AGACY Monitoring: A Hybrid Model for Activity Recognition and Uncertainty Handling

Hela Sfar(✉), Amel Bouzeghoub, Nathan Ramoly, and Jérôme Boudy

CNRS Paris Saclay, Telecom SudParis, SAMOVAR, Évry, France
{hela.sfar,amel.bouzeghoub,
nathan.ramoly,jerome.boudy}@telecom-sudparis.eu

Abstract. Acquiring an ongoing human activity from raw sensor data is a challenging problem in pervasive systems. Earlier, research in this field has mainly adopted data-driven or knowledge based techniques for the activity recognition, however these techniques suffer from a number of drawbacks. Therefore, recent works have proposed a combination of these techniques. Nevertheless, they still do not handle sensor data uncertainty. In this paper, we propose a new hybrid model called AGACY Monitoring to cope with the uncertain nature of the sensor data. Moreover, we present a new algorithm to infer the activity instances by exploiting the obtained uncertainty values. The experimental evaluation of AGACY Monitoring with a large real-world dataset has proved the viability and efficiency of our solution.

Keywords: Smart home · Uncertainty · Ontology · Machine learning

1 Introduction

Nowadays, we face an emerging use of context-aware systems in various domains in order to ensure of end-user well being and quality of life. The smart homes are a trending context aware applications. In smart homes, context information about the user context is gathered and used to monitor and track their activities. Smart environments rely on sensors to monitor the interaction between the users, objects, and the environment. However, in real world applications, sensor data are not always well-aimed and precise [10] due to hardware failure, energy depletion, etc. Therefore, a context-aware system should be sensitive enough to the missing or imprecise sensor data in order to make the right decisions. To recognize activities and handle sensors' uncertainty, the main solutions can be generally classified as *data driven*, *knowledge based*, and *hybrid* approaches. Data driven methods apply different supervised machine learning techniques to classify sensor data into activities based on the given training data. For example, Hidden Markov Models (HMM) [12] and Support Vector Machines (SVM) [13] are two well-known classifiers. Although this group of methods is suited to handle uncertainty and to deal with a broad range of sensors, it needs a large amount

E. Blomqvist et al. (Eds.): ESWC 2017, Part I, LNCS 10249, pp. 254–269, 2017.
DOI: 10.1007/978-3-319-58068-5_16

of training data to set up a model and estimate its parameters [10]. On the other hand, knowledge based methods use ontologies and reasoning engines to infer proper activities from current sensors input. Despite their powerful semantic representation of real world data and their reasoning capabilities, their use is restricted to a limited number of sensors: when big number of sensors are used then the manual creation of the ontology will be hard and painful. Therefore, more the number of sensor increased more the conception of the ontology and the reasoning part becomes difficult. The number of sensors in which the ontology is limited can be set in empirically. Given the limitations of both data driven and knowledge based approaches, combining them is a promising research direction as it was stated in the analysis done in [10]. Intuitively, a hybrid approach takes the "best of both worlds" by using a combination of methods. Such an approach is able to provide a formal, semantic and extensible model with the capability of dealing with uncertainty of sensor data and reasoning rules [10]. Therefore, proposing hybrid models has been the motivation of recent works including [3,11,14]. Nevertheless, the lack of sensor data uncertainty consideration is the main drawback of the aforementioned approaches. To overcome this limitation, this paper proposes a new hybrid model combining data driven and knowledge based methods for activity recognition and sensor data uncertainty handling. The main contributions of the paper are as follows:

1. Introduction of the AGACY Monitoring hybrid model, available online, that integrates knowledge based and data driven techniques for activity recognition. This novel approach handles uncertain sensor data and exploits these uncertainty values to compute the produced activities uncertainty values.
2. Improvement of an existing method for feature extraction [8] in order to deal with more time distant actions and their uncertainty values.
3. Invention of new algorithm for current activity instances inferring called AGACY.

The rest of this paper is organized as follows: Sect. 2 discusses related work about hybrid models for activity recognition, Sect. 3 presents the architecture of AGACY Monitoring model. Sections 4 and 5 provide more details about each layer in the architecture. Section 6 reports experimental results. Finally, Sect. 7 concludes the paper.

2 Related Work

The combination of data driven and knowledge based methods for activity recognition has been a recent topic of interest. Therefore, few hybrid activity recognition systems have been proposed in the literature.

COSAR [4,15] is a context-aware mobile application that combines machine learning techniques and an ontology. As a first step, the machine learning method is triggered in order to predict the most probable activities based on a provided training data. Then, an ontological reasoner is applied to refine the results by selecting the set of possible activities performed by a user based on his/her

location acquired by a localization server. Despite the fact that the sensor data are supposed to be certain, COSAR deals with the uncertainty of the transformation of the localization from a physical format to a symbolic one. Another hybrid model that combines a machine learning technique, an ontology, and a log-linear system has been proposed in [14]. The aim of this approach is to recognize a multilevel activity structure that holds 4 levels: atomic gesture (Level 4), manipulative gesture (Level 3), simple activity (Level 2), and complex activity (Level 1). The atomic gestures are recognized through the application of a machine learning technique. Moreover, using a probabilistic ontology defined by the log-linear, and standard ontological reasoning tasks, the manipulative gestures, simple activities, and complex activities are inferred. Each level is deduced based on a time window that contains elements from the previous level. Even though the work in [14] is similar to the previous one regarding the absence of sensor data uncertainty's handling, the inference of the 4 levels activities is based on a probabilistic reasoning that represents a sort of uncertainty.

FallRisk [11] is another pervasive system that combines data driven and knowledge based methods. Its main objective is to detect a fall of an elderly person living independently in a smart home. FallRisk is a platform that integrates several systems that use machine learning methods for fall detection. It filters the results of these systems thanks to the use of an ontology that stores the contextual information about the elderly person. The main advantage of this system is that it is extensible to integrate several fall detection systems. Moreover, the contextual information of the elderly is taken into account. However, this work does not consider any kind of uncertainty.

FABER [16] is a pervasive system used to detect abnormal behavior of a patient. Firstly, it deduces events and actions from the acquired sensor data. This is done based on simple ontological inference methods. Then, these events and actions are sent to a Markov Logic Network (MLN) as a machine learning method to analyze the event logs and infer the start/end time of activities. The inferred activity boundaries are communicated together with actions and events to the knowledge based inference engine. This engine evaluates the rules modeling abnormal behaviors and detected abnormal behaviors are communicated to the hospital center for further analysis by the doctors. Nevertheless, similarly to previous works this system does not handle uncertainty of sensor data.

SmartFABER [3] system is an extension and an improvement of FABER [16]. These two frameworks share the same aims. Regarding SmartFABER [3], instead of communicating the inferred events and actions to MLN classifier, the system sends them to a module that is in charge of building vectors of features based on the received events. Then, these features are communicated to a machine learning module for the classification of activities. Next, a proposed algorithm called SmartAggregation is applied to infer current activity instances. For deducing an activity's instance from a sequence of events classified to an activity, the algorithm verifies whether each event satisfies a set of conditions. These conditions are defined by a human expert after a deep analysis of the semantic of activity. If all events satisfy all conditions, then an activity instance could be

inferred. This work is proved to outperform FABER [16]. However, it suffers from two main drawbacks: (1) There is no uncertainty handling for sensor data; (2) The performance of the SmartAggregation algorithm depends heavily on the defined conditions. It can suffer from time consuming if there is a huge number of conditions that need semantic verification.

3 The AGACY Monitoring Architecture Overview

The overall architecture is composed of two layers: the Knowledge based layer and the Data driven layer, as depicted in Fig. 1. The multimodal fusion process aims to homogenize the sensor data and compute their uncertainty's value (e.g. FSCEP system [1]). This process is out of the scope of this paper. We assume that the homogeneous sensor data together with their uncertainty values are provided to the Knowledge based layer. The knowledge based layer then represents semantically the incoming sensor data together with their uncertainty values (`Ontological modeling`). Afterwards, it infers actions and events from the modeled sensor data (`Semantic reasoning`) and computes their uncertainty's values. The obtained actions and the computed uncertainty values are sent to the data driven layer. The layer is responsible for: (1) classification of the actions into features (`Time & Uncertainty-based features extraction`), (2) classification of the features and actions into activities (`Dempster Shafer theory for activity classification`), and finally (3) inferring of activities' instances (`Activities instances inferring under uncertainty`). In the following, further explanation of each layer is provided.

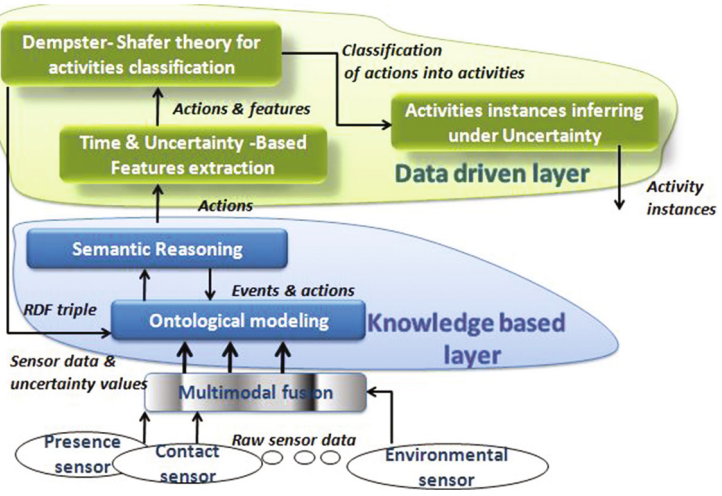

Fig. 1. The architecture of the AGACY Monitoring

4 Knowledge Based Layer

4.1 Ontological Modeling

The ontological modeling module allows sensor data to be formally conceptualized. This conceptualization, as the ontology is defined for, serves to provide a semantic model from the data. However, the native ontological representation is known to poorly handle uncertainty. Therefore, some attempts for uncertainty integration into ontological models have been realized [5,6].

In this work, we adopt the model proposed in [6] since it could be attached to any existing ontology without the need for redesigning it. However, the problem of this model is the lack of temporal representation. In order to distinguish two similar sensor data that have the same uncertainty value but come in two different timestamps, it is highly important to associate an uncertainty value (assigned to a sensor data) with the unique time. Hence, to overcome this problem, we extended the model in [6] by adding a temporal element (the class Time). Our proposed model is depicted in Fig. 2.

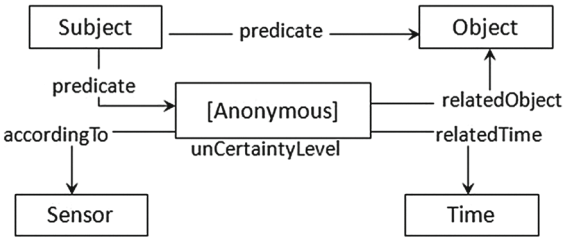

Fig. 2. The extended uncertainty representation model

Anonymous resource is related to the class: *Subject* through the ObjectProperty: *predicate* and to the *Object* through the ObjectProperty: *relatedObject*. It holds the uncertainty level of the triple < *Subject*; *predicate*; *Object* >, according to the model in Fig. 2, using its dataProperty: *unCertaintyLevel*. It is also linked to the source of uncertainty – namely a sensor or a set of sensors that derive uncertainty– with the objectProperty: *accordingTo*, and to class: *Time* via the objectProperty: *relatedTime*.

Through this model, the uncertainty can be easily integrated into any ontology. To do so, we have designed an ontology to represent sensors, sensor data, sensor data uncertainty values, actions, events, persons, time, and so on. Figure 3 shows an excerpt of this ontology, in which the model from Fig. 2 is used in order to represent the uncertainty. As Fig. 3 depicts, among the basic high level concepts in the ontology we find: *Sensor*, *Action*, *Event*, *Activity*, *Person*, *Object*, and *Time*.

Events and actions have also values of uncertainty that are represented as dataProperties. Next subsection will describe how the events and actions are inferred and how their uncertainty values are computed.

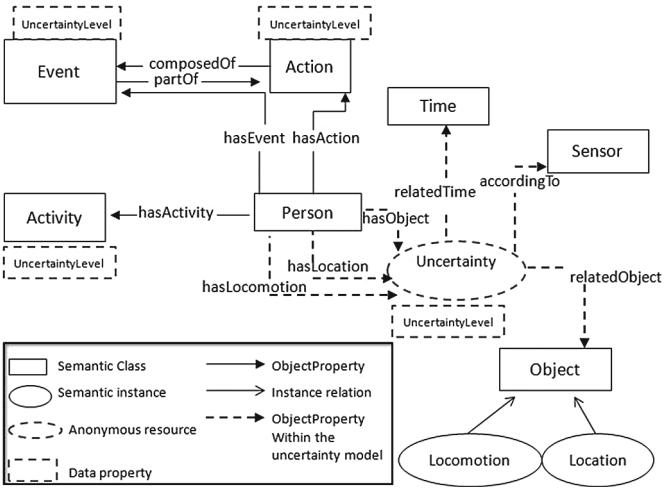

Fig. 3. An excerpt of the ontology with uncertainty integration

4.2 Semantic Reasoning

This module is mainly responsible for the inference of events and actions and their uncertainty values. Events are deduced from sensor data and actions are inferred from events. Formally we define an event as follows:

Definition 1. ***Event Instance:*** *Let E be the set of event labels; T a set of all possible time-stamps; and C a set of uncertainty values. An event instance is a triple $ev(e_i, t_i, c_i)$ where $e_i \in E$, $t_i \in T$ and $c_i \in C$.*

The events are deduced by combining the *logic with possibilities* [7] with the *modified semantic logic rules* in [6]. On the one hand, in [6] the traditional semantic logic rules are modified in order to integrate the uncertainty in a clause's definition. This is done according to the ontological model in Fig. 2 (without the concept Time). Then, this modified rule is able to propagate the uncertainty value from the premise to the conclusion of the rule. However, the propagation is done from only one clause within the premise. Thus, the case when the rule contains more than one clause is omitted. On the other hand, the *logic with possibilities* has the advantage to propagate values of possibilities (in our case values of uncertainty (denoted Unc in Eq. 1) in a rule from a premise with multiple clauses to the conclusion. This is done through the following rules:

$$\begin{aligned} Unc(p \wedge q) = min(Unc(p), Unc(q));\ Unc(p \vee q) = max(Unc(p), Unc(q)) \\ Unc(\neg p) = 1 - Unc(p);\ Unc(p \longrightarrow q) = max(1 - Unc(p), Unc(q)) \end{aligned} \quad (1)$$

Accordingly, in order to deduce the events, we combine the *logic with possibilities* with the *modified semantic logic rules* where the premise is a set of clauses linked with logic operators. A clause is a RDF triple translated in logic [6].

This RDF triple models the sensor data with their uncertainty according to the ontology depicted in Fig. 3. The conclusion is an event. To propagate the clauses' uncertainty towards the conclusion, the rules in Eq. 1 are applied.

In this example, we demonstrate how to infer an instance of event labeled *sitOnChair*:

$\forall$ se_1, $se_2 \in \{Sensors\}$, t_1, $t_2 \in \{Time\}$, and p $\in \{Person\}$
(p hasLocomotion [**a Uncertainty; uncertaintyLevel** n_1**; relatedObject SitOn; relatedTime** t_1**; accordingTo** se_1]
$\wedge$
(p hasObject [**a Uncertainty; uncertaintyLevel** n_2**; relatedObject Chair; relatedTime** t_2**; accordingTo** se_2])
$\longrightarrow$ ev(SitOnChair, max(t_1,t_2), min(n_1, n_2))

Example of rule for inferring the event instance sitOnChair

As we can see, the premise of this rule contains 2 clauses. Each one is a RDF triple representing an uncertain sensor data according to the ontology depicted in Fig. 3. The first clause means "*the resident p is observed to have the locomotion SitOn at time* t_1 *with uncertaintyLevel* n_1 *according to the sensor* se_1" while the second clause translates the information "*the resident p is observed to get the object Chair at time* t_2 *with uncertaintyLevel* n_2 *according to the sensor* se_2". The rule's output always creates a new event with an uncertainty value deduced according to rules described in Eq. 1. Then, a production of the event with label *sitOnChair* is deduced. Since the premise is a conjunction of clauses, its uncertainty value is the minimum between both values of clauses' uncertainty. Accordingly, this new event will be modeled in the ontology as an instance of the concept *Event* and the uncertainty value as a dataProperty of the instance.

After deducing events, the actions are inferred. An action is defined as follows:

Definition 2. ***Action Instance:*** *Let A be the set of actions labels, T the set of all possible timestamps, C the set of uncertainty values. An action instance is a triple* $ac(a_i, t_i, c_i)$ *where* $a_i \in A$, $t_i \in T$ *and* $c_i \in C$.

The actions and their uncertainty values are inferred by simply applying the logic with possibilities rules. The action is the conclusion of a rule, the clauses of its premise are the events or some defined restrictions.

This example is to illustrate how we can deduce the production of an action's instance with label *SitOnChairAtKitchenTable*:

ev(*SitOnChair*, t_1, n_1) $\wedge$ ev(*PresenceAtKitchenTable*, t_2, n_2) $\wedge$ ($t_1 \geq t_2$) $\wedge$ ((t_1-t_2) $\leq$ 5s) $\longrightarrow$ (*SitOnChairAtKitchenTable*, t_2, min(n_1,n_2))

The previous rules are examples of rules that are managed by the system. However, a set of required rules are defined by experts according to the semantic of the activities to be monitored and the types of sensors in the smart environment. Other rules can be inferred through an ontological inference engine according to the axioms defined in the ontology.

The deduced actions are then communicated to the module `Time & Uncertainty-based features extraction` described in the following section.

5 Data Driven Layer

5.1 Time and Uncertainty-Based Features Extraction

For each received action ac(a_i, t_i, c_i), this module is in charge of building a feature vector representing the sequence S of the recent actions in a time window with a size equal to n. S = ⟨ac(a_{i-n+1}, t_{i-n+1}, c_{i-n+1}), ..., ac(a_{i-1}, t_{i-1}, c_{i-1}), ac(a_i, t_i, c_i)⟩.

In this work, we improve the technique proposed in [8] for features extraction. We have chosen this method since it is proved to be effective in recognizing activities based on streams of sensor events or actions instead of streams of sensor data compared to traditional features extraction techniques. This technique builds a vector of features for each events sequence S. The produced vector of feature of a sequence of events S_i holds these information: (i) The label of the feature, K_i; (ii) The time t_0 of the first event in S_i; (iii) The time t_i of the last event in the sequence S_i; (iv) The list of events under S_i; (v) A weight value fine-tunes the contribution of each event in S_i, so that recent events participate more than the older ones. This weight value is computed as follows (Eq. 2):

$$F_{k_i}(S_i) = \sum_{ev(e_j,t_j)\in S_i} exp(-\chi(t_i - t_j)) \times f_{k_i}(ev(e_j, t_j)) \tag{2}$$

where the factor χ determines the time-based decay rate of the events in S_i; t_i-t_j is expressed in seconds and f_{k_i}(ev(e_j, t_j)) is the time-independent participation of ev(e_j, t_j) in the computation of the F_{k_i} value. The other way around if $ev(e_j, t_j)$ participates in the execution of k_i then $f_{k_i}(ev(e_j, t_j)) = 1$ else 0.

As we can see in Eq. 2, the greater difference is of the time distance between ev(e_j, t_j) and ev(e_i, t_i)), the less ev(e_j, t_j) participates in the computation of F_{k_i}.

In our understanding, this may be true when the approximate duration of the feature is short (e.g. in terms of seconds or some minutes). In contrast, when it is long (e.g. in terms of hours), this hypothesis is not always valid. Let us have the example of the feature "stove usage" in [3]. The duration of the execution of this feature, for a particular recipe, may be equal to a number of hours. We suppose that the vector of this feature contains the following two events *openStove* and *closeStove*. Since the duration of the feature "stove usage" is in terms of hours, the value of ($t_{closeStove} - t_{openStove}$) may be equal to 1 h or 2 h. Accordingly, by applying Eq. 2 the event *openStove* does not participate in the weight calculus of the feature "stove usage". Therefore, incorrect value of weight may be obtained. Accordingly, the execution of the event *openStove* must have a high impact in the execution of the feature. Intuitively, the stove can not be used if it is not opened. Moreover, this technique assumes that the only factor that may have impact on the computation of the feature's weight is the time distance between events. However, when information about actions and events uncertainty values is provided, this information should be taken into consideration in the computation of the feature's weight. Therefore, we made the following assumptions:

A1. The uncertainty values must have an impact on the calculus of the feature weight: the higher uncertainty value of event or action is, the more the weight of the feature increases.

A2. A product containing the uncertainty value of an event or action, must be higher or equal than this uncertainty value: The uncertainty value must not be decreased when it is multiplied.

To overcome the problem mentioned above, we propose a new version of Eq. 2 that meets the assumptions A1 and A2. We formally define the notion of Short Time Feature (STF) and Long Term Feature (LTF) as follows:

Definition 3. ***Short Time Features (STF)*** *is a set of features that holds only features having duration less than ten minutes. Let Dur(f) be the duration of feature f: $f \in STF \Leftrightarrow Dur(f) <= 600\,\mathrm{s}$.*

Definition 4. ***Long Term Features (LTF)*** *is a set of features that holds only features having duration more than ten minutes. Let Dur(f) be the duration of feature f: $f \in LTF \Leftrightarrow Dur(f) > 600\,\mathrm{s}$.*

Firstly, it is important to note that This value of ten minutes is chosen intuitively and it is variable according to the experiments. Then, to compute the weight of the feature, we propose the following Eq. 3.

$$F_{k_i}(S_i) = \sum_{ac(a_j,t_j,c_j)\in S_i} c_j \times Fact_{\chi,\delta t_{ij}} \times f_{k_i}(ac(a_j,t_j,c_j)) \tag{3}$$

$$Fact_{\chi,\delta t_{ij}} = \begin{cases} exp(-\chi * \delta^h t_{ij}) & If\ (k_i \in LTF) \\ \frac{1}{\chi * \delta^s t_{ij}} & If((\chi * \delta^s t_{ij} \succ 1)\ \&\ (k_i \in STF)) \\ 1 & Otherwise \end{cases}$$

It is worth mentioning that in our work the features vectors are built from actions instead of events, in contrast to [8] and the size n of the time window is chosen according to the nature of the feature (STF or LTF). We note that $c_j \times Fact_{\chi,\delta t_{ij}}$, in Eq. 3, is a sort of uncertainty where $c_j \times Fact_{\chi,\delta t_{ij}} \leq c_j$ (Assumption A2). $\delta t_{ij} = t_i - t_j$. $\delta^h t_{ij}$ means that δt_{ij} is expressed in hours and $\delta^s t_{ij}$ means that δt_{ij} is expressed in seconds. To fix the problem of the time delay in Eq. 2, we distinguish three cases: (1) The feature is a LTF ($k_i \in LTF$). Then, to compute $Fact_{\chi,\delta t_{ij}}$ the same function ($exp(-\chi * \delta t_{ij})$) in Eq. 2 is used. However, δt_{ij} is expressed in hours ($\delta^h t_{ij}$) instead of seconds. Accordingly, the actions that happened earlier could participate in the weight computation in contrast to Eq. 2; (2) The feature is a STF ($k_i \in STF$) and $\chi * \delta t_{ij} \succ 1$. Then, $Fact_{\chi,\delta t_{ij}} = \frac{1}{\chi * \delta^s t_{ij}}$. The quotient function is chosen to compute $Fact_{\chi,\delta t_{ij}}$ since it has a less decreasing shape than the exponentiation function. (3) If $(\chi * \delta^s t_{ij}) \prec 1$, then $\frac{1}{\chi * \delta^s t_{ij}} \succ 1$ and accordingly $c_j \times Fact_{\chi,\delta t_{ij}} \succ c_i$ that does not correspond to Assumption A2. Therefore, $Fact_{\chi,\delta^s t_{ij}} = 1$. This is due to the fact that since t_i and t_j are very close, $ac(a_j, t_j, c_j)$ must have the highest impact on the weight computation, i.e. the value 1 in [0..1] is chosen. As a result, computed weights of the feature are used as their uncertainty values. Afterwards, the actions and the features together with their uncertainty values are sent to `Dempster-Shafer theory for activity classification` module.

Algorithm 1. AGACY

Input: G, minUncert, tolUncert, β_{delay}, $maxDelay_A$, A
Output: Inst: a set of activity instances

1: $\aleph \longleftarrow segmentation(G, maxDelay_A)$;
2: $Inst \longleftarrow \phi$;
3: $LowUncert \longleftarrow \aleph$;
4: **for** each g in $\aleph$ **do**
5: **if** $(\forall ac(a_i, t_i, c_i) \in g \longrightarrow c_i \succ minUncer)$ **then**
6: $inst \longleftarrow$activity instance of A that is generated for the observation g;
7: $Inst \longleftarrow Inst \cup inst$;
8: $LowUncert \longleftarrow lowUncert \backslash g$;
9: $G \longleftarrow G \backslash all\ actions\ in\ g$;
10: **else**
11: $missClass \longleftarrow \phi$;
12: $ratioUncert \longleftarrow ComputeRatioCert(g)$;
13: **if** $ratioUncert \succ tolUncert$ **then**
14: $missClass \longleftarrow missClass \cup GetLowUncert(g)$;
15: $K \longleftarrow 0$;
16: $toBeReplaced \longleftarrow \phi$;
17: **while** $(k \prec (card(G))\&(card(toBeReplaced) \prec card(missClass))$ **do**
18: **if** $(ac(a_k, t_k, ck) \in G)\&(ac(a_k, t_k, ck) \notin g)\&(c_k \succ minUncert)\&(\forall ac(a_j, t_j, c_j) \in g \longrightarrow |t_j - t_k| \prec maxDelay_A + \beta_{delay})$ **then**
19: $toBeReplaced \longleftarrow toBeReplaced \cup ac(a_k, t_k, c_k)$;
20: **end if**
21: $K \longleftarrow K + 1$;
22: **end while**
23: **end if**
24: **end if**
25: **if** card(toBeReplaced) = card(missClass) **then**
26: $g \longleftarrow g \backslash missClass$;
27: $g \longleftarrow g \cup toBeReplaced$;
28: $inst \longleftarrow$ activity instance that is generated from the observation g;
29: $Inst \longleftarrow Inst \cup inst$;
30: $lowUncert \longleftarrow lowUncert \backslash g$;
31: Partition(missClass, $\aleph$, maxDelay_A);
32: $G \longleftarrow G \backslash \{all\ actions\ in\ g\}$;
33: **end if**
34: **end for**

5.2 Dempster-Shafer Theory for Activity Classification

In order to classify activities, we use the well known Dempster Shafer theory (DS) [2]. It has proven to provide decent result in comparison to other techniques such as J48 Decision Tree [9]. The model is used through a directed acyclic graph, where actions in AGACY Monitoring are equivalent to the named evidences in DS, the called hypothesis in DS are features in AGACY Monitoring, and outputs are activities with their uncertainty values. We note here that activities are composed of set of features and as sated in the previous section features are composed of set of actions. The uncertainty values of activities are computed based on the uncertainty values of the features and the mass function value defined in [9].

5.3 Activities Instances Inferring Under Uncertainty

In this section, we present *AGACY*, our algorithm for activity instances inference. The algorithm aims to improve an existing one called *SmartAggregation* [3].

We improve this method since it is accurate and it improves the well known algorithm Naive aggregation [3]. The basic idea of the *SmartAggregation* algorithm is the following: if two consecutive events that occurred respectively at t_i and t_{i+1} are classified with the same activity's class $A_i = A_{i+1}$ and verify the defined conditions of the activity A_i, $C^{(A)}$, they are considered as an observation[1] generated by the same activity instance of the class A_i. Otherwise, if an event does not satisfy all the conditions, the algorithm tries to integrate it into an observation of a previous inferred instance providing that it satisfies all the conditions defined by human experts. Hence, this algorithm, despite its accuracy, suffers from scalability and time consuming. Assuming a big number of events, the algorithm must check each event for satisfying all the conditions, which is demanding and time consuming. To tackle these problems, we refine our recognition method by the proposed Algorithm 1. It takes advantage of the obtained uncertainty values of the activities to infer the current instances of activity. The basic idea of Algorithm 1 is the following: a group of actions could be considered as an observation of an activity instance providing that (1) all the actions have uncertainty values beyond a defined minimum value, e.g. minUncert, (Lines 2–9) and (2) the time distance between every pair of consecutive actions within the group must be lower than $maxDelay_A$. The second condition is always true since the first step done by the algorithm is a segmentation over the output of the DS (Line 1): All the actions associated with the same activity class A and temporally close (according to $maxDelay_A$) are grouped together to obtain a set of groups ℵ. G is a set of all actions without segmentation. In this case, there are no conditions to be verified for the actions compared to *SmartAggregation* algorithm. Indeed, the uncertainty value of each action is replaced, before the execution of the algorithm, with the uncertainty value of its classified activity class. Therefore, when the uncertainty value c of an action ac is high, it is almost sure that the action ac is really produced and belongs to the correct class of activity act, hence, must belong to an instance of act.

This process is less time consuming than *SmartAggregation* algorithm. On the other hand, when at least one action in the group has an uncertainty value lower than minUncert, the algorithm accepts some time distance shift between actions regarding the uncertainty degree (Lines 10–32). To do so, it checks firstly if the ratio of the number of actions (ratioUncert) in the group that have lower uncertainty value than minUncert is higher than a defined value called tolUncert (Line 13). If so, the algorithm attempts to replace the actions that have lower uncertainty values with other actions from other groups. A new action can be added if it has higher uncertainty value than minUncert and the time distance between the action and each action in the group is lower than the $MaxDelay_A + \beta_{delay}$ (Lines 18 and 19). β_{delay} represents a sort of tolerance about the time distance between two actions. Thus, we assume that uncertainty value is more important than the time distance. Finally, when all uncertain actions are replaced with actions having high uncertainty values with some time distance tolerance, this new group can generate a new activity instance (Line 27). Afterwards, the removed actions having lower uncertainty values than

[1] An observation is a sequence of events that is generated by an activity instance.

minUncert in the old group are distributed to the rest of the groups according to $maxDelay_A$ (Line 31).

6 Evaluation and Discussion

For this experiments, we evaluate the proposed AGACY Monitoring system. In this section, we first describe the dataset used for the experiments. Then, the experiments and the achieved results are presented.

6.1 DataSet

We used real-life data collected in highly rich smart environments. The dataset[2] [17] was obtained as a part of the Opportunity project[3]. The dataset focus on activities concerning breakfast that holds (from AGACY Monitoring perspective) homogeneous sensor data (level 4), events (level 3), actions (level 2), and activities (level 1) that have been done by three persons S10, S11, and S12 with three different routines each (ADL1-3). In order to test AGACY Monitoring system, this dataset does not contain information about sensor data uncertainty values (level 4). Therefore, we have randomly annotated the level 4 in the dataset with high uncertainty values in [0.8..1]. Moreover, we have injected a set erroneous sensor data in the dataset annotated with low uncertainty values in [0..0.4].

6.2 Implementation and Experimental Setup

We have implemented the AGACY Monitoring system using JAVA. Regarding data driven layer, for this dataset, we have considered the set of features depicted in Table 1 to be treated in the *Time & Uncertainty based features extraction* module.

Table 1. List of considered features. STF: Short Time Feature, LTF: Long Term Feature

Feature name	STF/LTF	Duration (s)
PrepareCoffee	STF	600
Drink	STF	120
GatherCutlery	STF	600
GatherFood	STF	600
Eat	STF	1200
PutAwayFood	STF	600
PutAwayCutl	STF	600
Dishwasher	STF	300
Resting	STF	600

For the *Activity classification* module, the experiments are not limited only to the use of DS – we also have the SVM for activity classification. The results obtained from this module are compared with those obtained by Rim et al. [14]. Finally, regarding the *Activities' instances inferring under uncertainty* the algorithm AGACY has 4 variables: minCert, tolCert, maxDelay_A, β_{DELAY}. Due to the lack of space, we only present the most adequate values regarding this experiments: $\text{minCert} = 0.5$; $\text{tolCert} = 1$; $\text{maxDelay}_A = 30\,\text{s}$; $\beta_{\text{DELAY}} = 10\,\text{s}$;

[2] The dataset: http://webmind.dico.unimi.it/care/annotations.zip.
[3] http://www.opportunity-project.eu.

6.3 Evaluation and Results

Activity Recognition Evaluation. As described in the previous sections, the *Activity classification* module outputs the predicted activity class. It is worth mentioning that the system has no False Negative result (FN = 0): it outputs always at least one activity. Figure 4 depicts the average precision measure over the three routines for all subjects by varying the value n of the time window (Tw) (see Subsect. 4.1) in [60 s..300 s] with one erroneous sensor data for five correct ones (e.g. 1/5 erroneous sensor data). The code source of AGACY Monitoring is available online[4]. As it is clearly shown in Fig. 4, the DS with n = 180 s reaches 91% of precision recognition rate. For time windows shorter or longer than 180, DS tends to become less efficient: DS is efficient where time window are properly proportional to the activities: when the time window is too big, there may be conflict between activities, while when it is too small, DS does not have enough data to be efficient. On the other hand, SVM gives better results for short time window (n <= 120 s), but with the increase of n value, the accuracy of classification gets worse. This maybe explained by the fact that SVM is not as dependent on features weights as DS. In general, DS provides better results than SVM. Figure 5 shows the average precision measure over the three routines for all subjects by varying the proportion of the introduced uncertain sensor data compared to the correct sensor data with n = 180 s. As it is clearly shown in the figure the DS is more efficient than SVM: The precision values of DS are in range [0,74..0,91], however that of SVM are in range [0,65..0,77]. Both methods have a decreasing precision values when the number of uncertain sensor data in the dataset increases. This is an expected result since the methods will have less certain data to make the right decision. However, despite the dataset half contains uncertain sensor data, the system is able to predict the activity with a good precision level (74%).

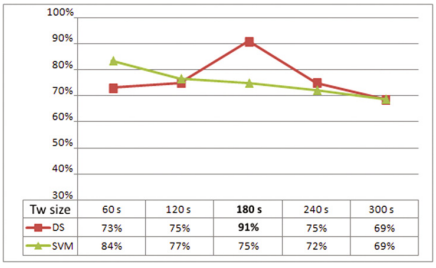

Fig. 4. Average recognition precision for all subjects over the three routines with different values of the size n of the time window (Tw)

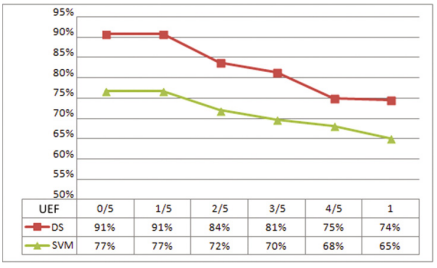

Fig. 5. Average recognition precision for all subjects over the three routines for varying frequency of uncertain event (UEF). The value 1/5 means there is one uncertain sensor data for five correct ones. 1 means there is one uncertain sensor data for one correct

[4] https://github.com/Nath-R/AGACY-monitoring.

Table 2. Average recognition results over three routines for the three subjects obtained by AGACY Monitoring (with DS) and the system proposed in [14] for n = 180 s.

All users	AGACY monitoring	[14]
Precision	0,91	0,91
recall	1	0,65

We have compared the obtained results with another system proposed in [14]. The system has been applied with the same dataset (without erroneous sensor data and uncertainty values).

As it is shown in Table 2 the two systems have the same average precision recognition. However, AGACY Monitoring outperforms the second system within recall value. This can be explained by the fact that the system in [14] returns null if it can not infer an activity. However, AGACY Monitoring has the advantage to always predict an activity. Moreover, the proposed AGACY Monitoring system is effective despite the introduced erroneous sensor data in the dataset. This confirms the ability of AGACY Monitoring to effectively handle uncertainty and predict activities.

Evaluation of AGACY Algorithm for Activities Instances Inference. In this paragraph, we show the evaluation of the AGACY algorithm for inferring the activities instances. The accuracy evaluation is based on the following metrics: precision, recall, and F1 score. Furthermore, we have evaluated the system for time consuming. We tested the AGACY algorithm, on the output of the activities classification module with DS (n = 180 s and the value 1/5 of erroneous sensor data in the dataset).

Moreover, we have implemented and tested the Smart Aggregation algorithm [3] with the same dataset without uncertainty annotations. The average time consuming results over the three routines of both AGACY and SmartAggregation are presented in Fig. 6. The results regarding average accuracy of activities instances detection are depicted in Table 3.

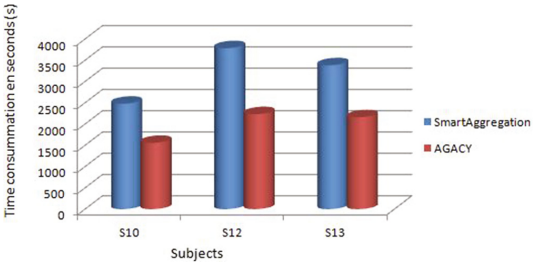

Fig. 6. Average time consuming of activity instances recognition for all subjects over the three routines

Table 3. Average recognition results for activities instances detection over three routines for subjects S10, S11, and S12.

All users	AGACY	SmartAgg.
Precision	0.916	0.854
Recall	0.830	0.872
F1	0.871	0.862

Using above-mentioned experimentation, we obtained two major results: firstly, as it is highlighted in Fig. 6, AGACY has significantly reduced the time execution for activities' instances recognition regarding *SmartAggregation*. Approximately, for the used dataset, AGACY reduces 40% of *SmartAggregation*'s time consummation. Secondly, in terms of performance both algorithms are efficient with similar F1

score value. As a result, the experiments have proved that AGACY is fast and efficient.

7 Conclusion and Future Work

In this paper, we proposed the AGACY Monitoring hybrid model for activity recognition and sensor data uncertainty handling. The main novelty of AGACY Monitoring is that it combines knowledge based with data driven methods. Thus, several modules contribute to compute the uncertainty value of the expected output. Unlike the related work, AGACY Monitoring supports the inherent uncertain nature of sensor data and exploits it to compute the uncertainty value of each module's output. Besides, the system is able to integrate background knowledge with the data driven method for the recognition of current activity instances. Moreover, the experimental results confirm the viability and the performance of the proposed system and its high level precision even during the presence of uncertain sensor data.

Our future work involves the reuse of an existed upper ontology such DOLCE ontology.

Acknowledgements. This work has been partially supported by the project COCAPS (https://agora.bourges.univ-orleans.fr/COCAPS/) funded by Single Interministrial Fund N20 (FUI N20).

References

1. Jarraya, A., Ramoly, N., Bouzeghoub, A., Arour, K., Borgi, A., Finance, B.: A new model for context perception in smart homes. In: OTM Conferences: COOPIS (2016)
2. Lotfi, A.Z.: A simple view of the Dempster-Shafer theory of evidence, its implication for the rule of combination. AI Mag. **7**, 85–90 (1986). ACM
3. Riboni, D., Bettini, C., Civitares, G., Janjua, Z.H.: SmartFABER: recognizing fine-grained abnormal behaviors for early detection of mild cognitive impairment. Artif. Intell. Med. **67**, 57–74 (2016). Elsevier
4. Riboni, D., Bettini, C.: COSAR: hybrid reasoning for context-aware activity recognition. Pers. Ubiquit. Comput. **3**, 271–289 (2011). Springer
5. Singh, A., Juneja, D., Sharma, A.: A fuzzy integrated ontology model to manage uncertainty in semantic web: the FIOM. Int. J. Comput. Sci. Eng. **3**, 1057–1062 (2011). Citeseer
6. Aloulou, H., Mokhtari, M., Tiberghien, T., Endelin, R., Biswas, J.: Uncertainty handling in semantic reasoning for accurate context understanding. Knowl.-Based Syst. **77**, 16–28 (2015). Elsevier
7. Dubois, D., Lang, J., Prade, H.: Automated reasoning using possibilistic logic: semantics, belief revision, and variable certainty weights. IEEE Trans. Knowl. Data Eng. **6**, 64–71 (1994). IEEE
8. Krishnan, N.C., Cook, D.J.: Activity recognition on streaming sensor data. Pervasive Mob. Comput. **10**, 138–154 (2014). Elsevier

9. Sebbak, D., Benhammadi, F., Chibani, A., Amirat, Y., Mokhtari, A.: Dempster-Shafer theory-based human activity recognition in smart home environments. Ann. Telecommun. **69**, 171–184 (2014). Springer
10. Ye, J., Dobson, S., McKeever, M.: Situation identification techniques in pervasive computing: a review. Pervasive Mob. Comput. **9**, 36–66 (2012). Elsevier
11. De Backere, F., Ongenae, F., Van den Abeele, F., Nelis, J., Bonte, P., Clement, E., Philpott, M., Hoebeke, J., Verstichel, S., Ackaert, A., De Turck, F.: Towards a social and context-aware multi-sensor fall detection and risk assessment platform. Comput. Biol. Med. **64**, 307–320 (2015). Elsevier
12. Patterson, D.J., Fox, D., Kautz, H., Philipose, M.: Fine-grained activity recognition by aggregating abstract object usage. In: ISWC (2005)
13. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. IEEE Intell. Syst. Appl. **13**, 18–28 (1998)
14. Helaoui, R., Riboni, D., Stuckenschmidt, H.: A probabilistic ontological framework for the recognition of multilevel human activities. In: UbiComp (2013)
15. Riboni, D., Bettini, C.: Context-aware activity recognition through a combination of ontological and statistical reasoning. In: Zhang, D., Portmann, M., Tan, A.-H., Indulska, J. (eds.) UIC 2009. LNCS, vol. 5585, pp. 39–53. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02830-4_5
16. Riboni, D., Bettini, C., Civitarese, G., Janjua, Z.H., Helaoui, R.: Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment. In: PerCom (2015)
17. Lukowicz, P., Pirkl, G., Bannach, D., Chavarriaga, R.: Recording a complex, multi modal activity data set for context recognition. In: ARCS, 1st Workshop on Context-Systems Design, Evaluation and Optimisation (2010)