Towards Context Consistency in a Rule-Based Activity Recognition Architecture

Tuan Anh Nguyen, Viktoriya Degeler, Rosario Contarino, Alexander Lazovik, Doina Bucur, Marco Aiello
Distributed Systems Group, Johann Bernoulli Institute, University of Groningen
Nijenborgh 9, 9747 AG, The Netherlands
Email: {t.a.nguyen,v.degeler}@rug.nl, contarino.rosario@gmail.com, {a.lazovik,d.bucur,m.aiello}@rug.nl

***Abstract*—Accurate human activity recognition (AR) is crucial for intelligent pervasive environments, e.g., energy-saving 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, wireless sensors. We employ context consistency diagrams (CCD) as a key component for fault correction. A CCD is a data structure that provides a mechanism for probabilistic reasoning about the current situation and determines the most probable current situation in the presence of inconsistencies, conflicts, and ambiguities in sensor readings. The implementation of the system and its evaluation in a living lab environment show that the CCD corrects up to 46.8% of sensor data faults, improving overall recognition accuracy by up to 11.1%, thus achieving reliable recognition results from unreliable sensor data.**

I. Introduction

Activity recognition (AR) is a core problem for intelligent pervasive environments. Its applications include, e.g., ambient assisted living, intelligent workspaces, and energy-saving buildings. These applications must recognize high-level activities from multiple low-level sensor input. Invasive technologies, such as cameras or wearable tags, are not desirable due to difficulty of deployment, cost and privacy issues. Recent research uses a number of dense, miniature wireless sensors embedded in the environment, which are a better option for creating a low-cost and easy-to-deploy solution.

AR algorithms can be broadly divided into two major categories. *Machine learning techniques*, including both supervised and unsupervised learning, primarily use probabilistic and statistical reasoning. Techniques based on *logical modeling and reasoning* [7], e.g., *ontology-based algorithms*, overcome a disadvantage of machine learning techniques, i.e., the need for a large amount of labeled training data, by exploiting symbolic reasoning. Nevertheless, ontology-based approaches for AR also have some drawbacks, such as their inability to represent the uncertainty of the 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.

We propose a *rule-based AR architecture* with embedded *sensor data fault correction* to recognize real-time office activities. We use wireless, low-cost, binary sensors to collect raw information from the environment, e.g., acoustic sensors to detect human speech, and electricity-measuring plugs to monitor appliances. For fault correction, we employ context consistency diagrams (CCD) [9], which provide a mechanism for probabilistic reasoning: it calculates the most probable current situation at each moment in time, in the presence of conflicts and ambiguities in available sensor readings. With reliable context information provided by CCD, we apply an ontological approach for AR. We model activities, artefacts and locations in a taxonomical way, and also express semantic relationships and constraints between these ontologies.

To test our architecture, a living lab was created on the premises of the University of Groningen, the Netherlands. Results from three days of experiments show that CCD corrects between 40% and 46.8% of faults in sensor readings, improving overall recognition accuracy by between 7.7% and 11.1%, thus producing a reliable recognition result from unreliable sensor data. Our solution is able to recognize six typical office activities (working at a desk with or without a PC, having a meeting, having a coffee break, and presence/absence) in three office activity areas (two working rooms and a coffee corner) for two persons with average accuracy over 88%.

The remainder of the paper is organized as follows. In Section II we present the background knowledge of CCDs. Section III shows the general architecture. Section IV specifies the technologies we use for our system implementation. Our experiments are described and evaluated in Section V. Section VI discusses the related work. Finally, conclusions are given in Section VII.

II. Context Consistency Diagrams

In many sensor deployments, some sensors are redundant to increase the reliability of data. Consider for example two sensors, which check (1) whether the PC is active, and (2) whether the monitor is active. We can definitely say that if (1) shows that the PC is off and (2) shows that the monitor is on, one of them must be faulty. Such conflicts and inconsistencies in sensor data are common, as sensors are often noisy, imprecise or not well calibrated. A number of sensor readings are called *inconsistent* if there is no single interpretation of the environment that is confirmed by all available sensor data.

By using such knowledge of dependencies between redundant sensors, we can partially detect and mitigate inconsistencies, and attempt to correct faulty sensor readings. The Context Consistency Diagram [9] is precisely a data structure designed to efficiently capture dependencies, and can be used with probabilistic reasoning to find the most probable current situation. The rules of sensor dependencies are entered by

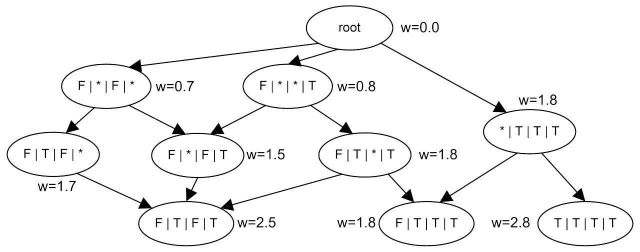

Fig. 1. CCD calculated for $LCD = T$, $PC = T$, $PIR = F$, $PR = T$. Every node represents a $[LCD|PC|PIR|PR]$ context. * denotes any value.

system users. Every sensor reading is then preprocessed to find the logical constraints it imposes on other readings. This preprocessed sensor reading is called a *context*, as it represents partial information about the environment. All contexts are combined and stored into the CCD. We refer a reader to our previous works [9], [10] for a formal definition of CCD and respective reasoning algorithms.

Example: Assume an environment (a subset of that presented in Sections IV and V) with four boolean variables of different trustworthiness, encoded with *weights*, w: PC with $w = 1$, LCD with $w = 1$, PIR for the keyboard with $w = 0.7$ and chair (PR)essure with $w = 0.8$. We need to establish their dependency rules. The monitor cannot be on if the PC is off. If the keyboard PIR detects movement, then the PC has a user, and the chair is occupied. Finally, the monitor is designed to turn off after 1 min of inactivity, thus it is only active if a user is present and works with the PC and keyboard. This gives three rules: 1) $LCD = T \Rightarrow PC = T$, 2) $PIR = T \Rightarrow PR = T$, and 3) $LCD = T \Rightarrow PIR = T \wedge PR = T$.

At every new reading, these rules are applied to obtain an *extended context*. E.g., for a reading $PIR = F$, the extended context after the rules are applied is $[LCD = F|PIR = F]$. More than one extended contexts are possible, e.g., applying rules to the reading $LCD = F$ returns two extended contexts: $[LCD = F|PR = F]$ and $[LCD = F|PR = T]$.

Figure 1 shows the CCD constructed (as described in details in [9]) if the following four sensor readings are received: $LCD = T; PC = T; PIR = F; PR = T$. Extended contexts from each of those readings receive a weight that corresponds to this sensor. The weight is then carried on to all children. At the end we have three possible situations. The combined weights for those situations are: $w = 1.8$ for $[F|T|T|T]$, $w = 2.5$ for $[F|T|F|T]$, and $w = 2.8$ for $[T|T|T|T]$, meaning that most probably the reading $PIR = F$ was incorrect.

III. System Architecture

The overall architecture is shown in Figure 2. The *Physical Layer* includes wireless sensor networks (WSNs). The *Fault Correction* component employs CCDs for raw sensor data preprocessing, detecting and correcting faults. The resulted probabilistically corrected data is essential for the *Activity Recognition* component, which recognizes activities through an ontological reasoning process; the *Ground Activities Ontology* does low-level activity recognition, while the *Generalised Activities Ontology* does high-level activity recognition.

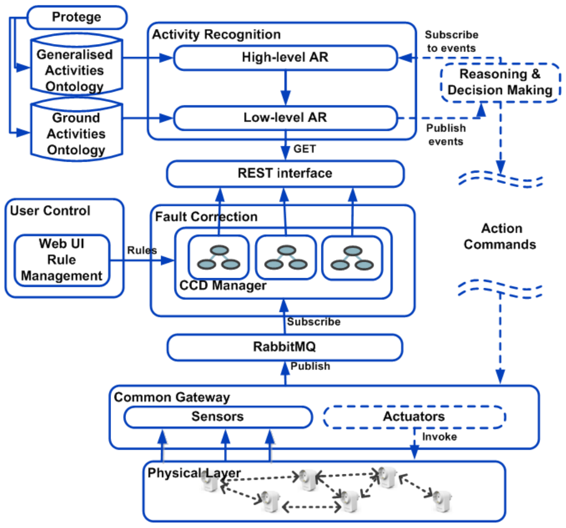

Fig. 2. System architecture

A. Wireless Sensor Networks

WSNs provide the basic infrastructure for gathering environmental information. Each sensor monitors an artefact in the environment. We use several WSNs in our system, e.g., a network of electricity-measuring plugs and a WSN of presence sensors. The *Common Gateway* merges sensor data from different networks into a common format, and decouples the Physical Layer from higher-level components.

B. Fault correction

Two components implement fault correction: the *Web UI*, employed by users of the building to control variables and dependency rules, and the *CCD Manager*, which receives raw sensor readings, creates and manages CCDs, and sends corrected readings to the Activity Recognition component.

The Web UI allows three user operations. (a) Through *variable configuration*, the user can see which variables v are currently configured in the system, can add/remove variables, and can configure their information, e.g., the location of a sensor, a sensor weight, or current context values $v = c$, with values c from a given domain D. (b) The user can also control the set of *dependency rules*. The rules are entered as formulas in predicate logic, where every atomic predicate is in the form $v = c$ or $v \in D$. (c) The user may also view *extended contexts*. The most probable extended context is automatically calculated after every sensor readings update, and is available for queries. This is the same information as passed to the AR component.

The CCD Manager manages all CCDs and is a common entry point of all input and output information. Every system can have more than one CCD, one for every *independent* subset of variables. The variables are dependent if there are rules which restrict certain value combinations of these variables, or if there exists another variable on which both variables are dependent. The CCD Manager first identifies the subsets of independent variables, and the relevant rule sets for each. Then, it creates a CCD for every subset. When new variables or rules are added or removed via the Web UI, the affected CCDs are reconstructed. When it receives a new sensor reading, it locates the appropriate CCD and sends the reading to it for addition. When a sensor reading has expired, the CCD is asked

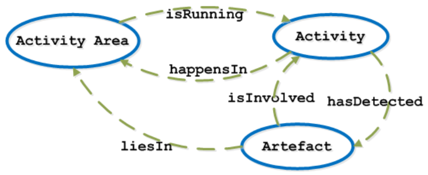

Fig. 3. The core of office activity area modeling

to remove the reading. The algorithms to add and remove contexts to/from a CCD dynamically are presented in [9]. When contacted by the Activity Recognition component, the Fault Correction component computes and sends the most probable current extended context.

C. Ontological Modeling for Office Activity

In order to perform activity recognition, we create an OWL-DL ontological model of activities, spatial contexts of office activity areas, and artefact contexts as well as the relations between them. The main ontologies of our model are `ActivityArea`, `Activity`, and `Artefact` [16]. They are illustrated in Figure 3. An `ActivityArea` is a location where a certain `Activity` happens. Each `ActivityArea` contains a certain set of `Artefacts`. The relationships between the `ActivityArea` and `Activity` ontologies are `isRunning` and `happensIn`.

We define two `ActivityAreas`: *WorkingRoom* and *CoffeeCorner*. In a WorkingRoom, there are five performed activities. (1) For *Working with PC*, the PC and LCD screen are turned on and the keyboard is used. The involved artefact list is: i) chair, ii) PC, iii) LCD screen, and iv) keyboard. (2) For *Working without PC*, a user is sitting at the desk but the PC and LCD are turned off. The artefact list is only i) user's chair. (3) In *Having a meeting*, two or more users are discussing. The involved artefacts are: i) users' chairs, ii) speech detector, and possibly iii) PC. (4) For *Presence*, a user is active in the room but no further specific activity is recognized, thus a movement detector (a PIR sensor) is the only artefact involved. At the CoffeeCorner, two activities are considered. (1) For *Coffee break*, users are having lunch/coffee or simply a break. Artefacts involved are i) microwave and ii) movement detector. (2) An *Absence* means that there is no one at the coffee corner, thus no artefacts are involved.

D. Office Activity Recognition

An activity specifies relationships between itself and other contextual entities [8]. E.g., *Working with PC* is defined as `is` a `Working` activity that `hasDetected` that the `chair`, `PC`, `screen`, and `keyboard` are used. The location of the activity is inferred from the locations of the involved artefacts; in this case, the activity `happensIn` the `Working Room|` area, as every artefact `liesIn` there.

At discrete time points, our algorithm takes as input sensor readings and the terminological part of the ontology. The output is an array A whose elements are occurring activities at corresponding locations. Algorithm 1 provides the classification procedure. Multiple sensor observations are stored in a vector V of sensor data. Data is requested periodically and the vector V is updated with the data and a time stamp. The algorithm implements a cycle; at the end of each time interval, it reads all sensor data from V and inputs the data to an ontological reasoner. The reasoner also uses the terminological part of the ontologies to recognize activities performed at every activity area in the current time interval.

Algorithm 1 Activity Recognition

```
Input: $V$: vector of multiple sensor observations
Input: Ontologies and their properties
Output: $A$: Activities performed at all activity areas
1: for each time interval do
2:     $read(V)$
3:     Convert sensor observations to ontological properties
4:     Aggregate sensor observations, performing reasoning to
       recognize activities performed at all activity areas
5:     $A_i = activity\ at\ area\ i^{th}$
6:     return A
7: end for
```

IV. Implementation

A. Living Lab Description

We have implemented the proposed architecture in a prototype living lab at the University of Groningen. The site consists of two working offices and one coffee corner. The layout of the three-room test site is illustrated in Figure 4, together with the topology of the ZigBee WSN of electricity-measuring plugs attached to electrical appliances, and the multi-hop WSN of binary sensors.

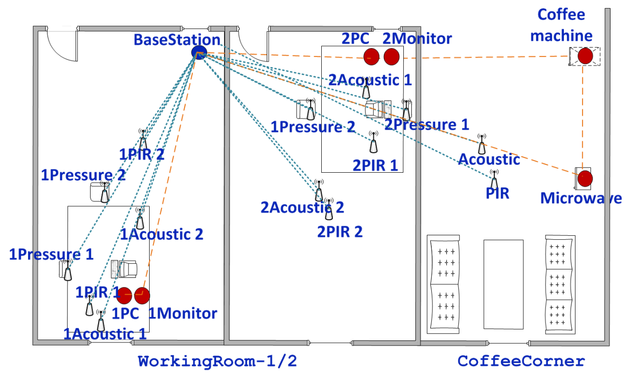

Fig. 4. Living lab: setup and wireless topology.

B. Wireless Sensor Networks

We use three types of sensors (pressure, acoustic, PIR) and electricity-measuring plugs. In a WorkingRoom, chair occupancy is detected by pressure sensors. We place one PIR sensor near the PC's mouse and keyboard to capture user hand movements. Two acoustic sensors are placed near the working table to detect human speech. One ceiling-mounted PIR sensor is placed in the center of each of the three areas. These are TelosB nodes running the TinyOS embedded operating system.

Electricity-measuring plugs are used to detect the state of appliances: the PC and PC screen in a WorkingRoom, and the microwave at the CoffeeCorner. These are Plugwise [3] sensors mounted between a device and the power socket. We implemented a component to communicate with the Plugwise network, using the xPL Protocol [5] and a Perl library for Plugwise [4]. More details can be found in [11].

A Common Gateway is implemented in Java, including a component to handle the readings from the binary sensors and a component to communicate with the Plugwise network. The gateway collects sensor readings from both sensor networks and merges the data into a common format, to be sent through the interfaces of a message broker. We chose RabbitMQ [18] as the message broker because it is a complete and highly reliable enterprise messaging system based on the emerging AMQP [1] standard.

C. The CCD component

The environment model contains 19 boolean variables. Each working room has 2 acoustic variables (general and keyboard), 2 PIRs (general and keyboard), 2 chair pressure variables, an LCD and a PC variable. The coffee corner has variables corresponding to the acoustic, PIR and microwave sensors.

Not all sensors are equally trustworthy: their weights are determined based on the empirical evaluation of sensor accuracy through several preliminary experiments. Our weights fall in the range 0.5-1.0, with a weight value simply representing a relative measure of that sensor's accuracy compared to the other sensors; an absolute weight value need not directly signify a certain level of sensor performance. The Plugwise sensors are most trustworthy, and they are assigned the highest relative weight. On the other hand, there are relatively many occurrences of false negatives and false positives for the PIR sensors, which, e.g., sometimes fail to detect movement, or misinterpret the occasional reflection of the sun in otherwise still environment; thus, the PIR sensors are assigned the lowest weight. The performance of the pressure and acoustic sensors is average.

We established 11 dependency rules about the environment, for the CCD to determine expanded contexts. For every office, 5 rules were designed; one more rule was added for the coffee corner. To describe the rules, we use variable names with wildcards, i.e. $xLCD$ refers to both LCD variables $1LCD$ and $2LCD$, and $xxPressure$ refers to all pressure sensor mounted on any chair in any office ($11Pressure$, $21Pressure$, etc).

Since it is not possible for the monitor to be on, if the computer is off, the first rule for working rooms is $xLCD = T \Rightarrow xPC = T$. The monitor was configured to turn off after 1 min of inactivity, to ensure that it is on when users are actively using the computer. A user is expected to be sitting on the chair, thus the PIR sensor monitoring the keyboard area should sense presence. Therefore: $xLCD = T \Rightarrow (x1Pressure = T \vee x2Pressure = T) \wedge xPIRKey = T$, and $xPIRKey = T \Rightarrow (x1Pressure = T \vee x2Pressure = T)$.

The acoustic keyboard sensor can easily be enabled on the same sensor node as the keyboard PIR sensor; therefore, the acoustic keyboard sensor acts as a backup for the main acoustic sensor in the room: $xAccKey \Rightarrow xAcoustic$. During our experiment, the acoustic sensors were calibrated to sense human speech. We also controlled the working atmosphere in the offices, i.e., we disallowed music running in the background without users present. Thus, the detection of sound in a room meant users were speaking in the room, which should produce sufficient motion for the PIR sensors to detect presence. To model this, the following rules were added: $xAcoustic = T \Rightarrow xPIRMotion = T$.

When a meeting takes place, conversation should be detected by the acoustic sensor: $x1Pressure = T \wedge x2Pressure = T \Rightarrow xAcoustic = T$. Finally, for the coffee corner, if users are either speaking or using the microwave, we expect the PIR sensor to detect presence: $3Acoustic = T \vee 3Microwave = T \Rightarrow 3PIRMotion = T$.

The CCD is constructed based on these rules, and every sensor reading increases the probability of those extended contexts that are consistent with these rules and this particular sensor reading. The weight of the sensor is added to these contextual situations (modelled by CCD leaf nodes) at every reading, and is removed when the reading becomes obsolete. When queried by the AR component, the extended context with the highest weight value is returned.

D. Ontologies and the Recognition Module

The ontologies are developed using *Protégé*. Ontological reasoning is performed using the *HermiT* [2] inference engine, and its Java APIs. The recognition algorithm (Algorithm 1) is developed and implemented in Java.

V. Evaluation

A. Experimental setup

Using the lab layout in Figure 4 and the sensors described in Section IV, we performed an experiment over three working days, daily from 10 AM to 17 PM, in March 2013. We aim to verify the performance of the proposed approach in terms of the accuracy of activity classification, but also in terms of the error correction rate of the CCD. The experiment concerns the recognition of six types of activities performed by two users in WorkingRoom-1, WorkingRoom-2, and the CoffeeCorner. During the experiment, the users take accurate ground-truth notes of actual activities happening at the site at 1-minute intervals, which are then used during the evaluation. Table I summarizes the ground truth of activity at each activity area. The collected ground truth is composed of 1201 activity instances.

B. CCD vs. Averaged

To evaluate how the CCD is able to detect and correct the errors in sensor readings and affect the accuracy of our recognition solution, we implement another system which uses the same architecture minus the CCD-based fault-correction component. Instead, to correct the faults in sensor readings

TABLE I. Ground truth of the occurrences of activity instance at each activity area

Area	**Activity**					
	A	P	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; P = Presence; W = Working without PC; Wpc = Working with PC; M = Having a meeting; C = Coffee break in this second system, we implement a simple majority-voting algorithm: the boolean value of a sensor reading is decided based on the most probable observation for a given combination of five readings collected in a minute. We call this algorithm *Averaged* to distinguish it from the CCD solution.

The gateway provides raw sensor readings for both the CCD and the Averaged components. After processing to correct the errors, both CCD and Averaged provide corrected sensor readings as inputs for the Activity Recognition component. The recognition results out of using both CCD and Averaged sensor readings are stored for later comparisons.

C. Results and Discussion

We denote by i) FN a false negative, i.e., a true activity failing to be recognized, ii) FP a false positive, i.e., a false activity incorrectly recognized, iii) TP a true positive, i.e., a true activity truly recognized. We adapt to our case the following performance measures. (1) *Recall* is computed as $Recall = \frac{TP}{TP+FP}$. (2) *Precision* is computed as $Precision = \frac{TP}{TP+FP}$. (3) *Success Rate* of the system is computed as $Success\ Rate = \left(1 - \frac{\sum_{act} FN}{T_{all}}\right)$, in which $\sum_{act} FN$ is the total number of minutes delivering false negative results, and T_{all} is the number of minutes in the experimental period. We calculate the success rate for each activity as well as for each monitored area.

The overall success rates per monitored area are shown in Table II. The first impression is that the CCD significantly helps to correct the faults in sensor readings, thus the success rates notably increase, compared to Averaged. In particular, the CCD helps to correct 90 minutes of wrong recognition in WorkingRoom-1, equivalent to 40%, improving the success rate for this room by 7.7%. The CCD correction for WorkingRoom-2 is even more significant, with 46.8% of faults corrected, and the accuracy of the recognition reaching 87.4%. That means it is 11.1% better compared to the 76.4% returned by Averaged correction. One witnesses the most remarkable corrections at CoffeeCorner, where 82.6% of faults are corrected by the CCD component. The high number of corrections can be explained when we investigate the raw readings from sensors at the CoffeeCorner: the CCD helps to correct PIR sensor readings using the rule: $\mathit{3Acoustic} = T \vee \mathit{3Microwave} = T \Rightarrow \mathit{3PIRMotion} = T$.

TABLE II. SUCCESS RATES OF THE SYSTEM AT EACH AREA

Area	$\sum_{act} FN$		%errors	Success rate		
	Aver.	CCD	fixed	Aver.	CCD	%Improved
WorkingRoom-1	230	138	40	80.9	88.5	7.6
WorkingRoom-2	284	151	46.8	76.3	87.4	11.1
CoffeeCorner	46	8	82.7	99.1	99.3	0.2

The experiment results also show that the performance of fault correction clearly is influenced by the weight assignment to sensors. Since this performance will vary with weight assignment, it is important to assign weights which reflect well the sensors' performance.

The improvement of success rates is also reflected in Figure 5, which compares actual activities in each monitored area with activities recognized by Averaged and CCD on the first day of the experiment. Table III illustrates the results in more detail by showing recalls and precisions of all activities at all areas. In all three areas, CCD improves the recall of the *Working with PC*, *Having a meeting*, and *Coffee break* activities. This satisfies one of the important criteria of an energy-saving building, i.e. that user comfort has higher priority than energy saving. For example, the recall of *Working with PC* at WorkingRoom-1 improves from 94.5% to 99.1%, while at WorkingRoom-2 it improves from 81.7% to 97.6%.

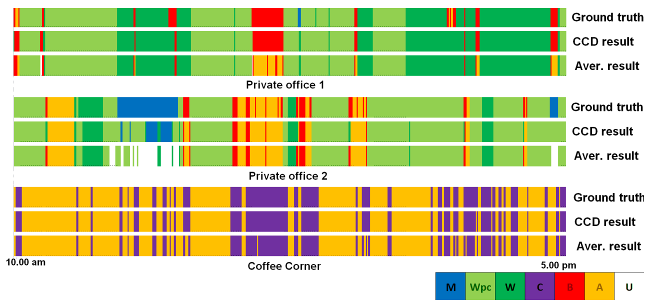

Fig. 5. Averaged vs. CCD results on the first experimental day

VI. RELATED WORK

Growing real-world needs for ambient-assisted living and energy saving applications have driven increased 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 [15], accordingly controlling various devices (artificial light, shades, HVAC devices, computers, etc.), for energy-saving buildings. Most recognition algorithms are based on machine learning techniques [15], including 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 [7].

Other recognition algorithms implement logical modeling and reasoning to automatically recognize complex contexts such as human activities. In particular, the ontological language OWL–DL [13] has been used to build activity ontologies, and to recognize activities based on context data, e.g., [8], [17]. These proposals are mainly concerned with monitoring people for medical reasons, and experimental evaluations of the effectiveness of ontology-based activity recognition techniques are missing.

Detection strategies for inconsistent sensor readings are studied in the work of Xu and Cheung et al. [20]. While this aims at fast detection of inconsistencies, the CCD structure concentrates on correct resolution once inconsistencies are found. Bu et al. [6] find inconsistencies by modelling context as RDF-triples using the OWL-lite language. They propose to discard one of the conflicting contexts based on their relative frequency. Xu et al. [19] 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.

TABLE III. RESULTS FOR WORKING ROOM 1, WORKING ROOM 2, AND COFFEE CORNER

(a) Confusion matrix (WorkingRoom-1/WorkingRoom-2/CoffeeCorner)

Recognized as →	Wpc		W		M		P		A		C		U	
	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD	Aver.	CCD
Working with PC	**605/616/-**	**634/736/-**	8/14/-	2/14/-	0/0/-	0/0/-	2/3/-	4/4/-	0/1/-	0/0/-	-/-/-	-/-/-	25/120/-	0/0/-
Working w/o PC	1/4/-	9/4/-	**255/130/-**	**247/131/-**	0/0/-	0/0/-	0/3/-	0/2/-	0/2/-	-/-/-	-/-/-	0/2/-	0/0/-	0/0/-
Having a meeting	1/12/-	1/37/-	2/4/-	2/5/-	**0/0/-**	**0/25/-**	0/0/-	0/0/-	0/0/-	0/0/-	-/-/-	-/-/-	0/51/-	0/0/-
Presence	1/0/-	6/2/-	12/6/-	17/17/-	0/0/-	0/0/-	**46/44/-**	**117/32/-**	168/62/-	92/61/-	-/-/-	-/-/-	5/0/-	0/0/-
Absence	0/0/-	0/2/-	3/0/-	5/0/-	2/0/-	3/0/-	0/1/-	0/1/-	**65/127/843**	**62/126/843**	-/-/0	-/-/0	0/1/-	0/0/-
Coffee break	0/0/-	0/2/-	3/0/-	5/0/-	2/0/-	3/0/-	0/1/-	0/1/-	-/-/46	-/-/8	**-/-/312**	**-/-/350**	0/1/-	0/0/-

(b) Precision and Recall (WorkingRoom-1/WorkingRoom-2/CoffeeCorner)

Activity	Precision (%)		Recall(%)	
	Aver.	CCD	Aver.	CCD
Working with PC	99.5/97.2/-	97.5/94.2/-	94.5/81.7/-	99.1/97.6/-
Working w/o PC	91.1/84.4/-	90.5/78.4/-	99.6/93.5/-	96.5/94.2/-
Having a meeting	0/0/-	0/0/-	0/-/-	0/37.3/-
Presence	95.8/86.3/-	96.7/76.9/-	19.8/39.3/-	50.4/26.8/-
Absence	27.9/66.1/94.8	40.3/66.3/99.1	92.9/98.4/100	88.6/97.7/100
Coffee break	-/-/100	-/-/100	-/-/87.2	-/-/97.8

Wpc = Working with PC; W = Working without PC; M = Having a meeting; P = Presence; A = Absence; C = Coffee break; U = Unrecognised

Kong et al. [14] propose to extend the OWL ontology with fuzzy membership to tolerate inconsistencies, but does not discuss a way to design recognition algorithms. Henricksen and Indulska [12] introduce a classification of context properties and inconsistencies, but do not provide precise algorithms for dealing with possible context inconsistencies.

VII. CONCLUSION

Activity recognition (AR) that uses raw sensor readings is often susceptible to errors caused by imperfect sensors. Ensuring sufficient quality for sensor readings is thus crucial for effective AR. We discussed a novel approach to AR in intelligent environments. We showed that the ontology-based activity recognition enhanced with Context Consistency Diagrams (CCD) can be used to successfully resolve most of the erroneous sensor readings and context inconsistencies, and achieve high recognition accuracy.

For our living lab, we have designed, implemented, and evaluated a complete pilot recognition system. The current system includes three offices, 19 sensors in two networks, and 11 logical dependency rules among the sensors' variables. It has execution times of less than 5ms for every data processing step, and the experiments showed that the CCD together with the ontological AR can correct up to 46.8% of errors and increases the correct activity recognition rate by up to 11.1%. Our solution is thus able to recognize six typical office activities with average accuracy of more than 88.5%. These results show the scalability of the proposed system when applied in real-world buildings. We plan to continue the improvements and evaluation of the system, to test its applicability in more complex settings and larger living labs.

ACKNOWLEDGEMENT

The work is supported by the EU FP7 Project GreenerBuildings, contract no. 258888 and the Dutch National Research Council Energy Smart Offices project, contract no. 647.000.004. Tuan Anh Nguyen is supported by the Vietnam International Education Development Program.

REFERENCES

[1] AMQP messaging system. http://www.amqp.org/, 2013.

[2] HermiT reasoner. http://hermit-reasoner.com/, 2013.

[3] Plugwise. http://www.plugwise.com/, 2013.

[4] Plugwise-Perl-lib. http://github.com/hollie/device-plugwise-perl, 2013.

[5] xPL Protocol. http://xplproject.org.uk/, 2013.

[6] Y. Bu, T. Gu, X. Tao, J. Li, S. Chen, and J. Lu. Managing quality of context in pervasive computing. In *Int. Conf. on Quality Software (QSIC)*, pages 193–200, 2006.

[7] 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.

[8] 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, 2012.

[9] V. Degeler and A. Lazovik. Interpretation of inconsistencies via context consistency diagrams. In *Int. Conf. on Pervasive Compt. and Comm. (PerCom)*, pages 20–27, 2011.

[10] V. Degeler and A. Lazovik. Reduced Context Consistency Diagrams for Resolving Inconsistent Data. *ICST Transaction on Ubiquitous Environments*, 12(10-12), 2012.

[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 Trans. on Smart Grid*, 3:2273–2285, 2012.

[12] K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Int. Conf. on Pervasive Compt. and Comm. (PerCom)*, page 33, 2004.

[13] 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.

[14] 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.

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

[16] 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*, To appear.

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

[18] A. Videla and J. J. Williams. *RabbitMQ in action*. Manning, 2012.

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

[20] C. Xu, S. C. Cheung, W. K. Chan, and C. Ye. Partial constraint checking for context consistency in pervasive computing. *ACM Trans. Software Engineering Methodology*, pages 1–61, 2010.