Towards Context Consistency in a Rule-Based Activity Recognition Architecture

Tuan Anh Nguyen t.a.nguyen@rug.nl
Viktoriya Degeler v.degeler@rug.nl
Rosario Contarino cntrsr@gmail.com
Alexander Lazovik a.lazovik@rug.nl
Marco Aiello m.aiello@rug.nl

Distributed Systems Group, Johann Bernoulli Institute, University of Groningen
Nijenborgh 9, 9747 AG, The Netherlands

Abstract

Human activity recognition (AR) is a crucial research area for intelligent pervasive environments such as energy-smart buildings. In order to gain precise and fine-grained AR results, a system must overcome partial observability of the environment and noisy, imprecise, and corrupted sensor data. In this work, we propose a rule-based AR architecture that effectively handles multiple-user, multiple-area situations, recognizing real-time office activities.The proposed solution is based on an ontological approach, using low-cost, binary, and wireless sensors. We employ context consistency diagrams (CCD) as the key component for fault corrections. CCD is a data structure that provides a mechanism for probabilistical reasoning about the current situation and calculates the most probable situation at each moment of time even with the presence of inconsistencies, conflicts, and ambiguities in available sensor readings. The implementation of the system and its testing in a living lab environment show that CCD corrects at least 26% and up to 56.88% of faults in sensor readings, improving overall recognition accuracy by at least 6% and up to 15%, thus producing reliable recognition results from unreliable sensor data. The results also show that our system can run online, in a distributed manner, with the execution times of less than 5ms.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Keywords

Context-aware computing, activity recognition, context reasoning, context consistency.

SenSys'13, November 11–15, 2013, Rome, Italy.

1 Introduction

Activity recognition is considered a crucial research issue for intelligent pervasive environments. Its potential applications include ambient assisted living, intelligent workspaces, classrooms, or energy saving buildings, just to name a few. Most of these applications require derived high-level activities composed of multiple low-level (sensors) input. While early research focused on the monitoring and analysis of visual information, e.g., images and surveillance videos, as means to recognize activities, recent research has moved towards the use of multiple miniature dense sensors embedded within environments [5]. Invasive technologies, such as cameras or wearable tags, might not be desirable because they suffer from deployability, cost and privacy issues. Instead, a wireless network of simple sensors are better option for creating a low-cost and easy-to-deploy solution.

Regarding activity recognition algorithms, they can be broadly divided into two major strands. The first one is based on machine learning techniques, including both supervised and unsupervised learning methods, which primarily use probabilistic and statistical reasoning. The second strand of activity recognition algorithms is based on logical modeling and reasoning [5]. Ontology-based algorithms overcome the disadvantage of machine learning techniques, that is, the demand on a sufficiently large amount of labeled data for training in order to perform well. Instead, without any training data, it is still possible to obtain potential performed activities of the user in a given context by exploiting symbolic reasoning. Nevertheless, logical-based approaches for activity recognition also have some drawbacks. The weaknesses of logical-based approaches mainly lay in their infeasibility to represent the uncertainty of environment. In addition, ontology-based approaches lack learning ability. Thus in order to gain precise and fine-grained recognition results, such systems must overcome partial observability of the environment and noisy, imprecise, and corrupted sensor data.

In this work, we propose a rule-based activity recognition architecture that effectively handles multiple-user, multiple-area situations, recognizing real-time office activities. The proposed architecture uses wireless networks of low-cost, simple, binary sensors at physical layer to collect raw information from environment, such as acoustic sensors to detect the appearance of human voice, pressure sensors for chair occupancy, electricity measuring plugs to monitor power state of appliances (PC, coffee machine, projector, etc.), and

so on. For data pre-processing and fault correction, we employ context consistency diagrams (CCD) in our architecture. The CCD data structure provides a mechanism for probabilistical reasoning about the current situation and calculates the most probable situation at each moment of time even with the presence of inconsistencies, conflicts, and ambiguities in available sensor readings. With reliable context information provided by CCD, we apply an ontological approach for activity recognition. We go beyond simple modeling of activities, artefacts and locations in a taxonomical way, but we also express semantic relationships and constraints between these ontologies. By having a complete and reliable picture of the given context, symbolic reasoning can be performed, making it feasible to recognize possible activities of the users. As another advantage, our proposed system works in a distributed manner, bringing flexibility to the solution.

The proposed architecture has been implemented in our own offices at the University of Groningen and tested over a short period as a proof of concept. The experiments in three full days show that CCD corrects at least 26% and up to 56.88% of faults in sensor readings, improving overall recognition accuracy by at least 6% and up to 15%, thus producing a reliable recognition result from unreliable sensor data. The results also show that our solution is able to recognize six typical office activities (working at a desk with or without a PC, having a meeting/discussion, having a coffee break, and presence/absence) at two office activity areas (working room, and coffee corner) for two persons with average accuracy of more than 84.44%. In addition, that our system can run online, in a distributed manner, with the execution times of less than 5ms.

The remainder of the paper is organized as follows. Section 2 discusses the related work. In Section 3, we present the background knowledge as well as an overview of our system. Section 4 shows the general architecture of our solution and describes each component in details. Section 5 specifies the technologies we use for the actual implementation of our system and living lab setting. Our experiments are described and evaluated in Section 6. Finally, conclusions are given in Section 7.

2 Related Work

Activity Recognition

Growing real-world application needs in such areas as ambient assisted living and energy saving have driven increasing interest in activity recognition. We begin by reviewing the prominent work on using activity recognition as means to control smart buildings for energy-awareness. Much research has focussed on integrating user activity and behavior as a key element for building energy and comfort management systems [24],accordingly controlling various devices like artificial light, shades, HVAC devices, computers, etc., making energy saving buildings. In proposing the AIM system [2], Barbato *et al.* build user profiles by using a learning algorithm that extracts characteristics from the user habits in the form of probability distributions. Bayesian networks are also used in [13] to support prediction of user behaviour patterns. In OBSERVE [10], Erickson *et al.* construct a multivariate Gaussian model, and a Markov Chain model for predicting user mobility patterns in buildings. In [14], the authors propose a general method to predict the possible inhabitant service requests based on the use of the Bayesian network to predict the user's behavior. Exploiting algorithms based on fuzzy logic, in [12], a system able to learn users preferences, to predict users needs and to self-adjust system behavior when users change their habits is proposed. The iDorm [12] learns and predicts the user's needs ability based on learning and adaptation techniques for embedded agents. In [9], the authors propose a belief network for occupancy detection within buildings. To summarize, in energy saving buildings most recognition algorithms are based on machine learning techniques, including both supervised and unsupervised learning methods, which primarily use probabilistic and statistical reasoning. However, the disadvantage of probabilistic methods is that they require a large amount of labelled training and test data [5].

A second strand of activity recognition algorithms based on logical modeling and reasoning to automatically recognize complex context data such as human activities is also receiving growing attention. In particular, in the area of pervasive computing, the ontological language OWL–DL [16] has been used to build activity ontologies, and to recognize activities based on context data, for instance in [6], [27], [20]. These proposals are mainly concerned with the support of monitoring people for medical reasons. The application is thus rather different from the requirements considered by energy-smart buildings. Besides, experimental evaluations of the actual effectiveness of ontology-based activity recognition techniques are missing.

Context Consistency Diagrams

Detection and resolution of sensor errors has been the objective of many studies. Detection strategies for inconsistent sensor readings, in particular, are studied in the work of Xu and Cheung et al. [29, 30, 32]. They propose to detect context contradictions based on predefined constraints, and convert each constraint into a tree with constraint operators as vertices and contexts as edges. They use partial constraint checking (PCC) algorithm to find only those parts of the constraint tree, which are affected by a new context. Huang et al. [18] continues the work by proposing to check branches probabilistically, which makes the processing faster. These works aim at fast detection of inconsistencies, while the CCD structure, presented in this paper, concentrates on correct resolution, once inconstencies are found.

Bu et al. [4, 3] find inconsistencies by modelling context as RDF-triples using OWL-lite language. They propose to discard one of the conflicting contexts based on their relative frequency. Xu et al. [31] propose other discarding policies, among which are drop-latest, drop-all, drop-random, and drop-bad. The CCD approach, on the other hand, never discards inconsistent context, instead forming several possible situations probabilistically.

Kong et al. [19] propose to extend the OWL ontology with fuzzy membership to tolerate inconsistencies, but does not discuss a way to get useful information out of it.

Henricksen and Indulska [15] introduce classification of context properties and inconsistencies, but do not provide

precise algorithms for dealing with possible context inconsistencies. Lu et al. [22] provide a mechanism for detecting failures in context-aware applications and means to test such applications. Huang et al. [17] study the inconsistencies introduced by asynchronous arrival of concurrent events, and propose an algorithm to detect initial order of events. Using CCD we combine all sensor readings together during their lifetime, so the initial order of arrival is not important.

The Context Consistency Diagrams (CCD) were first presented in [7]. The work defined the data structure and its properties, presented algorithms for maintaining the CCD in real-time and explained, how to calculate the probability of certain situations. In [8] an additional structure, Reduced Context Consistency Diagrams (RCCD) was introduced, which allows to decrease the memory and processing time requirements by trading off some of its expressivity.

3 Background Knowledge and System Overview

Motivated by the challenge of saving energy in buildings by exploiting activity and location information of the users, in [23, 26], the authors provide an investigation on detecting office activities by using simple sensors with an ontology-based solution. In this paper, we apply a similar approach which uses ontological modeling, representation and reasoning to recognize multiple-area activities (working at a desk with or without a PC, having a meeting/discussion inside private offices, having a coffee break at the coffee corner, and the presence/absence at the monitored areas). Furthermore, we go beyond only a simple ontology-based solution but we also propose a complete activity recognition architecture. The architecture not only is an ontology-based solution for area-based activity recognition but it also can effective handle and correct faults in sensor readings. Thus our system can reliably recognize multiple-user, multiple-area situations, recognizing real-time office activities, providing inputs for building energy and comfort management systems.

3.1 Ontological Modeling for Office Activity

3.1.1 Things-Oriented and Ontology-Based Activity Recognition

Things-oriented activity recognition was first proposed in [33]. The idea is that things, especially artefacts, inherently have functions that enable users to do particular activities. Thus, enabled activities can be identified by the functions of things in the area. In addition, there is a high correlation between activities and contexts. Spatial contexts relate to activity areas. Artefactual contexts contain dynamic state changes of artefacts caused by activities, for example, the changes of doors state, or the changes in the power consumption of a PC, coffee machine, or microwave. Based on this approach, each activity is identified by two contextual entities: 1) spatial contexts, i.e., office activity areas and 2) artefactual contexts, i.e., involved artefacts. For instance, 'working with PC' is an activity that happens in the 'WorkingRoom'. Involved artefacts are working chair, PC, and PC monitor.

Based on things-oriented activity recognition, in order to perform activity recognition, we build OWL-DL ontological model of activities, spatial contexts of office activity areas, and artefactual contexts as well as the relations between them. The main ontologies of our model are `ActivityArea`, `Activity`, and `Artefact`. They are illustrated in Figure 1. Detailed analysis and design of our ontologies for office activities are discussed in [26]. Web Ontology Language (OWL) is an ontology language for the Semantic Web, developed by the World WideWeb Consortium (W3C)Web Ontology Working Group. OWL was primarily designed to represent information about categories of objects and how objects are interrelated–the sort of information that is often called an ontology [16]. OWL is theoretically based on the well-developed knowledge representation formalism of Description Logic [1].

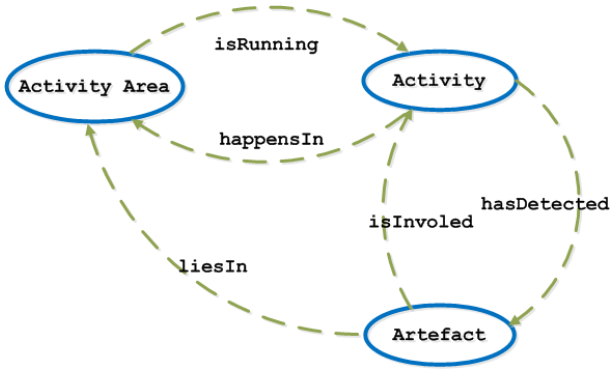


Figure 1. The core of office activity area modeling

In an ontological model, properties relate instances of concepts in the ontologies. For example, each activity area `isRunning` some activity, one activity `hasDetected` several artefacts, and each artefact `liesIn` an activity area. The set of activity areas, performed activities at each activity area, and artefactual context data defined in our ontologies are just the starting point for the recognition system and it is obviously non-exhaustive. However, we believe that these ontologies cover most popular office activity areas and performed activities. Furthermore, the ontologies are easily extensible to address more activity areas and their performed activities of the application domain.

3.1.2 The ActivityArea and Activity Ontologies

`ActivityArea`s are where some particular `Activity`s take place. Also, each `ActivityArea` contains a specific set of `Artefact`s. Our current ontology defines three descendants which are *WorkingRoom*, *MeetingRoom*, and *CoffeeCorner*.

The `Activity` ontology models the performed activities at each `ActivityArea`. The relationships and constraints between `ActivityArea` and `Activity` ontologies are `isRunning` and `happenedIn`. At a given moment, an `ActivityArea isRunning` one particular `Activity`. In other words, at that moment, an `Activity happensIn` an `ActivityArea`.

3.2 Context Consistency Diagrams

In many sensor deployment configurations there is a certain redundancy of sensors in order to get more reliable data. For example, in case there are two sensors, which respectively check whether the PC is working and whether the monitor is active, we can definitely say that if the first sen-

sor shows that the PC is off and the second one shows that the monitor is active, one of them must be faulty, as such a situation is not normally possible.

Such conflicts and inconsistencies in gathered data are common, as sensors are often noisy, impresice and their readings are easily corrupted. Several sensor readings when combined together form an inconsistency, which indicates that there is no single interpretation of the environment that is confirmed by all available sensor data.

In a presence of such conflict among sensor readings some research papers [4, 31] suggest to discard one of the readings that is deemed as incorrect one based on some heuristic strategy. Different heuristics are proposed, among which the removal based on relative frequency [4], drop-latest, drop-oldest, drop-all, or drop-random [31] strategies. However, premature removal of the information may cause more harm than good, especially in case the good reading is removed, while the wrong one is retained. A more cautious approach would be to retain all the information and probabilistically reason about the probability of several situations.

By using the information about sensor dependencies in a proper way we can at least partially detect and mitigate such conflicts, and try to correct faulty sensor readings. The Context Consistency Diagrams [7] is precisely a structure, designed to efficiently capture this information, and can be used with probabilistic reasoning to find the most probable situation at every moment of time.

The rules of sensor dependencies are entered to the system by its users. Every sensor reading is then preprocessed to find limitations it imposes on other sensors. The preprocessed sensor reading is called *a context*, as it represents a partial information about the environment. All contexts are combined into the context consistency diagram. All information is kept there for the duration of sensor readings' lifetime, thus even in case of inconsistency and different possible situations, further readings may help to refine the knowledge of the environment and make more informed decision about the correctness of certain sensor readings.

We will now briefly describe the important parts of the CCD as presented in [7], i.e. how the information is represented there and how it can be queried.

3.2.1 Environment and Context

DEFINITION 1 (ENVIRONMENT). *An* environment $\langle V, D\rangle$ *is defined by a set of context variables* $V = \{v_1, v_2, ..., v_n\}$. *Each variable* v_i *varies over a domain* $D_i = \{d_{i1}, d_{i2}, ..., d_{im_i}\}$ *with size* m_i.

Every sensor represents a single variable and the full set of possible sensor values represents a variable domain.

The single sensor reading may be not adequate to understand the full environment, but when we combine information from several sensor readings, and also take into account dependencies between sensors, which are defined as logical formulas, the combined information may give us a view of the part or of the full environment. Therefore, we define a notion of a context and an interpretation.

DEFINITION 2 (CONTEXT). *For a given environment* $\langle V, D\rangle$, *a* context *c is a valuation of all variables in V with a non-empty subset* D^c *of D.*

We represent a context by enumerating its possible context variables values: $D^c_0, \ldots, D^c_n$, or, alternatively, as $v_0 \in \{d_{0l}, \ldots, d_{0l}\}$. We write $c.v_i$ to refer to i-th variable of context c.

For example, for an environment, which contains two boolean variables *PC* and *Monitor*, the context may be $\{PC = \{true, false\};\ Monitor = true\}$, which means that the monitor is turned on and active, but we don't know anything about the state of the PC.

DEFINITION 3 (INTERPRETATION). *If all variables* v_i *are assigned one and only one specific value in* D_i, *a context is called an* interpretation.

If, for the same example, we add a logical formula $Monitor = true \Rightarrow PC = true$, then when we apply this rule to the context above, we know that the interpretation $\{PC = true;\ Monitor = true\}$ is probably the correct one.

Every variable in a context should have at least one possible value, as otherwise the context is impossible on practice. More than one value, on the other hand, represents an ambiguity and incomplete knowledge of the environment. Intuitively, every new sensor readings add additional knowledge about the environment, thus reducing the number of possible interpretations. Faulty sensor readings can be detected when impossible situation is created, i.e. when there is no interpretation, which is consistent with all available sensor readings.

Formally we define the notion of consistency as follows:

DEFINITION 4 (CONSISTENCY). *A set of contexts* $C = \{c_k\}$ *is* consistent *if there exist at least one interpretation* x : $x.v_i = d_{ij_i}, \forall i \in 1..n$ *such that* $d_{ij_i} \in c_k.v_i,\ \forall c_k \in C,\ \forall i \in 1..n$. *A set of contexts is* inconsistent *otherwise.*

Additionally, we define two relations over contexts:

Inclusion: $c_1 \subset c_2$ iff $\forall i \in 1..n :\ c_1.v_i \subset c_2.v_i$ Inclusion can be viewed as a relation of a more precise and less precise contexts. If $c_1 \subset c_2$ then context c_1 is more precise, than c_2, in other words, each variable of c_1 contains less values that are possible.

Intersection: $c_u = \bigcap_{j=1}^{k} c_j = c_1 \cap c_2 ... \cap c_k$ iff $\forall i \in 1..n$: $c_u.v_i = c_1.v_i \cap c_2.v_i ... \cap c_k.v_i$ An intersection of inconsistent contexts always equals to $\varnothing$. An intersection of consistent contexts is a context, that is at least as precise, that any of the originals: $\forall j \in 1..k\ \ c_u \subseteq c_j$.

3.2.2 CCD reasoning

We will now introduce the Context Consistency Diagram itself. The structure is essentially a compact representation of all possible interpretations of the environment, given the current sensor readings. The CCD is a directed acyclic graph, with a "full domain" context at the top (meaning "no information" is available) and more restrictive and knowledgeable contexts closer to the bottom.

If all sensor readings are consistent with each other, the CCD will have only one leave, which represents the most probable situation. If errors in sersor readings cause them to become inconsistent, several leaves will appear in the diagram. In this case we use probabilistic reasoning as described in [7] (we refer a reader to this paper for details) to decide, which leaf represents the most probable interpretation. We may also use different weights of sensors to show

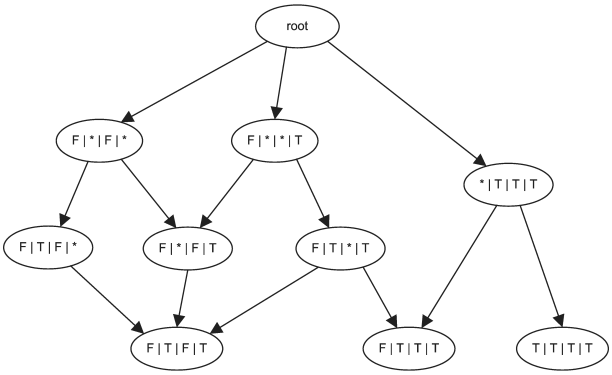


Figure 2. CCD example. Every node represents the context in [LCD|PC|PIR|PR] format. * means no value restrictions.

the trustworthiness of every sensor, because different sensors are more or less likely to give an erroneous result.

Formally the CCD is defined as follows:

DEFINITION 5 (CONTEXT CONSISTENCY DIAGRAM). *Given an environment $\langle V, D\rangle$ and a set of contexts $C_0 = \{c_k\}$, $k \in 1..N$, a* context consistency diagram (CCD) *is a tuple $G = \langle C, E, r\rangle$, where:*

- *$r = D$, is a special context, the* root;
- *$C = C_0 \cup C_u \cup r$ where C_u is the full set of intersections of a power set of C_0;*
- *$E \subseteq C \times C$, such that $(c_2, c_1) \in E$ iff $\exists c_1, c_2 \in C : c_1 \subset c_2$ and $\nexists c_m \in C : c_1 \subset c_m \subset c_2$.*

Example.

Let's assume we have an environment (which is a very small subset of the environment presented in Sections 5 and 6) with four boolean variables (with different weight): *PC* with weight 1, *LCD* with weight 1, PIR for (*K*)eyboard with weight 0.7, and (*PR*)essure on the chair with weight 0.8.

We need to establish rules of their dependency. First of all, the monitor cannot be turned on if PC is off. Next, if the PIR near keyboard detects movement, it must be typing or moving a mouse, and this mean the chair is occupied. And finally, the monitor is designed to turn off after a minute of inactivity, so it's only active is someone is present and actively works with PC (thus uses keyboard or mouse). So the rules are the following:

$$LCD = true \Rightarrow PC = true$$

$$PIR = true \Rightarrow PR = true$$

$$LCD = true \Rightarrow PIR = true \wedge PR = true$$

When we receive a sensor reading, we apply rules to it to obtain an extended context. For example, if we receive a reading $PIR = false$, the context after all rules are applied is $\{LCD = false; PIR = false\}$. Sometimes it is possible to have more than one context, for example for the reading $LCD = false$ we still need to account for the second rule, which gives us two contexts: $\{LCD = false; PIR = false\}$ and $\{LCD = false; PR = true\}$.

Figure 2 shows the CCD constructed if the following four sensor readings are received: $LCD = true; PC = true; PIR = false; PR = true$. We have three possible situations, but they all have different weight. The $[F|T|T|T]$ has weight 1.8, while $[F|T|F|T]$ has weight 2.5, and $[T|T|T|T]$ has weight 2.8, which means that most probably the reading $PIR = false$ was an incorrect one.

4 System Architecture

To realize a reliable activity recognition system, taking advantage of CCD for fault correction as well as ontological modeling for office activity recognition, we design an architecture that goes from the hardware level of wireless networks of simple sensors for collecting raw information from environment up to the ontology-based activity recognition. The overall architecture is shown in Figure 3. At the bottom is the Physical Layer that includes wireless networks of simple sensors used to collect raw information of interest from environment, above which there is the Common Gateway that handles readings different networks of sensors. The Fault Correction component employs CCD as the key mechanism for sensor data pre-processing, detecting and correcting faults in raw data provided from the Physical Layer. The reliable data processed with CCD is essential for the Activity Recognition (on top), which recognizes low-level as well as high-level activities by reasoning using the ontologies created with *Protégé*[1], a graphical tool for ontology development that simplifies design and testing. There are *Ground Activities Ontology* for low-level activity recognition and *Generalised Activities Ontology* for high-level recognition process.

4.1 Wireless Sensor Networks

Wireless Sensor Networks provide the basic infrastructure for gathering the information on the usage of artefacts. Artefactual contexts are usually captured through various sensors. Each sensor can be used to monitor and report the wanted situations of an artefact in the environment. For example, pressure sensors are used to determine chairs' occupancy, Passive Infrared (PIR) sensors detect movement and thus presence, electricity measuring plugs indicate the power inflow of appliances. In addition, the sensors have embedded wireless chip that is sufficient to form a wireless mesh network around the gateway, providing a cost effective and dynamic high-bandwidth network, with a relatively stable topology.

4.1.1 Unified Gateway

We use several different Wireless Sensor Networks in our system to collect different types of environmental information, for example a ZigBee-based network of electricity measuring plugs and another IEEE 805.14.5-based network of simple sensors. Thus it is necessary to have a middle component which is capable to read information from different networks and join data read from them into only one message, with a specific format, realizing a first data processing, thus providing complete picture of monitored environment. The middle component works as a *Gateway* between WSNs and upper components. So we can adopt a unique format for communicating with upper components, sending messages with an external message dispatcher. In addition, the middle Gateway hides the complexity of the networks of wireless

[1] http://protege.stanford.edu/

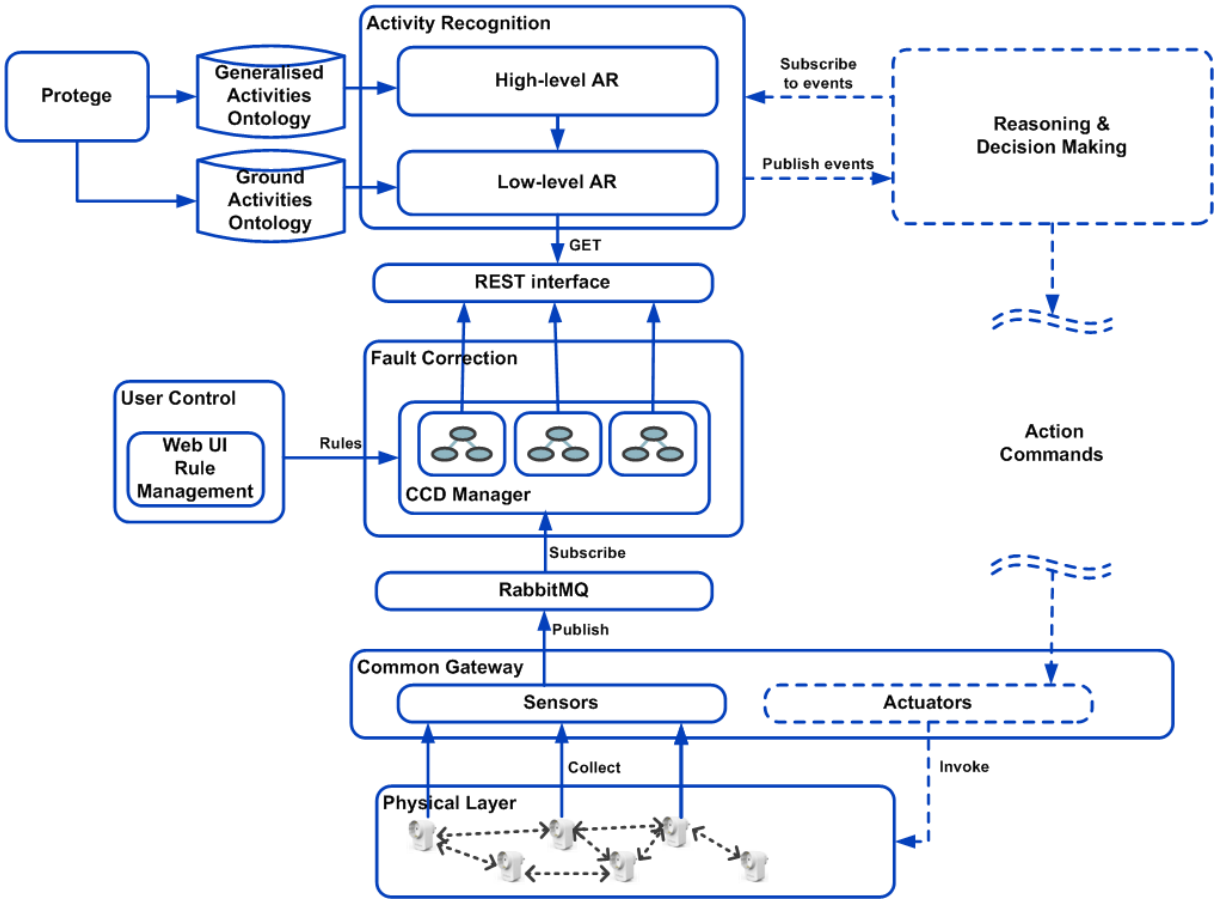


Figure 3. System architecture

sensors and makes components independent with each others. The gateway also include interfaces to handle the communication with upper components.

4.1.2 Interfaces to Upper Components

In our architecture, we propose to use a message broker in order to reduce coupling between low level components (WSNs and Gateway) and high level ones (i.e., activity recognition and fault correction). This makes it possible for the components of the system to run independently, in a distributed manner. The RabbitMQ[2] messaging framework is chosen chosen in our architecture because RabbitMQ is a complete and highly reliable enterprise messaging system based on the emerging AMQP[3] standard. RabbitMQ is robust messaging for applications, and runs on all major operating systems.

4.2 CCD component

The CCD component contains two main parts: the Web User Interface, which can be used by users of the building to control the rules and variables, and the CCD Manager, which actually creates and manages CCDs, gets sensor readings from the RabbitMQ and sends the results of correction to the Activity recognition component.

[2] http://www.rabbitmq.com/

[3] http://www.amqp.org/

The Web UI is quite straightforward. There are three main parts of the component that must be controllable/accessible by users. The first one is the variables configuration of the system. The user can access the see, which variables are currently present in a system, can add/remove additional variables, and can configure their information, such as the location of a device, the sensor weight, the available states, etc.

The second part which the user can control is the set of rules for the system. The rules are entered as formulas in predicate logic, where every atomic predicate is in a form $v_i = d_{ij}$ or $v_i \in \{d_{ij_1}, d_{ij_2}\}$, and takes $true$ if the variable v_i currently has one of the specified values, and $false$ otherwise. The user has the ability to alter the existing rules, add new ones, or remove the obsolete ones.

Finally, the last part of the web user interface is the environment information. The most probable interpretation gets automatically calculated after every sensor readings update, as is always kept in the system and is available for automatic queries or to be shown to a user. In a sense, this is the same information which is presented to the Activity Recognition component, but shown in a human-readable way.

The CCD Manager component manages all system's CCDs and it a common entry point of all input and output information. Potentially, every system can have more than one Context Consistency Diagram. As the diagram only helps to

improve information about dependent variables, there should be a single CCD for every subset of dependent variables. The variables are dependent if there are rules which restrict certain values combinations of these variables, or if there is another variable, with which both variables are dependent. For example, variables for PC and monitor are dependent, because the monitor cannot be on if the monitor is off. And two different chairs in the same room are dependent, because if somebody is working with PC, at least one or the other chair should be occupied, thus there is another variable which combines the two chairs. But chair in one room is independent from the chair in another room, as there is no rules, no common variables which combine them.

So, for the full set of variables the CCD Manager first identifies the subsets of independent variables, and the corresponding rules. Then the CCD Manager creates a CCD structure for every such subset of variables. When a new variable or a new rule is added or removed via a user interface, the affected CCD is reconstructed.

The CCD Manager registers itself with the RabbitMQ server, and subscribes for the events of sensors which are represented as variables. When the CCD Manager receives a new sensor reading, it finds the appropriate CCD based on the variable and sends the reading to this CCD for addition. When the lifetime of a sensor reading is expired, the CCD Manager contacts the CCD in order to remove the reading. The algorithms to add and remove contexts to the CCD dynamically are presented in [7], thus we refer to this paper for details about the algorithms.

The CCD Manager provides a REST interface as well for the activity recognition component (or any other component which may be interested in corrected sensor data). When contacted, it generates a message with a JSON object that contains the current most probable interpretation.

4.3 Ontology of Office Activities

4.3.1 The Use of Sensors in Artefact Ontology

Things or artefacts are very important in our model as they are used to recognize the occurring activity at each area. Enabled activities can be identified by the current state of artefacts in the area. The idea is that, at any given moment, the recognition system checks the state of all artefacts in a specific activity area and reasons with the ontologies to recognize which activity is taking place. Therefore, it is vital to be able to monitor the states and functions of artefacts in the environment, using sensor technology or other forms of state acquisition. Artefactual contexts are usually captured through various sensors. Each sensor can be used to monitor and report the wanted situations of an artefact in the environment. For example, pressure sensors are used to determine chairs' occupancy, Passive Infrared (PIR) sensors detect movement and thus presence, electricity measuring plugs indicate the power inflow of appliances. Each sensor in the `Artefact` ontology `liesIn` a specific `ActivityArea` and it `isInvolved` in a particular `Activity`. Our ontologies are detailed in [25]. Sensors used in the ontology are presented in Section 5. Performed activities at each activity area are recognized by ontological reasoning based on the checking of the state of all sensors involved. The descriptions of activities performed at each of the common office activity areas and sensors involved are illustrated in Table 1. The details of our activity recognition algorithm are presented in the next section.

4.4 Real-time office activity recognition

An activity in the ontologies is a concept described by a number of properties that specify relationships between the activity and other contextual entities [6]. As one example, *Working with PC* is defined as a *Working* activity taking place when the working chair, PC and PC monitor are being used. The location of the activity is inferred from the locations of the involved artefacts, in this case, the activity location is *Working Room* as the location of all artefacts is *Working Room*. This activity definition is described in OWL-DL as

$$\begin{aligned} &WorkingwithPC \sqsubseteq Working \sqcap \\ &\exists involvingArtefact.WorkingChair \sqcap \\ &\exists involvingArtefact.Computer \sqcap \\ &\exists involvingArtefact.ComputerMonitor \end{aligned}$$

At individual time points, our algorithm takes as inputs multiple sensor observations and the terminological part of the ontology. The output of the algorithm is an array A whose elements are occurring activities at corresponding office areas.

To be more precise, Algorithm 1 provides the formalization of the classification procedure. Multiple sensor observations are stored in a vector V of sensor data. Data is collected from sensors/plugs periodically and the vector V is updated with the data and a time stamp (e.g. each minute, each five minutes, etc.). The implemented time synchronization mechanism (Packet-level Time Synchronization [28]) guarantees that sensor data time stamps are accurate up to $\pm(T_r/2 - min)$ (where T_r is the round trip time and min is an estimated minimum time for sending any message). For a single hop network, the accuracy is very high, given the small value of T_r, especially when compared with the size of the time interval. By using reliable messaging, the vector V is also guaranteed to eventually get all sensor readings for each time interval; moreover, message omissions due to packet losses are handled by considering the current state valid up until a new message arrives.

The algorithm works as an infinite cycle that, at the end of each time interval, reads all the pushed sensor data from V and inputs the data to the ontological reasoner. The reasoner also takes the terminological part of the ontologies to derive the specific activities performed at every office activity area in the current time interval.

To illustrate the algorithm we consider the following scenario. Suppose that electricity measuring plugs attached to the projector inside a meeting room and to a PC, and to a PC monitor inside a private office report a high power state. At the same time interval, two pressure sensors inside the meeting room and a pressure sensor inside the private office are also activated. This means that a projector, three chairs, a PC, and a PC monitor, as instances of `Artefact`, are used in some activities. As each `Artefact` `liesIn` some `ActivityArea`, it is possible to infer the location of involved artefacts, i.e., two chairs and a projector are in the meeting room, another chair, a PC, and a PC monitor are in private office. As the `Artefact` ontology is the

Table 1. Performed activities at each activity area

Office area	Performed activities	Definition	Involved artefacts
WorkingRoom	Working with PC	User is using PC, PC and monitor are turn on	Chair, PC, and monitor
	Working without PC	User is sitting at the desk but PC and monitor are turned off	Chair
	Having a meeting or discussion	Two or more users are discussing	Chairs, human voice detector, PC may be involved
	Presence	User is active in the room but no further specific activity is recognized	Movement detector
	Absence	User is absent from the room	No artefacts involved
CoffeeCorner	Having a coffee break	Users are having lunch/coffee or just a break	Coffee machine, microwave, movement detector
	Absence	There is no one at the CoffeeCorner	No artefacts involved

range of the `hasDetected` property, it can be inferred one `hasDetected` is assigned the values two chairs and a projector, which all have `ActivityArea` is meeting room. Since the `hasDetected` is used to describe the `Activity` ontology, it can be inferred that *Giving a presentation* is taking place inside the meeting room. Similarly, *Working with PC (inside a private office)* is recognized by inferring from the activation state of a chair, a PC, and a PC monitor.

Algorithm 1 Activity Recognition

Require: V: vector of multiple sensor observations
Require: Ontologies and their properties
Ensure: A: Activities performed at all office activity areas
1: **for** each time interval **do**
2: *read*(V)
3: Convert sensor observations to corresponding properties in the ontologies
4: Use context ontologies to aggregate and fuse multiple sensor observations, performing the reasoning task in order to recognize activities performed at all office activity areas at the current time interval
5: Store the results in the output array A whose elements are happening activities at corresponding office areas
6: return A
7: **end for**

5 Implementation

We have implemented the proposed architecture in a prototype that we have deployed in our own offices. Next, we detail the realization of each component.

5.1 Living Lab Description

As the setup of living has an important influence on the rule set of CCD and activity reconginition ontology, we present here the description of our living lab. In our prototype implementation, we make a study in our own offices at the University of Groningen as the living lab. The test site consists of two working rooms that are occupied by two PhD candidates and one coffee corner. The layout of the three-room test site is illustrated together with the ZigBee mesh network of electricity measuring plugs attached to electrical appliances and multi-hop network of simple sensors in Figure 4. Activity areas and the artefacts inside them are listed in Table 2. In particular, we use electricity measuring plugs to detect the power state of five available devices (two PCs, two PC monitors inside two private offices, and a microwave at the coffee corner). Sensors are used to gather other crucial information, pressure sensor for chair's occupancy, acoustic sensor for human voice, and PIR sensor for motion detection.

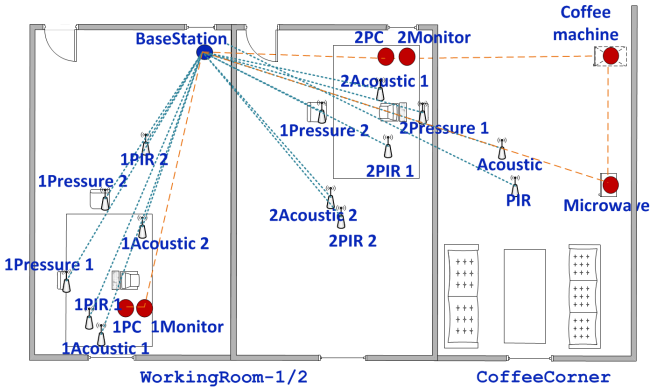


Figure 4. Living lab and setup

5.2 Wireless Sensor Networks

In our prototype implementation, we make a study in our own offices at the University of Groningen, using three types of sensors (pressure, acoustic, and PIR sensor) and electricity measuring plugs.

5.2.1 Simple Sensors

In `WorkingRoom`, chairs' occupancy is detected by pressure sensors. We place one PIR sensor near PC's mouse and keyboard to capture the hand's movements when user manipulates mouse/keyboard as the signs of using PC. Two acoustic sensors are placed near working table to detect human voice. One central ceiling mounted PIR sensor is placed in the center of each area (i.e., `WorkingRoom` and `CoffeeCorner`). Electricity measuring plugs are used to detect the power state of crucial appliances: PC and PC's monitor at `WorkingRoom` and microwave and coffee machine at `CoffeeCorner`.

We use IEEE 802.15.4 compliant wireless sensors based on the original open-source "TelosB" platform design developed and published by the University of California, Berkeley

Table 2. Experimental activity areas and their artefacts

Office activity areas	Artefacts
`WorkingRoom-1/2`	2 electricity measuring plugs for PC and PC's monitor, 1 pressure sensor for chair, 1 PIR sensor
`MeetingRoom`	1 electricity measuring plug for projector, 2 pressure sensors for chairs, 1 acoustic sensor, 1 PIR sensor
`CoffeeCorner`	2 electricity measuring plugs for microwave and coffee machine, 2 pressure sensor for chairs, 1 PIR sensor

("UC Berkeley"). The hardware is produced by Advantic Systems[4]. The sensors are equipped with ultra low-power 16-bit microcontroller MSP430 and run a low-power consumption management algorithm. The motes also have an extension interface that can be used to connect various sensor boards containing photo, temperature, humidity, pressure sensors, and accelerometers, magnetometers and microphones. The on-board PIR and Microphone are used together with FlexiForce pressure sensor. The motes are programmed in nesC and run on the TinyOS 2.1.1, a lightweight, low-power platform for embedded operating systems [21]. More details about the implementation of simple sensors can be found in [26].

5.2.2 *Electricity measuring plugs*

For electricity measuring plugs, we use Plugwise[5] products consisting of plug-in adapters that fit between a device and the power socket. The adapters can turn the plugged device on and off, and can at the same time measure the power consumption of the device that is attached. When attached appliances are not in use, i.e., either off or idle, the power consumption measured by the plug is very small, in between 1.0e-6 kWh to 1.0e-3 kWh. When attached appliances are actively in use, their real power consumptions are measured by the plug (1.5 kWh for the microwave, 0.09 kWh for PCs and PC monitors, 0.252 kWh for the projectors). The plug sends the real power consumption value of attached appliance to the base station every 30 seconds. The plugs are called "Circles" and they form a wireless ZigBee mesh network around a coordinator (called "Circle+"). The network communicates with the base station through a link provided by a USB stick device (called "Stick"). We implement a Gateway which is a process running in the background that report power state of controlled devices. It is written in Perl using xPL Protocol[6]. More details can be found in [11].

5.2.3 *Unified Gateway and Interfaces to Upper Components*

The gateway includes several modules as shown in the Figure 5. Proceeding from bottom to up we can find:

SensorReader This component handles the readings from simple sensors. The readings occur in asynchronous mode, every time a new message arrives from the network of simple sensors, SensorReader checks message senders and push new data to internal storage component called *SensorReading*.

PlugwiseWrapper This subcomponent is used to communicate with Plugwise network. The communication is possible thanks to the perl library[7]. The PlugwiseWrapper creates a new external process every time a reading is requested and at the end it parse the output obtained. Different than the SensorReader, in this case the reading is synchronous, therefore it is necessary to call a reading method to obtain data from the network.

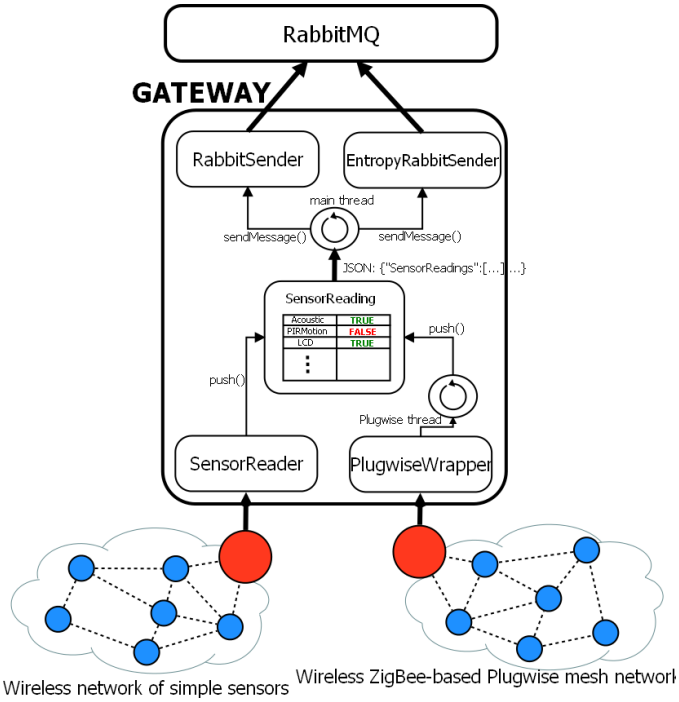


Figure 5. Subcomponents of Gateway and their interaction.

PlugwiseThread This is the thread responsible for the periodic reading of Plugwise plugs. Results obtained from network are used to update information inside the central data structure SensorReading.

SensorReading The gateway is designed to use a central information class, able to collect data from WSNs and provide the latest ones in an proper format, ready to be sent through RabbitMQ interfaces. SensorReading class contains a alphabetically sorted queue, in which information from WSNs are pushed. This class provide also a "flush" method to get current stored data in JSON format and clean the internal queue. In this way the structure is ready to store new values from sensors.

[4] `http://www.advanticsys.com/`
[5] `http://www.plugwise.com/`
[6] `http://xplproject.org.uk/`
[7] https://github.com/hollie/device-plugwise-perl

Table 3. Sensor weights in the CCD

Variable	Weight	Variable	Weight
1AcousticKeyboard	0.8	2LCD	1.0
1Acoustic	0.7	2PC	0.9
1LCD	1.0	2PIRKeyboard	0.5
1PC	0.9	2PIRMotion	0.5
1PIRKeyboard	0.5	2Pressure1	0.8
1PIRMotion	0.5	2Pressure2	0.8
1Pressure1	0.8	3Acoustic	0.7
1Pressure2	0.8	3Microwave	1.0
2Acoustic	0.7	3PIRMotion	0.5
2AcousticKeyboard	0.8		

RabbitSender As described before, our system uses RabbitMQ as message broker that communicates with different components. The RabbitSender class implements necessary methods to send a message to remote RabbitMQ instance.

EntropyRabbitSender This class is similar to RabbitSender, but it is created to simulate random errors in sensors readings. Every time it is going to send a message, for each variable it try to change the value from TRUE to FALSE according to a probability parameter. The results of this process are sent to an different queue from the one used for correct values.

main thread This is part of Gateway class and it is the main thread of this application. Its task is to flush periodically the SensorReading andsend the results to upper level components through the RabbitSender and the EntropyRabbitSender.

5.3 CCD component

The full environment, as modelled in CCD, contains 19 variables. Since not all sensors are equally trustworthy, we introduced the initial weights for all sensors, which is presented in Table 3. For example, the Plugwise are the most trustworthy ones, and they have the biggest relative weight, while there are relatively many situations when PIRs fail to detect movement or catch the occasional reflection of the sun in otherwise still environment, so the PIR sensors have the lowest weight. The pressure and acoustic sensors fall in between. The weights were assigned based on the empirical evaluation of sensors, and several smaller scale experiments to evaluate their accurance rate. It should be noted that in this setting the weights range in the region 0.5-1.0, but this does not corresponds to the actual probability of the sensor sending a true value. I.e. the sensors with the weight 1.0 still occasionally return erroneous values, while the sensors with the weight 0.5 give correct results much more frequently than 50% of the time. The weight of sensors only matters on relative scale, i.e. w.r.t. other sensors, and the absolute value of the sensor weight is not important.

We established eleven rules about the environment for the CCD to construct expanded contexts. For every office 5 rules were designed, and one more rule for the coffee corner. We will now describe all of them.

The first rule was already mentioned in the paper as example of sensor dependencies. It is not possible for the monitor to be on if the computer is turned off. Therefore the first rules:

$1LCD = true \Rightarrow 1PC = true$

$2LCD = true \Rightarrow 2PC = true$

The monitor was configured to turn off after 1 minute of inactivity, thus we could ensure that it is running only when people are actually working with the computer. A working person should be sitting on the chair and the PIR sensor which catches only the keyboard and mouse area should actually catch the respective activity. Also if the PIR Keyboard detects a movement of the mouse or above the keyboard, someone must be sitting and moving the mouse or typing.

$$1LCD = true \Rightarrow (1Pressure1 = true \vee 1Pressure2 = true) \wedge 1PIRKeyboard = true$$

$$2LCD = true \Rightarrow (2Pressure1 = true \vee 2Pressure2 = true) \wedge 2PIRKeyboard = true$$

$$1PIRKeyboard = true \Rightarrow (1Pressure1 = true \vee 1Pressure2 = true)$$

$$2PIRKeyboard = true \Rightarrow (2Pressure1 = true \vee 2Pressure2 = true)$$

The Acoustic Keyboard sensor can easily be enables on the same device which has the PIR Keyboard sensor. Thus we have an additional source of information for us to use. Therefore the Acoustic Keyboard sensor acts as a backup for the main acoustic sensor in the room:

$$1AcousticKeyboard = true \Rightarrow 1Acoustic = true$$

$$2AcousticKeyboard = true \Rightarrow 2Acoustic = true$$

During our experiment the acoustic sensors were configured to catch human voices. We also ensured the working atmosphere in the offices, i.e. no music running in the background when no one is around. Given the setting, the recongnition of a sound inside a room meant people speaking inside, which gives enough movement for the PIR sensors to recognize it. To ensure this the following rules were added:

$$1Acoustic = true \Rightarrow 1PIRMotion = true$$

$$2Acoustic = true \Rightarrow 2PIRMotion = true$$

When the meeting is taking place, the conversations are expected, which should be caught by the acoustic sensor:

$$1Pressure1 = true \wedge 1Pressure2 = true \Rightarrow 1Acoustic = true$$

$$2Pressure1 = true \wedge 2Pressure2 = true \Rightarrow 2Acoustic = true$$

Finally, for the coffee corner, if people are around and either speaking or using microwave, we expect the PIR sensor to detect them:

$$3Acoustic = true \vee 3Microwave = true \Rightarrow 3PIRMotion = true$$

The CCD is constructed based on those rules, and every sensor reading increases the probability only of those situations that are consistent with these rules and this reading.

The weight of the sensor is added to these situations (CCD leaves) with every reading and removed when the reading becomes obsolete. When contacted by AR component, the situation with the biggest weight is returned.

5.4 Ontologies and Recognition Module

The ontologies are developed using *Protégé*. Ontological reasoning is performed using the *HermiT*[8] inference engine, and its application programming interfaces (APIs) for the Java programming language. The recognition algorithm (Algorithm 1) is developed in Java and implemented as an online recognition system.

6 Evaluation

We have deployed the system in our own offices in order to assess the with such a system.

6.1 Experimental setup

Using the layout of Figure 4, the sensors just described in the Section 5, we perform an experiment in three separate days in a week, that is, daily from 10:00 AM to 17:00 PM, in March 26, March 29, and April 3, 2013 to verify the accuracy of the proposed architecture and approach in terms of activity classification, but also the error correction rate of CCD. The experiments concern the recognition of six different activities performed by two PhD candidates in `WorkingRoom-1`, `WorkingRoom-2`, and `CoffeeCorner`. During the experiment, the user takes accurate notes of actual activities happening in the office every minute, which is used as golden truth for the evaluation. Table 4 summarizes the real occurrences of activity instance at each activity area, that is then used as ground truth for evaluation. By using 1 minute time extent, the collected ground truth is composed of 1201 activity instances.

Table 4. Ground truth of the occurrences of activity instance at each activity area

Area	Activity					
	A	B	W	Wpc	M	C
`WorkingRoom-1`	70	232	256	640	3	0
`WorkingRoom-2`	129	112	139	754	67	0
`CoffeeCorner`	843	0	0	0	0	358

A = Absence; B = Being present; W = Working without PC; Wpc = Working with PC;M = Having a meeting/discussion; C = Coffee break

6.2 Evaluation Methodology

6.2.1 CCD vs. Averaged

In order to evaluate how CCD is able to detect and correct the errors in sensor readings and how CCD corrections affect the accuracy of our recognition solution, we implement another system which uses the same architecture except CCD-based fault correction component. Instead, to correct the faults in sensor readings, we implement a simple major voting algorithm, that is, the value of a sensor reading (TRUE or FALSE) is decided based on the most probable observation for a given combination of five readings collected in a minute. We call our algorithm *Averaged* in order to distinguish with CCD solution.

[8] `http://hermit-reasoner.com/`

The Gateway provides raw sensor readings for both CCD and Averaged component. After processing to correct the errors, CCD and Averaged provide corrected sensor readings as inputs for Activity Recognition component. The recognition results by using CCD and Averaged sensor readings are store to make comparisons.

6.2.2 Evaluation Metrics

Given an activity A, its FP, FN, and TP are defined as,

- FP (False Positive): the number of happening minutes of other activities recognized incorrectly as A.
- FN (False Negative): the number of happening minutes of A recognized falsely as not A.
- TP (True Positive): the number of happening minutes of A recognized truly as A.

We adapt to our case the following performance measures:

- Recall: the proportion of the time correctly recognized as A against the real happening time of A. Also called "True Positive Rate" - the percentage of positives the system recognizes correctly.

$$Recall = \frac{TP}{TP+FN} \tag{1}$$

- Precision: the rate of the real happening time of A over all the time recognized as A.

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

The overall success rate of the system is computed based on the following formula:

$$success\ rate = \left(1 - \frac{\sum_{\forall activities} FN}{T_{all}}\right) \tag{3}$$

in which, $\sum_{\forall activities} FN$ is the total minutes of wrong recognition. T_{all} is the number of minutes of total experimental period. We calculate the success rate for each activity as well as for each monitored area.

6.3 Results and Discussion

The overall success rates of the system for each monitored area are shown in Table 5. The first impression is that CCD significantly helps to correct the faults in sensor readings, thus the success rates notably increase, compared to Averaged ones. In particular, CCD helps to correct 90 minutes of wrong recognition at `WorkingRoom-1`, equivalent to 40%, improving the success rate for this room by 7.66%. The CCD corrections for `WorkingRoom-2` is even more significant with 46.83% of faults corrected, thus the accuracy of the recognition reaches 87.42%. That means it is 11.07% better compared to 76.35% returned by Averaged correction. One witness the most remarkable corrections at `CoffeeCorner`, where 82.60% of faults are corrected by CCD component. The hight number of corrections can be explained when we investigate the raw readings from sensors at the `CoffeeCorner`. Because CCD helps to correct PIR sensor readings with the following rule

$$3Acoustic = true \vee 3Microwave = true \Rightarrow 3PIRMotion = true$$

The improvement of success rates is also reflected in the Fig-

Table 5. Success rates of the system at each area

Area	$\sum_{\forall activities} FN$		% of errors
	Averaged	CCD	fixed
WorkingRoom-1	230	138	40.00%
WorkingRoom-2	284	151	46.83%
CoffeeCorner	46	8	82.60%
$T_{all} = 1201$ *minutes*			
Area	**Success rates**		
	Averaged	**CCD**	**Improvement**
WorkingRoom-1	80.85%	88.51%	7.66%
WorkingRoom-2	76.35%	87.42%	11.07%
CoffeeCorner	99.17%	99.33%	00.16%

ure 6, which shows comparison between actually happened activities in each monitored area and the recognized activities from Averaged result and CCD result.

Table 6 to Table 8 illustrate the result in more detail by showing recalls and precisions of all activities and all areas. In all three areas, CCD improves Recall of *Working with PC*, *Having a meeting/discussion*, and *Having a coffee break* activities, making sure that PC/meeting-related devices are working probably while the users are really working with the PC or having a meeting. These satisfy one of the important criterion of a smart building that is the user comfort has higher priority than energy saving. For example, Recall of *Working with PC* at WorkingRoom-1 improves from 94.53% to 99.06%, while this improvement at WorkingRoom-2 is from 81.69% to 97.61%.

Table 8. Results for coffee corner

(a) Confusion matrix

Recognized as →	C		A	
	Aver.	CCD	Aver.	CCD
Having a coffee break	**312**	**350**	46	8
Absence	0	0	**843**	**843**

(b) Precision/Recall

Activity	Precision (%)		Recall(%)	
	Aver.	CCD	Aver.	CCD
Having a coffee break	100	100	87.15	97.76
Absence	94.82	99.06	100	100

C = Having coffee break; A = Absence

7 Conclusion

Activity recognition that uses information directly from raw sensor readings is often susceptible to errors in the presence of erroneous sensor readings. Therefore ensuring the sufficient quality of sensor readings is very important when used for such activities. We discussed the novel approach to the activity recognition in intelligent pervasive environments. We showed that the ontology based activity recognition provides a high recognition rate, and that the Context Consistency Diagrams can be used together with the ontological activity recognition to successfully resolve most of the erroneous sensor readings and context inconsistencies, which corrects at least 26% and up to 56,88% of errors.

In our living lab we have implemented a full system and discussed the architecture design, and implementation issues. The experiments, performed in the living lab showed the the usage of CCD together with the ontological activity recognition increases the correct activity recognition rate by up to 15%.

The system can be efficiently maintained in real-time, with the execution times less than 50ms for every system update. The distributed manner of the system shows its high potential for scalability. We plan to continue the improvements and evaluation of the system, to test its applicability for bigger and even more complex settings of the whole buildings.

8 References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[2] A. Barbato, L. Borsani, A. Capone, and S. Melzi. Home Energy Saving through a User Profiling System based on Wireless Sensors. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, pages 49–54, New York, NY, USA, 2009. ACM.

[3] Y. Bu, S. Chen, J. Li, X. Tao, and J. Lu. Context consistency management using ontology based model. volume 4254 of *Lecture Notes in Computer Science*, pages 741–755. 2006.

[4] Y. Bu, T. Gu, X. Tao, J. Li, S. Chen, and J. Lu. Managing quality of context in pervasive computing. In *QSIC '06: Proc. Int. Conf. on Quality Software*, pages 193–200, 2006.

[5] L. Chen and I. Khalil. Activity Recognition: Approaches, Practices and Trends. In L. Chen, C. D. Nugent, J. Biswas, and J. Hoey, editors, *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 1–31. Atlantis Press, 2011.

[6] L. Chen, C. D. Nugent, and H. Wang. A Knowledge-Driven Approach to Activity Recognition in Smart Homes. *IEEE Trans. on Knowl. and Data Eng.*, 24(6):961–974, jun 2012.

[7] V. Degeler and A. Lazovik. Interpretation of inconsistencies via context consistency diagrams. In *9th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'11)*, pages 20–27, 2011.

[8] V. Degeler and A. Lazovik. Reduced context consistency diagrams for resolving inconsistent data. *ICST Transactions on Ubiquitous Environments*, 12(10-12), 2012.

[9] R. H. Dodier, G. P. Henze, D. K. Tiller, and X. Guo. Building Occupancy Detection through Sensor Belief Networks. *Energy and Buildings*, 38(9):1033–1043, 2006.

[10] V. Erickson, M. Carreira-Perpinan, and A. Cerpa. OBSERVE: Occupancy-based System for Efficient Reduction of HVAC Energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 258 –269, april 2011.

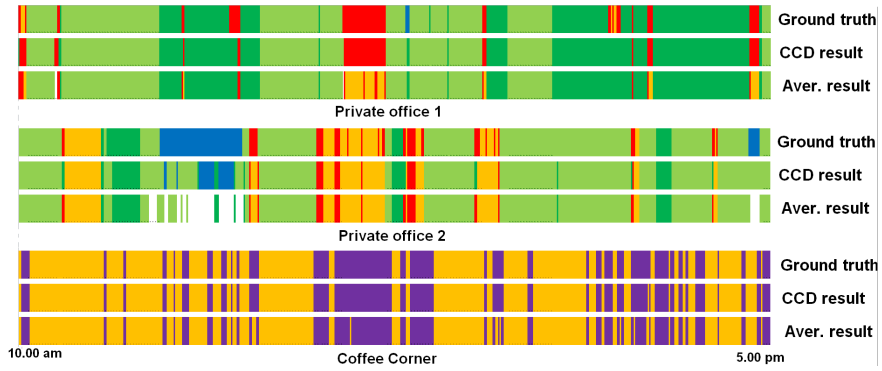
[11] I. Georgievski, V. Degeler, G. A. Pagani, T. A. Nguyen, A. Lazovik, and M. Aiello. Optimizing Energy Costs for Offices Connected to the Smart Grid . *IEEE Transactions on Smart Grid*, 3:2273–2285, 2012.

[12] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman. Creating an Ambient-Intelligence Environment Using Embedded Agents. *Intelligent Systems, IEEE*, 19(6):12 – 20, nov.-dec. 2004.

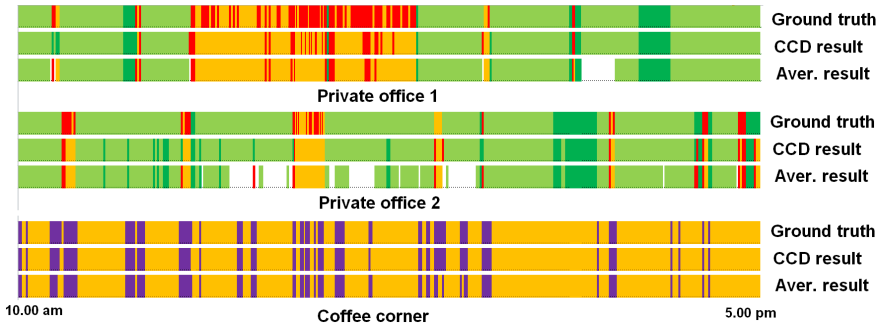
[13] C. Harris and V. Cahill. Exploiting User Behaviour for Context-Aware Power Management. In *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, volume 4, pages 122 – 130 Vol. 4, aug. 2005.

[14] L. Hawarah, S. Ploix, and M. Jacomino. User behavior prediction in energy consumption in housing using Bayesian networks. In *Proceedings of the 10th international conference on Artificial intelligence and soft computing: Part I*, ICAISC'10, pages 372–379, Berlin, Heidelberg, 2010. Springer-Verlag.

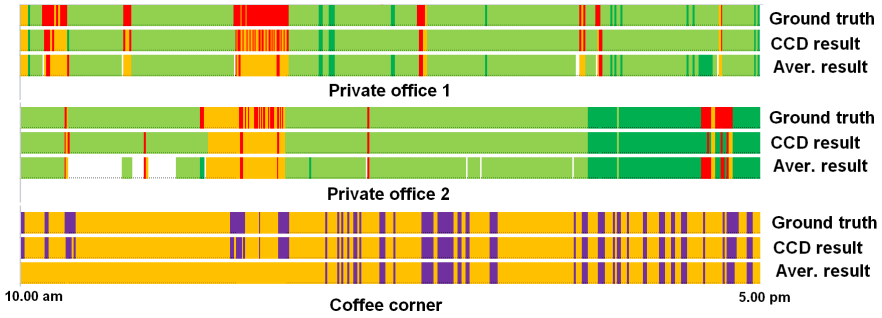
[15] K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Proc. of the 2nd IEEE Annual Conf. on Perv. Computing and Communications*, page 33, 2004.



(a) Result day 1



(b) Result day 2



(c) Result day 3

Figure 6. Experiment in three days

[16] I. Horrocks, P. F. Patel-Schneider, and F. V. Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1:2003, 2003.

[17] Y. Huang, X. Ma, J. Cao, X. Tao, and J. Lu. Concurrent event detection for asynchronous consistency checking of pervasive context. In *IEEE Int. Conf. Pervasive Computing and Communications*, pages 1–9, 2009.

[18] Y. Huang, X. Ma, X. Tao, J. Cao, and J. Lu. A probabilistic approach to consistency checking for pervasive context. In *EUC '08: Proc. IEEE/IFIP Int. Conf. on Embedded and Ubiquitous Computing*, pages 387–393, 2008.

[19] H. Kong, G. Xue, X. He, and S. Yao. A proposal to handle inconsistent ontology with fuzzy owl. In *Proc. WRI World Congress on CS and Inf. Eng.*, volume 1, pages 599–603, 2009.

[20] F. Latfi, B. Lefebvre, and C. Descheneaux. Ontology-Based Management of the Telehealth Smart Home, Dedicated to Elderly in Loss of Cognitive Autonomy. In *OWLED 2007 Workshop on OWL: Experiences and Directions*, 2007.

[21] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An Operating System for Sensor Networks. In *in Ambient Intelligence*. Springer Verlag, 2004.

[22] H. Lu, W. Chan, and T. Tse. Testing pervasive software in the presence of context inconsistency resolution services. In *Proc. Int. Conf. on Software engineering*, pages 61–70, 2008.

[23] T. A. Nguyen and M. Aiello. Beyond Indoor Presence Monitoring with Simple Sensors. In *Proc. 2nd Int. Conf. on Perv. & Embedded Comp. and Comm. Sys*, 2012.

[24] T. A. Nguyen and M. Aiello. Energy Intelligent Buildings based on User Activity: a Survey. *Energy and Buildings*, 56(0):244–257, 2012.

[25] T. A. Nguyen, A. Raspitzu, and M. Aiello. Ontology of Office Activities. Technical Report JBI preprint 2012-12-9, University of Groningen, 2012.

[26] T. A. Nguyen, A. Raspitzu, and M. Aiello. Ontology-based Office Activity Recognition with Simple Sensors for Energy Saving Buildings. *Journal of Ambient Intelligence and Humanized Computing*, 2013. To appear.

[27] D. Riboni and C. Bettini. COSAR: hybrid reasoning for context-aware activity recognition. *Personal Ubiquitous Comput.*, 15(3):271–289, Mar. 2011.

Table 6. Results for working room 1

(a) Confusion matrix

Recognized as →	Wpc		W		M		B		A		U	
	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD
Working with PC	**605**	**634**	8	2	0	0	2	4	0	0	25	0
Working without PC	1	9	**255**	**247**	0	0	0	0	0	0	0	0
Having meeting/discussion	1	1	2	2	**0**	**0**	0	0	0	0	0	0
Being present	1	6	12	17	0	0	**46**	**117**	168	92	5	0
Absence	0	0	3	5	2	3	0	0	**65**	**62**	0	0

(b) Precision/Recall

Activity	Precision (%)		Recall(%)	
	Aver.	CCD	Aver.	CCD
Working with PC	99.50	97.54	94.53	99.06
Working without PC	91.07	90.47	99.60	96.48
Having a meeting/discussion	0	0	0	0
Being present	95.83	96.69	19.82	50.43
Absence	27.89	40.25	92.85	88.57

Wpc = Working with PC; W = Working without PC; M = Having a meeting/discussion;B = Being present; A = Absence; U = Unrecognised minutes

Table 7. Results for working room 2

(a) Confusion matrix

Recognized as →	Wpc		W		M		B		A		U	
	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD
Working with PC	**616**	**736**	14	14	0	0	3	4	1	0	120	0
Working without PC	4	4	**130**	**131**	0	0	3	2	2	2	0	0
Having meeting/discussion	12	37	4	5	**0**	**25**	0	0	0	0	51	0
Being present	0	2	6	17	0	0	**44**	**32**	62	61	0	0
Absence	0	2	0	0	0	0	1	1	**127**	**126**	1	0

(b) Precision/Recall

Activity	Precision (%)		Recall(%)	
	Aver.	CCD	Aver.	CCD
Working with PC	97.16	94.23	81.69	97.61
Working without PC	84.41	78.44	93.52	94.24
Having a meeting/discussion	0	0	-	37.31
Being present	86.27	76.92	39.28	26.78
Absence	66.14	66.31	98.44	97.67

Wpc = Working with PC; W = Working without PC; M = Having a meeting/discussion;B = Being present; A = Absence; U = Unrecognised minutes

[28] TEP133. TinyOS Enhancement Proposals 133: Packet-level Time Synchronization, 2008.

[29] C. Xu and S. C. Cheung. Inconsistency detection and resolution for context-aware middleware support. In *Proc. Joint 10th European software engineering conference and 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 336–345. ACM, 2005.

[30] C. Xu, S. C. Cheung, and W. K. Chan. Incremental consistency checking for pervasive context. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 292–301. ACM, 2006.

[31] C. Xu, S. C. Cheung, W. K. Chan, and C. Ye. Heuristics-based strategies for resolving context inconsistencies in pervasive computing applications. In *Proc. 28th Int. Conf. Distributed Computing Systems*, pages 713–721, 2008.

[32] C. Xu, S. C. Cheung, W. K. Chan, and C. Ye. Partial constraint checking for context consistency in pervasive computing. *ACM Trans. Softw. Eng. Methodol.*, pages 1–61, 2010.

[33] N. Yamada, K. Sakamoto, G. Kunito, Y. Isoda, K. Yamazaki, and S. Tanaka. Applying Ontology and Probabilistic Model to Human Activity Recognition from Surrounding Things. *IPSJ Digital Courier*, 3:506–517, 2007.