii, viii, and ix)

[48] Catarci, T., Ciccio, C., Forte, V., Iacomussi, E., Mecella, M., Santucci, G., Tino, G.: Service composition and advanced user interfaces in the home of tomorrow: The sm4all approach. In: Ambient Media and Systems, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 70, pp. 12–19 (2011) (Cited on page 42)

[49] Chen, H., Finin, T., Joshi, A.: An ontology for context-aware pervasive computing environments. The Knowledge Engineering Review 18(03), 197–207 (2003) (Cited on page 15)

[50] Chen, L., Nugent, C., Mulvenna, M., Finlay, D., Hong, X.: Semantic smart homes: towards knowledge rich assisted living environments. Intelligent patient management pp. 279–296 (2009) (Cited on page 14)

[51] Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Exploiting inductive logic programming techniques for declarative process mining. T. Petri Nets and Other Models of Concurrency 2, 278–295 (2009) (Cited on page 92)

[52] Chesani, F., Mello, P., Montali, M., Storari, S.: Towards a DecSerFlow declarative semantics based on computational logic. Tech. rep., DEIS, Università degli Studi di Bologna (2007) (Cited on page 92)

[53] Cheung, Y.L., Fu, A.W.C.: Mining frequent itemsets without support threshold: with and without item constraints. IEEE Trans. on Knowledge and Data Engineering 16(9), 1052–1069 (2004) (Cited on page 80)

[54] Cimino, M., Lazzerini, B., Marcelloni, F., Ciaramella, A.: An adaptive rule-based approach for managing situation-awareness. Expert Systems with Applications (2012) (Cited on page 16)

[55] Cincotti, F., Pichiorri, F., Arico, P., Aloise, F., Leotta, F., de Vico Fallani, F., del R Millan, J., Molinari, M., Mattia, D.: Eeg-based brain-computer interface to support post-stroke motor rehabilitation of the upper limb. In: Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE. pp. 4112–4115. IEEE (2012) (Cited on page xi)

[56] Clarke, E.M., Grumberg, O., Peled, D.: Model checking. MIT press (1999) (Cited on page 16)

[57] Coles, A., Fox, M., Long, D., Smith, A.: Teaching forward-chaining planning with javaff. In: 23rd AAAI Conference on Artificial Intelligence (2008) (Cited on page 67)

[58] Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 5(4), 277–298 (2009) (Cited on pages iv and 1)

[59] Cook, D.: Learning setting-generalized activity models for smart spaces. IEEE intelligent systems (2010) (Cited on pages 19 and 73)

[60] Cook, D., Crandall, A., Thomas, B., Krishnan, N.: Casas: A smart home in a box. Computer 46(7), 62–69 (July 2013) (Cited on pages vii and 6)

[61] Cook, D., Krishnan, N., Rashidi, P.: Activity discovery and activity recognition: A new partnership. Cybernetics, IEEE Transactions on 43(3), 820–828 (June 2013) (Cited on pages 20, 21, and 88)

[62] Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Trans. Softw. Eng. Methodol. 7(3), 215–249 (1998) (Cited on page 90)

[63] Cortadella, J., Kishinevsky, M., Lavagno, L., Yakovlev, A.: Deriving petri nets from finite transition systems. Computers, IEEE Transactions on 47(8), 859 –882 (aug 1998) (Cited on page 91)

[64] Cristani, M., Farenzena, M., Bloisi, D., Murino, V.: Background subtraction for automated multisensor surveillance: a comprehensive review. EURASIP Journal on Advances in signal Processing 2010, 43:1–43:24 (2010) (Cited on pages 25 and 31)

[65] Cucchiara, R., Grana, C., Piccardi, M., Prati, A.: Detecting moving objects, ghosts, and shadows in video streams. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(10), 1337 – 1342 (oct 2003) (Cited on pages 31 and 58)

[66] Darrell, T., Demirdjian, D., Checka, N., Felzenszwalb, P.: Plan-view trajectory estimation with dense stereo background models. In: Proceedings of 8th IEEE International Conference on Computer Vision. vol. 2, pp. 628–635 (2001) (Cited on pages 31 and 53)

[67] De Giacomo, G., Di Ciccio, C., Felli, P., Hu, Y., Mecella, M.: Goal-based composition of stateful services for smart homes. In: On the Move to Meaningful Internet Systems: OTM 2012, pp. 194–211. Springer (2012) (Cited on page 12)

[68] Degeler, V., Gonzalez, L.I., Leva, M., Shrubsole, P., Bonomi, S., Amft, O., Lazovik, A.: Service-oriented architecture for smart environments (short paper). In: Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on. pp. 99–104 (Dec 2013) (Cited on page 78)

[69] Degeler, V., Lazovik, A.: Dynamic constraint reasoning in smart environments. In: IEEE Int. Conf. Tools wth Artificial Intelligence (2013) (Cited on pages 12 and 78)

[70] Degeler, V., Lazovik, A., Leotta, F., Mecella, M.: Itemset-based mining of constraints for enacting smart environments. In: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on. pp. 41–46 (March 2014) (Cited on page ix)

[71] Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society 39, 1–38 (1977) (Cited on page 31)

[72] Desel, J., Reisig, W.: The synthesis problem of petri nets. Acta Informatica 33, 297–315 (1996), http://dx.doi.org/10.1007/s002360050046, 10.1007/s002360050046 (Cited on page 91)

[73] Di Ciccio, C., Mecella, M.: Mining artful processes from knowledge workers' emails. IEEE Internet Computing 17(5), 10–20 (09 2013) (Cited on page 92)

[74] Di Ciccio, C., Mecella, M.: A two-step fast algorithm for the automated discovery of declarative workflows. In: 4th IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, April 16-19, 2013. pp. 135–142. IEEE (04 2013) (Cited on pages 65, 66, 92, and 93)

[75] Di Ciccio, C., Mecella, M., Caruso, M., Forte, V., Iacomussi, E., Rasch, K., Querzoni, L., Santucci, G., Tino, G.: The homes of tomorrow: Service composition and advanced user interfaces. ICST Transactions on Ambient Systems 11(10-12), e2 (12 2011) (Cited on pages 12, 22, and 77)

[76] Diekert, V., Gastin, P.: First-order definable languages. Logic and Automata: History and Perspectives 2, 261 (2008) (Cited on page 65)

[77] Dixon, C., Mahajan, R., Agarwal, S., Brush, A., Lee, B., Saroiu, S., Bahl, P., Santos, N., Rodrigues, R., Gummadi, K.P., et al.: An operating system for the home. In: Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation. pp. 175–188 (Cited on page 22)

[78] Dockhorn Costa, P., Guizzardi, G., Almeida, J., Ferreira Pires, L., van Sinderen, M.: Situations in conceptual modeling of context. In: Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW '06. 10th IEEE International. p. 6 (oct 2006) (Cited on page 14)

[79] Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: MACHINE LEARNING: PROCEEDINGS OF THE TWELFTH INTERNATIONAL CONFERENCE (1995) (Cited on page 8)

[80] Faulkner, W., Alwood, R., Taylor, D., Bohlin, J.: GPS-denied pedestrian tracking in indoor environments using an imu and magnetic compass. In: Proceedings of the 2010 International Technical Meeting of the Institute of Navigation (ITM). pp. 198–204 (2010) (Cited on pages 54 and 55)

[81] Fikes, R., Nilsson, N.: Strips: A new approach to the application of theorem proving to problem solving. Artificial intelligence 2(3), 189–208 (1972) (Cited on page 66)

[82] Fox, M., Long, D.: Pddl2.1: An extension to pddl for expressing temporal planning domains. J. Artif. Intell. Res. (JAIR) 20, 61–124 (2003) (Cited on page 66)

[83] Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. In: Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 163–173. ACM (1980) (Cited on page 65)

[84] Galushka, M., Patterson, D., Rooney, N.: Temporal data mining for smart homes. Designing Smart Homes pp. 85–108 (2006) (Cited on pages viii and 7)

[85] Gottfried, B., Guesgen, H., Hübner, S.: Spatiotemporal reasoning for smart homes. In: Augusto, J., Nugent, C. (eds.) Designing Smart Homes, Lecture Notes in Computer Science, vol. 4008, pp. 16–34. Springer Berlin / Heidelberg (2006) (Cited on page 13)

[86] Grewal, M.S., Weill, L.R., Andrews, A.P.: Global Positioning Systems, Inertial Navigation, and Integration. Wiley-Interscience (2007) (Cited on page 28)

[87] Gu, T., Pung, H., Zhang, D., Pung, H., Zhang, D.: A bayesian approach for dealing with uncertain contexts. In: Austrian Computer Society. Citeseer (2004) (Cited on page 17)

[88] Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications 28(1), 1 – 18 (2005), http://www.sciencedirect.com/science/article/pii/S1084804504000451 (Cited on page 15)

[89] Gu, T., Wu, Z., Tao, X., Pung, H.K., Lu, J.: epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In: Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on. pp. 1–9. IEEE (2009) (Cited on page 20)

[90] Gu, T., Wu, Z., Wang, L., Tao, X., Lu, J.: Mining emerging patterns for recognizing activities of multiple users in pervasive computing. In: Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, 2009. MobiQuitous' 09. 6th Annual International. pp. 1–10. IEEE (2009) (Cited on page 20)

[91] Hamid, R., Maddi, S., Bobick, A., Essa, I.: Unsupervised analysis of activity sequences using event-motifs. In: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks. pp. 71–78. ACM (2006) (Cited on page 21)

[92] Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I., Isbell, C.: A novel sequence representation for unsupervised analysis of human activities. Artificial Intelligence 173(14), 1221–1244 (2009) (Cited on page 21)

[93] Han, J., Kamber, M., Pei, J.: Data mining: concepts and techniques. Morgan kaufmann (2006) (Cited on pages viii and 7)

[94] Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, second edn. (2004) (Cited on page 27)

[95] Harville, M.: Stereo person tracking with adaptive plan-view templates of height and occupancy statistics. Image and Vision Computing 22(2), 127 – 142 (2004) (Cited on pages 27, 31, 32, 53, 54, 55, and 60)

[96] Harville, M., Gordon, G., Woodfill, J.: Foreground segmentation using adaptive mixture models in color and depth. In: Proceedings of 2001 IEEE Workshop on Detection and Recognition of Events in Video. pp. 3–11 (2001) (Cited on page 31)

[97] Helal, A., Mendez-Vazquez, A., Hossain, S.: Specification and synthesis of sensory datasets in pervasive spaces. In: IEEE Symposium on Computers and Communications. pp. 920–925 (2009) (Cited on page 73)

[98] Helaoui, R., Riboni, D., Niepert, M., Bettini, C., Stuckenschmidt, H.: Towards activity recognition using probabilistic description logics. In: Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence (2012) (Cited on page 17)

[99] Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: Models and approach. Pervasive and Mobile Computing 2(1), 37 – 64 (2006), http://www.sciencedirect.com/science/article/pii/S1574119205000441 (Cited on page 7)

[100] Hong, X., Nugent, C., Mulvenna, M., McClean, S., Scotney, B., Devlin, S.: Evidential fusion of sensor data for activity recognition in smart homes. Pervasive and Mobile Computing 5(3), 236–252 (2009) (Cited on page 16)

[101] Horprasert, T., Harwood, D., Davis, L.S.: A statistical approach for real-time robust background subtraction and shadow detection. In: IEEE Frame Rate Workshop (1999) (Cited on pages 31 and 58)

[102] Huynh, T., Fritz, M., Schiele, B.: Discovery of activity patterns using topic models. In: Proceedings of the 10th international conference on Ubiquitous computing. pp. 10–19. ACM (2008) (Cited on page 21)

[103] Ijsselmuiden, J., Stiefelhagen, R.: Towards high-level human activity recognition through computer vision and temporal logic. In: KI 2010: Advances in Artificial Intelligence, pp. 426–435. Springer (2010) (Cited on page 16)

[104] ISTAT: Censimento popolazione e abitazioni: Popolazione residente, famiglie e convivenze (2011), http://dati-censimentopopolazione.istat.it/ [19 February 2014] (Cited on page 50)

[105] Jensen, C.S., Lu, H., Yang, B.: Indoor - a new data management frontier. IEEE Data Engineering Bulletin 33(2), 12–17 (2010) (Cited on page 29)

[106] Kaldeli, E., Warriach, E.U., Lazovik, A., Aiello, M.: Coordinating the web of services for a smart home. ACM Transactions on the Web 7(2) (2013) (Cited on page 12)

[107] Kamar, E., Hacker, S., Horvitz, E.: Combining human and machine intelligence in large-scale crowdsourcing. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1. pp. 467–474. AAMAS '12, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2012), http://dl.acm.org/citation.cfm?id=2343576.2343643 (Cited on page 75)

[108] Kamar, E., Horvitz, E.: Incentives and truthful reporting in consensus-centric crowdsourcing. Tech. rep., Technical report, MSR-TR-2012-16, Microsoft Research (2012) (Cited on page 75)

[109] Katz, S.: Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. Journal of the American Geriatrics Society (1983) (Cited on page 2)

[110] Kealy, A., Roberts, G., Retscher, G.: Evaluating the performance of low cost mems inertial sensors for seamless indoor/outdoor navigation. In: Proceedings of 2010 IEEE/ION Position Location and Navigation Symposium (PLANS). pp. 157 –167 (may 2010) (Cited on page 28)

[111] Kleih, S.C., Kaufmann, T., Zickler, C., Halder, S., Leotta, F., Cincotti, F., Aloise, F., Riccio, A., Herbert, C., Mattia, D.: Out of the frying pan into the fire–the p300-based bci faces real-world challenges. Progress in brain research 194, 27–46 (2010) (Cited on page xi)

[112] Koh, Y.S., Rountree, N.: Finding sporadic rules using apriori-inverse. In: Advances in Knowledge Discovery and Data Mining, pp. 97–106. Springer (2005) (Cited on page 80)

[113] Krishnan, N., Cook, D.J., Wemlinger, Z.: Learning a taxonomy of predefined and discovered activity patterns. Journal of Ambient Intelligence and Smart Environments 5(6), 621–637 (2013) (Cited on page 20)

[114] Krishnan, N.C., Cook, D.J.: Activity recognition on streaming sensor data. Pervasive and Mobile Computing (2012) (Cited on pages 9, 10, 19, and 73)

[115] Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Inducing declarative logic-based models from labeled traces. In: BPM. pp. 344–359 (2007), http://dx.doi.org/10.1007/978-3-540-75183-0_25 (Cited on page 92)

[116] Lamma, E., Mello, P., Riguzzi, F., Storari, S.: Applying inductive logic programming to process mining. In: ILP. pp. 132–146 (2007), http://dx.doi.org/10.1007/978-3-540-78469-2_16 (Cited on page 92)

[117] Lee, E.A.: Cyber physical systems: Design challenges. In: Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on. pp. 363–369. IEEE (2008) (Cited on pages iv and 1)

[118] Leotta, F., Mecella, M.: Plathea: a marker-less people localization and tracking system for home automation. Software: Practice and Experience pp. n/a–n/a (2014), http://dx.doi.org/10.1002/spe.2262 (Cited on pages vii, ix, and 23)

[119] Leotta, F., Mecella, M., Aloise, F.: Pericles: a performance evaluation platform for indoor localization systems. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness. pp. 23–30. ISA '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2077357.2077362 (Cited on pages vii, ix, and 48)

[120] Li, Q., Feng, L., Wong, A.: From intra-transaction to generalized inter-transaction: landscaping multidimensional contexts in association rule mining. Information Sciences 172(3), 361–395 (2005) (Cited on page 21)

[121] Li, S.Z., Jain, A.K.: Handbook of face recognition. Springer (2011) (Cited on page 28)

[122] Liu, H., Hussain, F., Tan, C.L., Dash, M.: Discretization: An enabling technique. Data mining and knowledge discovery 6(4), 393–423 (2002) (Cited on page 8)

[123] Liu, H., Darabi, H., Banerjee, P., Liu, J.: Survey of wireless indoor positioning techniques and systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37(6), 1067–1080 (2007) (Cited on page 27)

[124] Liu, Y., Yang, Z., Wang, X., Jian, L.: Location, localization, and localizability. Journal of Computer Science and Technology 25, 274–297 (2010) (Cited on page 27)

[125] Logan, B., Healey, J., Philipose, M., Tapia, E.M., Intille, S.: A long-term evaluation of sensing modalities for activity recognition. In: Proceedings of the 9th international conference on Ubiquitous computing. pp. 483–500. UbiComp '07, Springer-Verlag, Berlin, Heidelberg (2007), http://dl.acm.org/citation.cfm?id=1771592.1771620 (Cited on pages vi and 4)

[126] Loke, S.W.: Logic programming for context-aware pervasive computing: Language support, characterizing situations, and integration with the web. In: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence. pp. 44–50. IEEE Computer Society (2004) (Cited on page 14)

[127] Loke, S.W.: Incremental awareness and compositionality: A design philosophy for context-aware pervasive systems. Pervasive and Mobile Computing 6(2), 239–253 (2010) (Cited on page 15)

[128] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60, 91–110 (2004) (Cited on page 32)

[129] Lühr, S., West, G., Venkatesh, S.: Recognition of emergent human behaviour in a smart home: A data mining approach. Pervasive and Mobile Computing 3(2), 95–116 (2007) (Cited on pages 21 and 88)

[130] Maggi, F.M., Bose, R.P.J.C., van der Aalst, W.M.P.: Efficient discovery of understandable declarative process models from event logs. In: CAiSE. pp. 270–285 (2012) (Cited on page 92)

[131] Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: CIDM. pp. 192–199 (2011) (Cited on pages 91 and 92)

[132] Magherini, T., Fantechi, A., Nugent, C.D., Vicario, E.: Using temporal logic and model checking in automated recognition of human activities for ambient-assisted living. Human-Machine Systems, IEEE Transactions on 43(6), 509–521 (2013) (Cited on page 16)

[133] Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3), 259–289 (1997) (Cited on page 20)

[134] Martin, M., Nurmi, P.: A generic large scale simulator for ubiquitous computing. In: Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on. pp. 1–3. IEEE (2006) (Cited on page 74)

[135] McNaughton, R., Papert, S.A.: Counter-Free Automata (MIT research monograph no. 65). The MIT Press (1971) (Cited on page 65)

[136] Medeiros, A.K., Weijters, A.J., Aalst, W.M.: Genetic process mining: an experimental evaluation. Data Min. Knowl. Discov. 14(2), 245–304 (2007) (Cited on page 90)

[137] Mennicken, S., Huang, E.: Hacking the natural habitat: An in-the-wild study of smart homes, their development, and the people who live in them. In: Kay, J., Lukowicz, P., Tokuda, H., Olivier, P., KrÃijger, A. (eds.) Pervasive Computing, Lecture Notes in Computer Science, vol. 7319, pp. 143–160. Springer Berlin Heidelberg (2012) (Cited on page 2)

[138] Menon, V., Jayaraman, B., Govindaraju, V.: Multimodal identification and tracking in smart environments. Pers. and Ubiquitous Comp. (2010) (Cited on page 74)

[139] Merico, D., Bisiani, R.: An agent-based data-generation tool for situation-aware systems. In: 7th Intl. Conf. on Intelligent Environments (2011) (Cited on page 73)

[140] Miller, B.A., Nixon, T., Tai, C., Wood, M.D.: Home networking with universal plug and play. IEEE Communications Magazine 39(12), 104–109 (2001) (Cited on page 22)

[141] Miller, R.J., Yang, Y.: Association rules over interval data. In: ACM SIGMOD Record. vol. 26, pp. 452–461. ACM (1997) (Cited on page 8)

[142] Moeglein, M., Krasner, N.: An introduction to snaptrack server-aided gps technology. In: Proceedings of the 11th International Technical Meeting of the

Satellite Division of The Institute of Navigation. pp. 333–342 (1998) (Cited on page 26)

[143] Montali, M.: Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach, Lecture Notes in Business Information Processing, vol. 56. Springer (2010) (Cited on page 92)

[144] Müller-Putz, G.R., Breitwieser, C., Cincotti, F., Leeb, R., Schreuder, M., Leotta, F., Tavella, M., Bianchi, L., Kreilinger, A., Ramsay, A.: Tools for brain-computer interaction: a general concept for a hybrid bci. Frontiers in neuroinformatics 5 (2011) (Cited on page xi)

[145] Müller-Putz, G.R., Breitwieser, C., Tangermann, M., Schreuder, M., Tavella, M., Leeb, R., Cincotti, F., Leotta, F., Neuper, C.: Tobi hybrid bci: principle of a new assistive method. International Journal of Bioelectromagnetism 13(3), 144–145 (2011) (Cited on page xi)

[146] Neumann, B., Moller, R.: On scene interpretation with description logics. Image and Vision Computing 26(1), 82 – 101 (2008), cognitive Vision-Special Issue (Cited on page 16)

[147] Nguyen, N., Phung, D., Venkatesh, S., Bui, H.: Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. vol. 2, pp. 955–960. IEEE (2005) (Cited on page 18)

[148] Nguyen, T.A., Degeler, V., Contarino, R., Lazovik, A., Bucur, D., Aiello, M.: Towards context consistency in a rule-based activity recognition architecture. In: Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC). pp. 625–630. IEEE (2013) (Cited on pages 12 and 86)

[149] Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., Ito, M.: UbiREAL: Realistic Smartspace Simulator for Systematic Testing. In: Proc. 8th Int'l Conf. on Ubiquitous Computing (2006) (Cited on page 74)

[150] OASIS: Devices profile for web services (dpws) version 1.1 (2009), http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01 [19 February 2014] (Cited on pages 22 and 42)

[151] Palmes, P., Pung, H., Gu, T., Xue, W., Chen, S.: Object relevance weight pattern mining for activity recognition and segmentation. Pervasive and Mobile Computing 6(1), 43–57 (2010) (Cited on page 20)

[152] Pareschi, L., Riboni, D., Agostini, A., Bettini, C.: Composition and generalization of context data for privacy preservation. In: Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on. pp. 429–433. IEEE (2008) (Cited on page 6)

[153] Pareschi, L., Riboni, D., Bettini, C.: Protecting users' anonymity in pervasive computing environments. In: Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on. pp. 11–19. IEEE (2008) (Cited on page 6)

[154] Patrikakis, C., Karamolegkos, P., Voulodimos, A., Wahab, M.H.A., Taujuddin, N.S.A.M., Hanif, C., Pareschi, L., Riboni, D., Weber, S.G., Heinemann, A., ching Samson Cheung, S., Chaudhari, J., Paruchuri, J.K.: Security and privacy in pervasive computing. IEEE Pervasive Computing 6(4), 73–75 (2007) (Cited on page 6)

[155] Pellegrini, S., Iocchi, L.: Human posture tracking and classification through stereo vision and 3d model matching. Journal of Image Video Processings 2008, 7:1–7:12 (2008) (Cited on page 28)

[156] Pesic, M., Schonenberg, H., van der Aalst, W.: Declare: Full support for loosely-structured processes. In: 11th IEEE International Enterprise Distributed Object Computing Conference. pp. 287–287 (2007) (Cited on page 64)

[157] Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Business Process Management Workshops. pp. 169–180 (2006) (Cited on page 92)

[158] Pesic, M., Bosnacki, D., van der Aalst, W.M.P.: Enacting declarative languages using ltl: Avoiding errors and improving performance. In: SPIN. pp. 146–161 (2010), http://dx.doi.org/10.1007/978-3-642-16164-3_11 (Cited on page 92)

[159] Pešić, M., Bošnački, D., van der Aalst, W.M.: Enacting declarative languages using ltl: avoiding errors and improving performance. In: Model Checking Software, pp. 146–161. Springer (2010) (Cited on page 65)

[160] Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: Declare: Full support for loosely-structured processes. In: EDOC. pp. 287–300 (2007) (Cited on page 91)

[161] Plataniotis, K.N., Venetsanopoulos, A.N.: Color image processing and applications. Springer (2000) (Cited on page 60)

[162] Pnueli, A.: The temporal logic of programs. In: Foundations of Computer Science, 1977., 18th Annual Symposium on. pp. 46–57. IEEE (1977) (Cited on page 16)

[163] Poppe, R.: A survey on vision-based human action recognition. Image and vision computing 28(6), 976–990 (2010) (Cited on page 14)

[164] Rajkumar, R.R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: the next computing revolution. In: Proceedings of the 47th Design Automation Conference. pp. 731–736. ACM (2010) (Cited on pages iv and 1)

[165] Ramos, C., Marreiros, G., Santos, R., Freitas, C.F.: Smart offices and intelligent decision rooms. In: Handbook of Ambient Intelligence and Smart Environments, pp. 851–880. Springer (2010) (Cited on page 3)

[166] Ranganathan, A., McGrath, R., Campbell, R., Mickunas, M.: Use of ontologies in a pervasive computing environment. The Knowledge Engineering Review 18(03), 209–220 (2003) (Cited on page 15)

[167] Rashidi, P., Cook, D.: Keeping the resident in the loop: Adapting the smart home to the user. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 39(5), 949–959 (Sept 2009) (Cited on pages viii and 7)

[168] Rashidi, P., Cook, D.J., Holder, L.B., Schmitter-Edgecombe, M.: Discovering activities to recognize and track in a smart environment. Knowledge and Data Engineering, IEEE Transactions on 23(4), 527–539 (2011) (Cited on pages 20, 21, and 88)

[169] Raychoudhury, V., Cao, J., Kumar, M., Zhang, D.: Middleware for pervasive computing: A survey. Pervasive and Mobile Computing (2012) (Cited on page 22)

[170] Reinhardt, A., Baumann, P., Burgstahler, D., Hollick, M., Chonov, H., Werner, M., Steinmetz, R.: On the Accuracy of Appliance Identification Based on Distributed Load Metering Data. In: Proceedings of the 2nd IFIP Conference on Sustainable Internet and ICT for Sustainability (SustainIT). pp. 1–9 (2012) (Cited on pages vii and 5)

[171] Riboni, D., Bettini, C.: Context-aware activity recognition through a combination of ontological and statistical reasoning. Ubiquitous Intelligence and Computing pp. 39–53 (2009) (Cited on page 16)

[172] Riccio, A., Leotta, F., Aloise, F., Bianchi, L., Mattia, D., Cincotti, F.: Evaluation of a p300 overlaid stimulation for controlling an assistive technology software. International Journal of Bioelectromagnetism 13, 141–143 (2011) (Cited on page xi)

[173] Riccio, A., Leotta, F., Bianchi, L., Aloise, F., Zickler, C., Hoogerwerf, E., Kübler, A., Mattia, D., Cincotti, F.: Workload measurement in a communication application operated through a p300-based brain–computer interface. Journal of neural engineering 8(2), 025028 (2011) (Cited on page xi)

[174] Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006), http://dx.doi.org/10.1007/s10994-006-5833-1 (Cited on page 92)

[175] Roos, T., Myllymäki, P., Tirri, H., Misikangas, P., Sievänen, J.: A probabilistic approach to wlan user location estimation. International Journal of Wireless Information Networks 9(3), 155–164 (2002) (Cited on pages 54 and 55)

[176] Rosenthal, L., Stanford, V.M.: Nist smart space: Pervasive computing initiative. In: Proc. of the 9th IEEE Intl. Workshop on Enabling Technologies: Infrastructure for Collab. Enterprises. pp. 6–11 (2000), http://dl.acm.org/citation.cfm?id=647068.715632 (Cited on pages vii and 5)

[177] Ruotsalainen, M., Ala-Kleemola, T., Visa, A.: Gais: A method for detecting interleaved sequential patterns from imperfect data. In: Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on. pp. 530–534. IEEE (2007) (Cited on page 20)

[178] Russell, S.J., Norvig, P.: Artificial intelligence: A modern approach (1995) (Cited on pages v, 11, and 12)

[179] Sarkar, S., Phillips, P., Liu, Z., Vega, I., Grother, P., Bowyer, K.: The humanid gait challenge problem: data sets, performance, and analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(2), 162–177 (2005) (Cited on page 28)

[180] Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision 47, 7–42 (2002) (Cited on page 27)

[181] Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. In: Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition. pp. 195–202 (2003) (Cited on page 27)

[182] Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Proceedings of the 2006 IEEE Conference on Computer vision and pattern recognition. vol. 1, pp. 519–528 (2006) (Cited on page 27)

[183] Sentz, K., Ferson, S., Laboratories, S.N.: Combination of evidence in Dempster-Shafer theory. Citeseer (2002) (Cited on page 9)

[184] Shafer, G.: A mathematical theory of evidence, vol. 1. Princeton university press Princeton (1976) (Cited on page 9)

[185] Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. Springer (1996) (Cited on page 8)

[186] Stevenson, G., Knox, S., Dobson, S., Nixon, P.: Ontonym: a collection of upper ontologies for developing pervasive systems. In: Proceedings of the 1st Workshop on Context, Information and Ontologies. pp. 9:1–9:8. CIAO '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1552262.1552271 (Cited on page 16)

[187] Stikic, M., Larlus, D., Ebert, S., Schiele, B.: Weakly supervised recognition of daily life activities with wearable sensors. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33(12), 2521–2537 (2011) (Cited on page 20)

[188] Szathmary, L., Napoli, A., Valtchev, P.: Towards rare itemset mining. In: Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on. vol. 1, pp. 305–312. IEEE (2007) (Cited on page 80)

[189] Tazari, M.R., Furfari, F., Fides-Valero, Á., Hanke, S., Höftberger, O., Kehagias, D., Mosmondor, M., Wichert, R., Wolf, P.: The universAAL reference model for AAL. Handbook of Ambient Assisted Living 11, 610–625 (2012) (Cited on pages iv, v, 1, 2, 22, and 44)

[190] Trakhtenbrot, B.A.: Finite automata and monadic second order logic. Siberian Math. J 3, 101–131 (1962) (Cited on page 65)

[191] Uckelmann, D., Harrison, M., Michahelles, F.: An architectural approach towards the future internet of things. In: Architecting the Internet of Things, pp. 1–24. Springer (2011) (Cited on pages iv and 1)

[192] Van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. In: Proceedings of the 10th international conference on Ubiquitous computing. pp. 1–9. ACM (2008) (Cited on page 19)

[193] Viani, F., Rocca, P., Benedetti, M., Oliveri, G., Massa, A.: Electromagnetic passive localization and tracking of moving targets in a wsn-infrastructured environment. Inverse Problems 26(7), 074003 (2010) (Cited on pages 54 and 55)

[194] Viola, P., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004) (Cited on page 33)

[195] Wang, L., Gu, T., Tao, X., Chen, H., Lu, J.: Recognizing multi-user activities using wearable sensors in a smart home. Pervasive and Mobile Computing 7(3), 287–298 (2011) (Cited on page 18)

[196] Weijters, A.J., van der Aalst, W.M.: Rediscovering workflow models from event-based data using little thumb. Integrated Computer-Aided Engineering 10(2), 151–162 (2003) (Cited on pages 90 and 91)

[197] Wen, L., van der Aalst, W.M.P., Wang, J., Sun, J.: Mining process models with non-free-choice constructs. Data Min. Knowl. Discov. 15(2), 145–180 (2007) (Cited on page 90)

[198] Westergaard, M.: Better algorithms for analyzing and enacting declarative workflow languages using ltl. In: BPM. pp. 83–98 (2011) (Cited on page 92)

[199] Wilke, T.: Classifying discrete temporal properties. In: STACS 99. pp. 32–46. Springer (1999) (Cited on page 65)

[200] William, C., et al.: Fast effective rule induction. In: Twelfth International Conference on Machine Learning. pp. 115–123 (1995) (Cited on page 12)

[201] Yang, J.Y., Wang, J.S., Chen, Y.P.: Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. Pattern recognition letters 29(16), 2213–2220 (2008) (Cited on page 19)

[202] Ye, J., Coyle, L., Dobson, S., Nixon, P.: Ontology-based models in pervasive computing systems. The Knowledge Engineering Review 22(4), 315–347 (2007) (Cited on page 15)

[203] Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: A review. Pervasive and Mobile Computing 8(1) (2012), http://www.sciencedirect.com/science/article/pii/S1574119211000253 (Cited on pages iv, viii, 1, 7, and 14)

[204] Yun, H., Ha, D., Hwang, B., Ho Ryu, K.: Mining association rules on significant rare data using relative support. Journal of Systems and Software 67(3), 181–191 (2003) (Cited on page 80)

[205] Zadeh, L.: Fuzzy sets. Information and control 8(3), 338–353 (1965) (Cited on page 8)

[206] Zhao, T., Nevatia, R., Wu, B.: Segmentation and tracking of multiple humans in crowded environments. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 1198–1211 (2008) (Cited on page 27)

[207] Zichermann, G., Cunningham, C.: Gamification by design: Implementing game mechanics in web and mobile apps. " O'Reilly Media, Inc." (2011) (Cited on page 74)

[208] Zickler, C., Riccio, A., Leotta, F., Hillian-Tress, S., Halder, S., Holz, E., Staiger-Sälzer, P., Hoogerwerf, E.J., Desideri, L., Mattia, D.: A brain-computer interface as input channel for a standard assistive technology software. Clinical EEG and Neuroscience 42(4), 236–244 (2011) (Cited on page xi)