Elsevier Editorial System(tm) for Pervasive and Mobile Computing
Manuscript Draft

Manuscript Number:

Title: Dynamic Sensor Data Segmentation for Real-time Activity Recognition

Article Type: Research Paper

Corresponding Author: Mr. George Onyango Okeyo,

Corresponding Author's Institution: University of Ulster at Jordanstown

First Author: George Onyango Okeyo

Order of Authors: George Onyango Okeyo; Liming Chen; Hui Wang; Roy Sterritt

Dynamic Sensor Data Segmentation for Real-time Activity Recognition

George Okeyo Liming Chen Hui Wang Roy Sterritt
Computer Science Research Institute, School of Computing and Mathematics, University of Ulster, Shore Road, BT37 0QB, Newtownabbey, UK.
okeyo-g@email.ulster.ac.uk, {l.chen,h.wang,r.sterritt}@ulster.ac.uk

Abstract-Approaches and algorithms for activity recognition have recently made substantial progress due to advancements in pervasive and mobile computing, smart environments and ambient assisted living. Nevertheless, it is still difficult to achieve real-time continuous activity recognition as sensor data segmentation remains a challenge. This paper presents a novel approach to real-time sensor data segmentation for continuous activity recognition. Central to the approach is a dynamic segmentation model, based on the notion of varied time windows, which can shrink and expand the segmentation window size by using temporal information of sensor data and activities as well as the state of activity recognition. The paper first analyses the characteristics of activities of daily living from which the segmentation model that is applicable to a wide range of activity recognition scenarios is motivated and developed. It then describes the working mechanism and relevant algorithms of the model in the context of ontology-based activity recognition. The presented approach has been implemented in a prototype system and evaluated in a number of experiments. Results have shown average recognition accuracy above 83% in all experiments for real time activity recognition, which proves the approach and the underlying model.

Index Terms—ontology, sensor data segmentation, time window, real-time activity recognition, ontological activity modelling, temporal information

1 INTRODUCTION

Ambient Assisted Living (AAL) is motivated by the need to support independent living, whereby technology is used to provide people with proactive services in their normal environments, e.g. at home. Smart Homes (SH) have emerged as a viable technology that can support individuals, such as the elderly and disabled, for independent and dignified living. To provide assistance for individual inhabitants of an SH, activity recognition is required to identify the task that the individual is currently undertaking. In addition, activity recognition also determines whether the individual has any difficulties completing tasks.

To perform activity recognition, three important tasks are undertaken, namely activity modelling, activity monitoring, and pattern recognition. During activity modelling, suitable computational models of activities are created and presented in a format that can be automatically processed by computer systems. Existing literature provides a number of modelling approaches that fall in two main categories: data-driven and knowledge-driven activity modelling. In data-driven activity modelling [1-3], activity models are learnt from pre-existing activity datasets. In knowledge-driven activity modelling [4-6], knowledge engineers and/or domain experts employ knowledge engineering techniques to specify activity models explicitly. The resulting knowledge bases capture and encode commonsense domain knowledge. The activity monitoring task captures an inhabitant's contextual information, e.g. location, time, objects used, and previous tasks performed, and which is then used to infer ongoing activities. Various monitoring techniques, such as dense sensing [7, 8] , computer vision [9-11], and wearable sensors [12, 13], have been adopted for collecting contextual information. Finally, during pattern recognition, incoming sensor data is processed against the activity models to infer the ongoing activities. Analogous to activity modelling approaches, pattern recognition can be performed through either the data-driven or the knowledge-driven approach. Data-driven approaches [13-15] use machine learning techniques, typically statistical and probability analysis methods, to process sensor data against the activity models for pattern recognition. Conversely, knowledge-driven approaches [4, 5, 16-18] make use of knowledge-based inference techniques to infer ongoing activities. Usually, they take as input the available sensor data and process them against the predefined explicit activity models.

While vision-based activity monitoring has been widely used in security surveillance, dense sensor based activity monitoring has gained currency in SH environments due to privacy and ethical considerations. In such environments, sensors are attached to objects in the environment (e.g. fridges, cupboards, e.t.c.) and an inhabitant's interactions with these objects are monitored and used to identify the ongoing activities of daily living (ADLs). A key problem in dense sensor based activity recognition when sensors are activated along a timeline is how the sensor data are segmented so that the set of sensor interactions represents exactly a unique activity.

Recently, ontology-based knowledge driven approach to activity recognition has attracted increasing attention. Ontology is essentially a formal, explicit specification of a shared conceptualization of a domain [19]. It provides a vocabulary for modelling a domain by specifying the latter's objects and/or concepts, properties, and relationships. In this way, domain and prior knowledge can be exploited to predefine activity models, i.e., the so-called activity ontologies. Whenever sensor data are obtained, the approach determines the likely ADL by reasoning against the model through ontological inference. Nevertheless, existing works on ontology-based activity recognition [8, 9, 20] and similar work on knowledge-driven activity recognition [16, 17, 21] do not clearly articulate the mechanism about how and what sensor data are selected from a live data stream for performing activity inference. In some research experiments that support on-line continuous activity recognition, the experiments restart manually each time an ADL is identified. For the approach to be applicable to real-world use scenarios it is necessary that after an ADL is identified, the activity recognition process should continue on fresh sensor data and decide what to exclude from those already used in the previously identified ADL(s). Obviously, this is not a trivial task and requires the development of a suitable discriminating strategy. To this end, we develop a segmentation approach that makes use of temporal information associated with sensor data and temporal characteristics of an activity for real-time activity recognition. The approach addresses two important issues: 'segmentation and aggregation' and 'the conditions that trigger ontological reasoning'. *Segmentation* breaks down a sensor data stream into fragments that can be mapped to activity descriptions; while *aggregation* combines a finite collection of sensor data items available in a segment for activity inference.

The main purpose of this work is to develop an approach that can dynamically decide an appropriate set of sensor data from a live sensor data stream for real-time activity recognition. In addition, the approach is able to support continuous segmentation and aggregation along a timeline, thus allowing real-time ongoing activity recognition. To achieve this goal, in this paper we (1) describe a time window based segmentation model and related algorithms for real-time ontological activity recognition; (2) develop various mechanisms for dynamically manipulating time window parameters; (3) implement the proposed model and reasoning algorithms; (4) develop tools to obtain temporally-rich ADL data for testing and evaluation; and (5) evaluate the performance of the proposed model and algorithms in supporting real-time activity recognition. The research is based on typical ADL activities that an inhabitant can perform in the kitchen, lounge, and bathroom of a Smart Home, e.g. cooking, watching television, and showering.

By developing a systematic approach to dynamic sensor data segmentation for real-time continuous activity recognition, we have made a number of contributions. Firstly, we propose a time window based segmentation model that is applicable to a wide range of activity recognition scenarios. Secondly, we develop various mechanisms for dynamic manipulation of model parameters during activity recognition, such as the setting, shrinking, and expansion of the time window's length, thus adapting the segmentation model in terms of the way activities are performed. Thirdly, we integrate the dynamic sensor data segmentation approach into an ontology-based algorithm for real-time, continuous activity recognition. This provides a basis for the implementation of re-usable knowledge-driven algorithms and applications for real-time activity recognition. In addition, we develop a synthetic ADL data generator that can be used to quickly generate temporally-rich synthetic ADL data for evaluation of activity recognition algorithms. We believe the time window based segmentation model and associated algorithms in activity recognition provide a realistically scalable, reusable approach to continuously recognising activities of different complexities in a Smart Home context.

The remainder of the paper is organised as follows. Section 2 outlines related work. Section 3 describes the proposed approach, including ontological activity modelling. Sensor data stream segmentation and analysis is described in Section 4 which includes formal time window based modelling, recognition algorithms, and mechanisms used to dynamically vary the time windows. The implementation of various components and evaluation of the approach is presented in Section 5. Finally, Section 6 summarizes the results and discusses future work.

2 RELATED WORK

In this section we first briefly review related work in activity recognition. Secondly, since this work is motivated by the need to segment a sensor data stream using temporal information, we also review papers that use temporal segmentation for activity recognition.

2.1 Activity recognition

In [22] the authors explain activity recognition as the process that allows an actor's behaviour and their environment to be monitored and analyzed in order to infer the ongoing tasks. Activity recognition can be classified based on two criteria: 1) how are the activities monitored? ; 2) how are activities modelled, represented and subsequently processed to infer the ongoing activities?

Activity monitoring allows the context information associated with an activity to be captured for use in

activity inference. Based on activity monitoring, there are two main categories of activity recognition: vision-based and sensor-based activity recognition. Vision-based activity recognition relies on visual sensing equipments that monitor an actor's behaviour and associated environment changes [9-11, 23, 24]. Techniques from computer vision are then used to analyze the obtained visual information to obtain features for use in pattern recognition. The main criticism levelled against the adoption of this approach is that it is intrusive and may interfere with the privacy of actors. In contrast, sensor-based activity recognition uses a variety of sensor technologies, e.g. wireless sensor networks, wearable sensors, radio frequency identification (RFID) and geographical positioning systems (GPS), to monitor and track the actor's behaviour and environment [25]. In the literature, several sensor technologies have been used for monitoring, e.g. wearable sensors [12, 13, 26, 27], and object-based monitoring [7, 28]. In wearable sensor-based monitoring, sensors are attached to individuals and used to monitor activities related to human physical movements, e.g. climbing stairs, walking and typing. Conversely, in object-based monitoring common objects are embedded with sensors and the actor's interactions with these objects tracked. Activity inference is performed on these interactions. Sensor-based activity recognition is capable of addressing some of the privacy concerns associated with vision-based recognition. However, it is important to note that no approach can be said to be superior to the other. Instead, the nature of the application will dictate the choice of method to use and whether or not to combine both vision-based and sensor-based techniques.

Based on the second criteria (i.e., activity modelling, representation and inference), there are two main approaches to activity recognition: the so-called data-driven and knowledge driven activity recognition. Data-driven activity recognition uses state-of-the-art machine learning techniques that elicit activity models from existing datasets. Typically, probabilistic and statistical reasoning is used to perform activity inference. A number of techniques and tools have been investigated, e.g. hidden Markov models (HMM) [7], Dynamic Bayes Nets (DBNs)[7] , naive Bayes [29], nearest neighbour [27], support vector machines (SVM) [12], conditional random fields (CRF) [30] , and multiple eigenspaces [14]. Data-driven activity recognition techniques, e.g. HMMs and DBNs, are considered better in handling noisy, uncertain and incomplete data. For instance, by using probabilities, heuristics about the domain used to deal with uncertainty, e.g. activity X is more likely than activity Y, can be captured and modelled. In addition approaches such as HMM, DBN and CRF that inherently support handling of temporal information provide better support for modelling and recognition of complex activity patterns, e.g. interweaved activities. The main criticism that has been made on these techniques is that it could become computationally expensive to learn activity models when there is a large diversity of activities. Furthermore, the learnt models may have to be revised due to variations in an actor's behaviour and environment. In addition, probabilities may not be expressive enough to intuitively model certain domain knowledge concepts.

In contrast, knowledge-driven activity recognition is inspired by logical modelling and reasoning. It uses logic-based knowledge representation to model activities and sensor data and then exploits logical reasoning for activity inference. In the literature, several methods have been explored such as the use of event calculus [31], description logic and lattice theory [16], description logic [4, 20], temporal reasoning and active databases [32], spatiotemporal reasoning [21] and spatiotemporal and context reasoning [17]. Given that knowledge-driven activity recognition is grounded on logic theory, the resulting activity models are semantically clear and elegant. In addition, it is easy to capture and model domain structure and heuristics. The main criticism is that these approaches handle uncertainty poorly. In addition, it is difficult to find the most optimal model of the activities and sensor data. Further, since learning ability is not inbuilt into these models, adaptive capability must be deliberately implemented to deal with variations in activities attributed to changes in the actor's behaviour and their environment. Of particular interest is the use of ontologies in activity recognition - an area of growing interest [4, 20][8, 9]. The key idea is to use ontologies to describe the actor's environment and provide semantics that automated systems can easily reason with. In [8, 9], the authors use ontologies during activity modelling to describe items in the domain. The resulting models are processed for activity inference using independent algorithms, i.e. finite state machines [9] and probabilistic inference [8], respectively. However, in [4, 20] the authors describe a different approach that integrates both modelling and activity recognition into a unified framework. The presented approach models ADLs using web ontology language (OWL) [33]-based ontologies and then use description-logic based reasoning [34] to infer ongoing activities. They use ADL ontologies to represent explicit activity models by creating description based models of the activities and sensor data. By combining activity modelling and recognition in this way, agents can perform knowledge-based intelligent processing to infer activities.

2.2 Temporal Segmentation in Activity recognition

The issue of sensor data segmentation in knowledge-driven activity recognition has received little attention in existing work. For instance, in [20] the authors present a knowledge-driven activity

recognition approach but do not provide the details of the method for sensor data selection. Despite showing that ontology-based activity recognition is feasible, the absence of a suitable method for sensor data selection makes the presented method difficult to replicate. However, another knowledge-driven activity recognition method presented in [17] uses competing hidden Markov models to segment a sensor data stream. The selected sensor data is used to perform spatiotemporal and context reasoning for activity recognition. They use a variable window length and the window moves over a sequence of observations. The main weakness of this approach is that it requires a pre-existing dataset to determine the optimal size of the time windows and the segmentation rules that it uses. Since the same individual or different individuals may perform the same activity in many different ways, this method will be difficult to reuse. In addition, the derived optimal window lengths have to be revised to deal with new situations.

The use of one minute time slices to evaluate the effectiveness of ontology-based activity recognition is presented in [18]. Sets of sensor data are selected every minute for activity inference. The work is based on van Kasteren dataset [35] and the main limitation is that the ontology used is modelled on and closely tied to the dataset making it difficult to re-use. In addition, the fixed-size time slices used may lead to a huge computational expense since the activity inference engine is forced to periodically sample the data stream even when no new sensors have been activated. Furthermore, its ability to support real-time activity recognition has not been discussed. Ontologies and video are used for activity recognition in [9], whereby an ontology-based knowledge base supports the recognition of human activities from video sequences. The ontology models human activities, in terms of entities, environments and interactions, and creates semantic links between events and activities. Vision-based techniques are used to select the input data for activity inference based on a pre-existing dataset. Our work is modelled on a dense sensing framework, and as a result the computer vision based techniques used in [9] are less suitable. However, the authors adopt a method to select the input data used in activity recognition which is comparable to the problem that we aim to address, i.e., to select a subset of sensor data for activity inference. From the foregoing, it is clear that in most knowledge-driven activity recognition work the method used to select sensor data is either non-existent or, at best, ad hoc. There is a need to develop a systematic approach that can be applicable in different knowledge-driven activity recognition approaches to help segment and then aggregate sensor data.

In the data-driven activity recognition community, the problem of sensor data segmentation has been widely explored [1, 12, 29, 35-37]. The notion of time windows is adopted to provide a basis for handling time-dependent data, e.g., the sensor data stream. However, some sensor data segmentation approaches use static sliding windows to segment the data stream [1, 12, 35] while others use dynamically derived time window lengths [29, 36, 37]. The notion of time slices is used in [35] to derive segments used to perform activity recognition. In [1] and [12], a sliding window method is used to derive features used in activity inference by the proposed algorithms. The time windows used in [1] are made to have 50% overlap. The main criticism for static sliding windows is that incorrect lengths can truncate an activity instance or overlap activity instances leading to recognition failure. Due to the above problem, our work is closely related to [29, 36, 37] whereby time window parameters are varied.

The work in [29] uses temporal information (i.e., the average activity duration) to set different length values to the time windows at initialization; however, once a time window is activated its length cannot be dynamically modified. This can cause the time window to overlap the end of one activity and the beginning of the next one, thus leading to recognition failures. The notion of time contiguity in sensor data is used together with location context to segment sensor data from state change sensors in [36]. Any noted changes in location context between two consecutive sensors are used to signify a *break point.* The break point then helps identify the start and end of segments on which activity inference is eventually performed by evidential fusion [38]. This approach will work well when consecutive activities occur in different locations; however, segmenting the sensor data stream arising from the same location may prove difficult if the break point is not detected. Since, recognition is only attempted on a segment after the start and end points are identified, this approach assumes that the user always performs activities correctly, making diagnosis that is necessary for activity assistance difficult or impossible.

Although our work is in the area of knowledge driven activity recognition, it is slightly similar to the work in [37] since it uses dynamic windows. In [37], the length of the window is dynamically derived at runtime based on the occurrence of specified low-level events, e.g. the change of sensor state. The key difference with our work is that while they use primitive events to dynamically manipulate time window parameters, we use high level context information such as activity duration, and the current status of recognition resulting from a high-level activity inference event. As a result, our approach is able to utilize high quality knowledge in sensor data segmentation

Following the above discussion, we contend that by capturing some temporal features of sensor data and by extension that of activities, the sensor data stream can be broken into segments for real-time activity recognition. We use dynamically varied time windows based on the temporal information of

activities to support this segmentation and activity recognition is then performed on these segments. The main strengths of the proposed approach are that it is: (1) systematic; (2) simple, well-defined and easy to implement; and (3) not specific to any user or dataset; hence, can be replicated.

3 REAL-TIME CONTINUOUS ACTIVITY RECOGNITION

3.1 Ontological Activity Modelling

Ontological modelling allows the creation of logical activity models to formally conceptualize the Smart Home domain. Activity models are based on objects, environmental elements, events, and interrelationships (e.g. "is-a" and "part-of" relations) between activities. Ontological activity modelling encodes activities as ADLs and uses ontologies to represent this knowledge for use in activity recognition. The resulting activity models can be processed by an automated system, through semantic reasoning, to directly infer activities. The ADL ontologies capture the contextual information of activities in an SH domain.

Typically, inhabitants of a SH perform routine ADLs in specific locations, with certain objects, and at particular times. For instance, an inhabitant may prepare a glass of juice in the evening after dinner. This may involve the use of the fridge and the glass cupboard, both of which are located in the kitchen. This information is generally referred to as the context and can be associated with a specific activity. This contextual information, together with other information, e.g. the different ways a person performs the same activity, is part of domain knowledge. By using activity modelling, this domain knowledge can be represented as ADL ontologies.

Ontological modelling allows ADL activities to be structured in a hierarchical tree with the most specific ADL descriptions represented as leaf concepts - all leaf concepts have no child classes. Each concept is associated with a number of role restrictions. All child concepts inherit all the roles of their parent concepts but may specify further constraints. In this way, ontological ADL models can facilitate progressive activity recognition of both generic and specific activities. A generic activity refers to an ADL class that has associated descendant classes. On the other hand, a specific activity (the so-called *leaf activity*) is an activity with no descendant classes in the ontology. For instance, 'make drink' is a generic activity, while its descendants 'make tea' and 'make coffee' are specific activities.

3.2 The Approach for Continuous Activity Recognition

Continuous, real-time activity recognition helps to identify ongoing ADLs as they occur, thus offering the possibility to provide timely assistance for SH inhabitants. In ontology-based activity recognition, when an ADL is performed along a timeline, the contextual information associated with the ADL is captured incrementally and subsumption reasoning is used to infer the ongoing ADL. At the initial stages of an ADL, subsumption reasoning may only classify the contextual information to a generic ADL class. However, as more contextual information is obtained over time, and reasoning is continuously performed, it would be possible to recognize the specific ongoing ADL.

In a dense sensing based SH, contextual information is captured through a variety of sensors, with each sensor representing a particular view of the prevailing situation. From the activated sensors, an agent can infer physical and contextual entities, e.g. objects, locations, times, and events. For example, a pressure sensor can be attached to the sofa in the lounge. Given that this knowledge is explicitly encoded in the ADL ontology, whenever this pressure sensor is activated it is possible to infer that the inhabitant is in the lounge and is sitting on the sofa. Consequently, this allows the inference of an activity that occurs in the lounge while sitting on the sofa, e.g. *reading a book* or *watching television*. Since most ADLs require the fusion of data from multiple sensors over time to infer high-level activities, it is necessary to first aggregate a sequence of sensor activations in order to generate a situation at a particular time point during activity recognition. For a more detailed description of the ontology-based activity recognition approach, we refer the interested reader to [4] due to space limitations.

While work in [4] described the rationale and algorithm for semantic reasoning for activity recognition, it did not present any details about sensor data segmentation that is critical for continuous real-time activity recognition. In this paper, we focus on extending the ontology-based approach in [4] with a sensor data segmentation mechanism so as to support real-time activity recognition. Fig. 1 shows the three-layer architecture for the extended approach, namely context selection, iterative action inference, and activity recognition layers.

Whenever activities are performed, the incoming sensor data is received as a sensor data stream and segmented in the *context selection layer*. Context selection refers to the process by which the stream is divided into a set of fragments using temporal segmentation. In temporal segmentation, the stream is analyzed along a temporal dimension using the temporal properties (of both sensor data and activities) captured by time windows. This ensures that only those sensor activations occurring within a given time window are included in the segment. To achieve its goal, this layer uses activity monitoring and dynamic segmentation components. The *activity monitoring* component allows sensor data to be received, while

dynamic segmentation component implements the temporal segmentation algorithm. We model the notion of a time window as a data structure made up of a number of parameters. Section 4 provides the formal model of the time window mechanism, provides a detailed description, and the algorithm that utilizes it in activity recognition.

The *iterative action inference layer* analyzes the segments of the data stream that are generated in the context selection layer to identify a collection of actions (also called low-level activities) associated with the activated sensors. Typically, a time window's sensor activations are processed against the ADL ontology to determine the ongoing primitive action, e.g. 'cup is used'. This action can be represented as context information in the ontology by a property assertion that is equivalent to the description: '*hasContainer property associated with unknown ADL activity X has value cup'*. This process is repeated for all sensor activations that have so far been received in the time window. A collection of such low-level (simple) activities may combine to constitute the activity description of one or more high-level (complex) activities. The *activity inference layer is* responsible both for the inference of ongoing activities and the initiation of dynamic modification of time window parameters. To this end, it is made up of two main components, namely aggregation and high-level inference components. The *aggregation* component collects the individual property assertions from the iterative action inference layer together to derive the overall description of the current activity. The resulting activity description is passed on to the *high-level activity inference* component. The *high-level activity inference* component is made of *activity inference* and *time window manipulator* components. The *action inference* component uses the activity description, ADL ontology, and ontological reasoning to infer the ongoing ADL, e.g. 'make tea'. If a specific ADL is inferred, the recognition process is considered successful and the result is reported. Otherwise, a generic ADL is reported and the system will wait for additional sensors to be activated before attempting recognition again. In this way, ongoing ADLs can be progressively inferred. The system can dynamically initiate the shrinking or expansion of time windows whenever necessary through the *time window manipulator* component. The mechanism for shrinking and expansion is described in the next section. To ensure perpetual, real-time activity recognition, the entire process continues to run with new time windows continuously and dynamically generated.

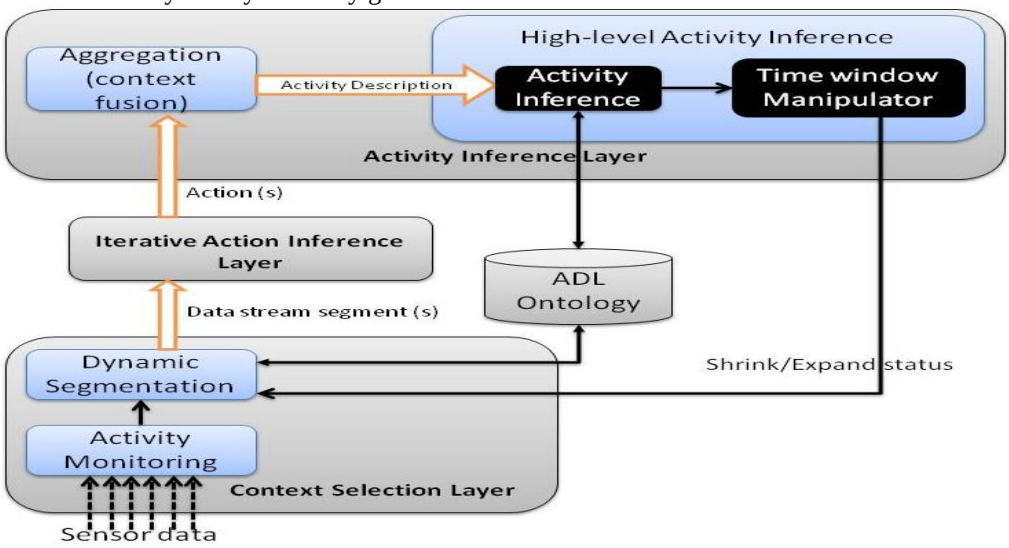


Figure 1: Architecture of real-time activity recognition approach

4 SENSOR DATA SEGMENTATION AND ANALYSIS

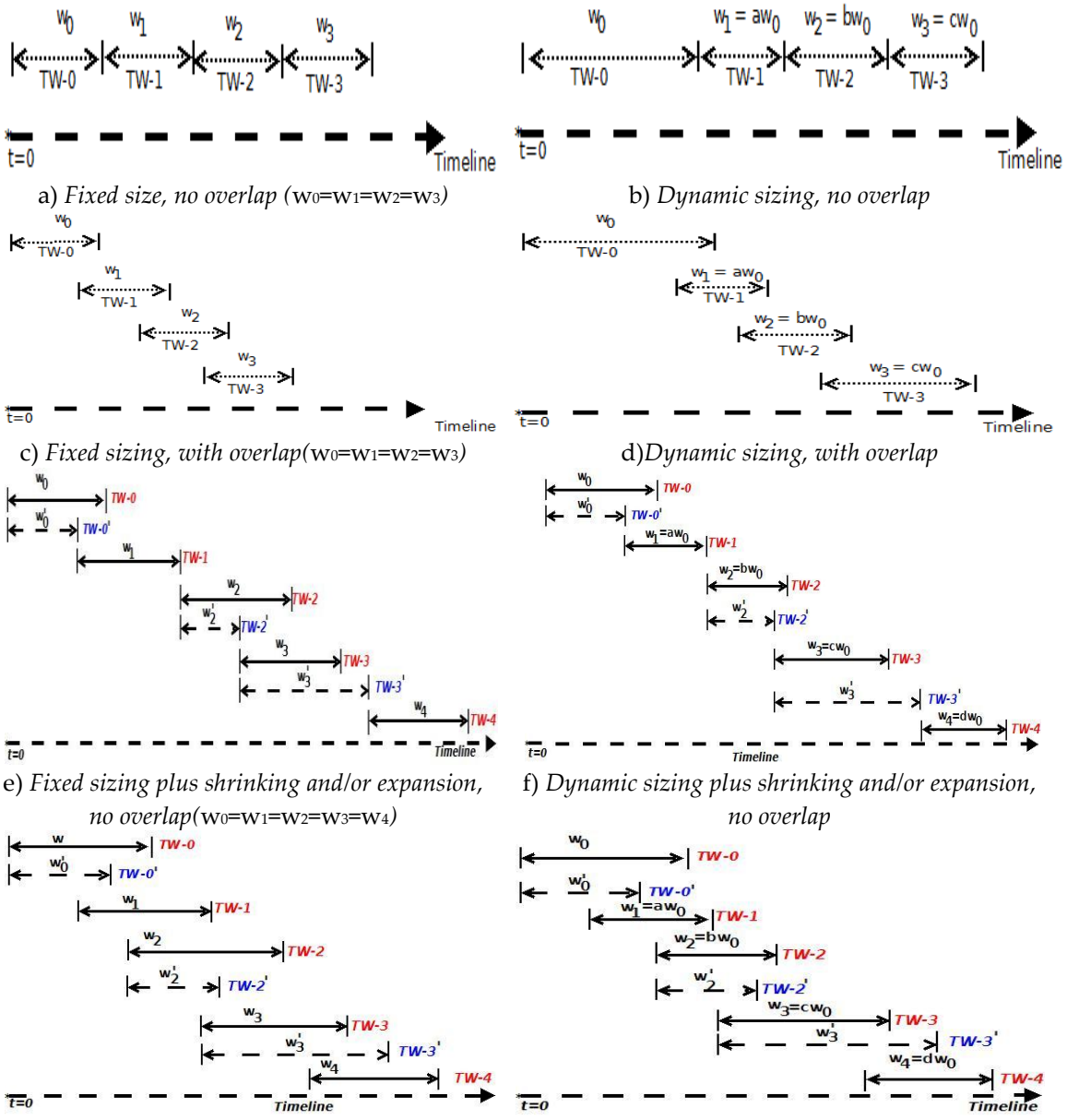
4.1 Characterisation of Segmentation and Recognition

A key factor in continuous real-time activity recognition is how to select the set of activated sensor data to be aggregated for activity classification. In a typical Smart Home, sensors will be continuously activated and the resulting sensor data sequence needs to be broken down into fragments that can be mapped to specific ADL activities. To segment a sensor data stream, this work presents a number of scenarios and configurations that can be considered. The scenarios are divided into two main categories: overlapping and non-overlapping time windows. In overlapping time windows, two or more distinct time windows can share some activated sensors. On the other hand, whenever non-overlapping time windows are used, no single activated sensor is shared by two or more time windows.

Under each category, there are four scenarios to be considered. The first scenario uses fixed-sized time windows, whereby all time windows are created of the same size (i.e., given the initial time window has length w_0, any newly created window will also have the length set to w_0). The second scenario uses

variable-sized time windows. In this case, the lengths of newly created time windows are dynamically derived at run-time such that the length of any new window is a multiple of that of the initial window (i.e., given the initial window has length w_0, any new window will have the length set to $a*w_0$, where a is a positive real number). Regarding both scenarios, a key challenge is how to choose optimal sizes at runtime. The third and fourth scenarios allow time windows to be dynamically shrunk and/or expanded at runtime as a result of activity inference. Scenario three is a variation of scenario one because it uses fixed-sized time windows. Similarly, scenario four is a variant of scenario two. A key challenge is the criteria for triggering shrinking or expansion of a time window. The resulting eight distinct configurations are depicted in Fig. 2 (a)-(h).

From the scenarios provided, it is clear that although the task of segmenting a sensor data stream with time windows is complex, there are various methods that can be used to achieve it. However, choosing the most suitable method for segmentation is a non-trivial task. Providing support for the different configurations described requires careful design of the time windows together with an appropriate choice of the parameters and strategies for manipulation. In the next sections, we present a time window based approach and algorithms that model and implement the presented scenarios.



a) *Fixed size, no overlap* ($w_0=w_1=w_2=w_3$)

b) *Dynamic sizing, no overlap*

c) *Fixed sizing, with overlap*($w_0=w_1=w_2=w_3$)

d)*Dynamic sizing, with overlap*

e) *Fixed sizing plus shrinking and/or expansion, no overlap*($w_0=w_1=w_2=w_3=w_4$)

f) *Dynamic sizing plus shrinking and/or expansion, no overlap*

g) *Fixed sizing plus shrinking and/or expansion, with overlap*($w_0=w_1=w_2=w_3=w_4$)

h) *Dynamic sizing plus shrinking and/or expansion, with overlap*

Figure 2: Representation of sensor data segmentation scenarios

4.2 Sensor Data Segmentation Mechanism

This paper presents a mechanism that uses time windows to decide on which sensor activations to use for activity inference. The mechanism utilizes a sensor data segmentation model that is modelled by a time window data structure. To implement the different configurations shown in Figure 2, the time window data structure provides various parameters. Some parameters can be preset but can remain unchanged or be varied, while others are dynamically set at runtime. The time window model and its parameters are described in the next section.

To illustrate the use of the time window model, the scenario in Fig. 2 (e) that allows windows to be shrunk and/or expanded was selected. The lines marked in the form **TW-N'** indicate that the corresponding initial time window has been shrunk or expanded. In the diagram, we have two windows that are not modified (TW-1, TW-4), two that are shrunk (TW-0, TW-2), and one window (TW-3) that is expanded. In the current work, only non-overlapping time windows are investigated. During expansion, the time window's length is extended so as to accept further sensor data. This occurs whenever additional sensor data is required to successfully infer an activity but the pending window length would be inadequate to cover the given activity's duration as provided in the activity ontologies. In addition, a window can be expanded whenever a generic activity has been identified but the pending window length is inadequate and thus requiring additional time to successfully infer any of the specific descendants of the activity.

Conversely, during shrinking the time window is truncated before its preset length is exhausted. Typically, as soon as an ADL is recognized, the ADL ontology is used to determine whether additional sensor activations should be anticipated or not. In addition, the identified activity's duration information available in the activity ontologies can also be used to determine if the time window should be truncated. If there are no further sensor activations expected or the duration has been exhausted, the current time window is closed and a fresh time window activated. Alternatively, if it requires additional sensor activations or the duration has not been exhausted, then the time window will continue to be active until either its preset length is exhausted or further sensor activations are obtained. In addition, an assistive system can be invoked to provide some interventions, e.g. prompts and suggestions, to the user in an Ambient Assisted Living (AAL) environment.

4.3 Formal Time Window Modelling and Manipulation

We propose a formal time window model whose characteristics and operation are described below. We define a number of parameters to describe how the time window is manipulated. Significant parameters include start time, end time, window length, the enclosed collection of sensor data, and overlap, shrinking and expansion capabilities. Other parameters are used to provide a means for manipulating the time window data structure. Some (dependent) parameters (e.g. end time) are assigned dynamically during recognition processes, while the independent ones (e.g. window length) are preset.

4.3.1 Definitions

Let:

- **α**: the start time for a time window
- **ω**: the end time for a time window
- **w**: the length of a time window
- $\boldsymbol{\Omega_\alpha}$: a time window whose start time is α.
- **Ψ**: sensor data set. This is a data structure for storing the set of sensor data belonging to a given time window.
- **A**: a vector of activity labels assigned to the time window after activity inference.
- **γ**: reasoning start mode. Used to determine when to trigger activity inference.
- **ρ**: time window factor. Used to derive the size of a new time window from the initial time window.
- **μ**: sliding factor. Used to determine the size of the slide applied to the active time window to move it over the sensor data stream.
- **δ**: change factor. Used to determine the magnitude used to expand or shrink the length of a time window.

We can define a time window, Ω_α, as a 9-tuple with nine properties: Ψ, α, ω , w, γ, ρ , μ, δ, and A as shown in the expression below:

$$\Omega_\alpha:< \Psi, \alpha, \omega, w, \gamma, \rho, \mu, \delta, A> \quad (1)$$

The end time, ω, can be computed from the start time, α, and window length, w, as shown below:

$$\omega= \alpha + w \quad (2)$$

Given that a sensor activation arriving at time, t, is denoted by sa_t; Ψ can be defined below:

$$\Psi :\{sa_t| \ \alpha \leq t \leq \omega, \text{ for all } t \} \quad (3)$$

4.3.2 Time window manipulation

A number of operations can be performed on the time window model, namely sizing, activation, deactivation, sliding, shrinking, expansion and overlapping.

Sizing: This sets the initial length of the time window. To determine the length of time windows, let the length of the initial time window $\Omega_{\alpha\text{-}0}$, be set to w_0. The length of each time window is delimited by a minimum size, w_{min}, and a maximum size, w_{max}. The values of w_{min} and w_{max} are obtained from activity duration information that is derived from prior domain knowledge. For instance, w_{min} is set to the duration of the shortest activity, while w_{max} is set to that of the longest activity plus some slack time. Typically, given the initial time window, $\Omega_{\alpha\text{-}0}$, then the length of any new window, $\Omega_{\alpha\text{-}i}$, can be assigned using the formula below:

$$w_i=\rho * w_0,\ w_{min} \leq w_i \leq w_{max},\ i=1,2,...n \quad (4)$$

The value of ρ is chosen such that the resulting time window length lies between w_{min} and w_{max}. To minimize computational complexity, both w_{min} and w_{max} are set constant for all time windows. In this paper, the value of $\rho=1$ is chosen to set the default size of all time windows as equivalent to the initial time window. However, the default size is dynamically varied through the *shrinking* or *expansion* operations.

Activation/Deactivation: A Boolean flag, *activated*, is used to activate or deactivate a time window. It is set to *true* to indicate that the time window is active and *false* to show deactivation. By default, the flag is set to *false* and must be changed to *true* so as to use the time window model to segment a sensor data stream. Before the deactivation operation, the state of the time window must be logged in a suitable storage.

Sliding: The sliding operator allows the shifting of the current time window by some factor in order to derive a new time window. To determine the criteria for sliding, a sliding factor, μ, is defined. The sliding factor is a value that satisfies the constraint $0 < \mu \leq 1$. In this way we can obtain the size of the slide that needs to be applied to the current time window. The slide determines by how much the start time for the current time window, $\Omega_{\alpha\text{-}i}$, is shifted forward to determine the start time of the new time window, $\Omega_{\alpha\text{-}j}$. Let the slide to be applied to the current window to obtain the start time for the next time window to be denoted by θ_i. We can compute θ_i by using the following formula:

$$\theta_i = \mu * w_i,\ i=0,1,2,...,n \quad (5)$$

Given that the start time for the current time window is denoted by α_i and that of the succeeding time window by α_{i+1}, we can apply the slide to derive α_{i+1} using the formula:

$$\alpha_{i+1} = \alpha_i + \theta_i \quad (6)$$

Overlapping: This refers to the process of having two or more time windows active at the same time. By choosing a sliding factor value less than one ($\mu < 1$) two time windows are made to overlap. A value of one ($\mu=1$) means that the time windows are successive and non-overlapping. Furthermore, by examining the time windows being created and activated we can identify two properties:

Property (1): Two time windows, $\Omega_{\alpha\text{-}i}$ and $\Omega_{\alpha\text{-}j}$, i<j, and $\Omega_{\alpha\text{-}i}$ starts before $\Omega_{\alpha\text{-}j}$, are said to overlap in time if the start time, α_j, of $\Omega_{\alpha\text{-}j}$ is less than the end time, ω_i, of $\Omega_{\alpha\text{-}i}$. This is denoted by the expression below:

$$\alpha_j < \omega_i \quad (7)$$

Property (2): Two time windows, $\Omega_{\alpha\text{-}i}$ and $\Omega_{\alpha\text{-}j}$, i<j, are said to overlap in activations if property (1) is true and the intersection between the data sets in the two time windows is non-empty, i.e., at least one sensor activation belongs to both time windows. The non-emptiness is denoted by the following expression:

$$\Psi_i \cap \Psi_j \neq \emptyset \quad (8)$$

Whenever property (2) is satisfied, sensor activations can be used in two or more time windows during activity inference. This scenario can be used to facilitate the recognition based on complex activity patterns such as interleaved and concurrent activities.

Shrinking/Expansion: Given that the time window size is preset, an ADL may be identified before the expiry of the window. Whenever this happens, the time window length may be reduced dynamically (this is called *shrinking*). Conversely, whenever it can be established that the time window may expire before an ongoing ADL is conclusively identified, the window length can be increased (this is called *expansion*) to keep the time window active for a little longer. To perform the shrinking operation and to compute the new window length, w'_i, we define the shrink time, *st*. Shrink time, dynamically derived at runtime, refers to the time at which the decision to shrink the current time window is made. The new window length, w'_i, is then computed using the formula below:

$$w'_i=st- \alpha_i \quad (9)$$

Similarly, to expand we define the length of expansion, *exp*. The new window length is then computed using the formula below:

$$w'_i=w_i+exp \quad (10)$$

4.4 Algorithms for Continuous Activity Recognition

Once sensor activations are received, then by using the ADL ontology, each of them is converted into the corresponding ADL property assertion and added to a set of property assertions. At an appropriate time, the reasoning engine attempts to recognize the ongoing ADL. There are three modes that can be used to trigger reasoning. In the first mode, γ=0 and each time a sensor is activated, activity recognition is performed. Using the second mode γ=1 and reasoning occurs periodically at regular intervals during the length of the time window. The intervals can be set at configuration time. The activity inference engine should check the existence of new sensor activations before further attempts at reasoning. This requires that the current and previous sensor states are tracked to determine whether or not fresh activations have been obtained. Finally, using the third mode γ=2 and reasoning occurs only at the expiry of the time window. The success of this mode depends entirely on optimal choice of time window lengths. At the deactivation of each time window, all sensor activations used within it are discarded.

4.4.1 Recognition algorithms

In order to support continuous, real-time activity recognition we modify the algorithm in [4, 20]. To manipulate the time window data structure the ontology-based activity recognition algorithm is enhanced with temporal segmentation ability as shown in Fig. 3 and 4. Fig. 3 shows the pseudo-code for the time-window based algorithm, and Fig. 4 shows the pseudo-code for the ontological reasoning component of the algorithm.

Three operation modes are proposed to support the manipulation of time windows and to demonstrate the impact of dynamic manipulation. The first is ***no-shrink-no-expand***, whereby the time window is effectively static and the size is not reduced or extended at runtime. The second is ***shrink-only*** mode for which the length of a time window can be reduced but cannot be extended. Finally, in ***shrink-and-expand*** mode, the length of a time window can be extended, reduced or both. The *shrink-only* and *shrink-and-expand* modes show the use of dynamic time windows. The mode used can be specified at configuration time. The pseudo-code in Fig. 4 supports both shrinking and expansion whenever ***shrink-only*** or ***shrink-and-expand*** modes are selected.

```
Input: Receives the time window data structure (Ωα) and the ADL ontology (ADL-O)
Output: A matrix of time windows and corresponding text strings representing the likely activity labels, V.
```

```
RECOGNIZE-ADL (Ωα, ADL-O)
Set the initial time window
While active Do
   If initial window Then Activate time window End
   While time window unexpired Do
      Obtain and add sensor activations to Ωα
      If overlapping=true And overlapping window not active Then
         Compute slide and derive next time window
         Activate newly derived window
      End
      If reasoning mode  on-sensor Or at-intervals Then
        DO-ONTOLOGICAL-AR(Ωα, ADL-O)
       Else If reasoning mode = at-intervals And interval-elapsed Then
        DO-ONTOLOGICAL-AR(Ωα, ADL-O)
      End
    End(inner loop)
       If reasoning mode  on-expiry Then
       DO-ONTOLOGICAL-AR(Ωα, ADL-O)
       End
    Update matrix (V)
    Discard previous time window's sensor activations
    If overlapping=false Then
        Derive new window
        Activate time window
    End
End (outer loop)
Return (V)
```

Figure 3: Time-window segmentation based ontological activity recognition algorithm

4.4.2 Algorithm for shrinking time window

A time window can be shrunk under two conditions. Firstly, if all property assertions needed to

describe a leaf activity have been specified, then the recognition system can truncate the current time window and spawn a new window. This is done by checking the activity description derived from the time window against the restrictions that have been defined for the given ADL class in the ontology. Secondly, the recognition system can choose to truncate the current time window if it determines that the ongoing activity has exhausted its duration and hence it is least likely to generate further sensor activations. Both cases have been captured by the pseudo-code in Fig. 5.

```
Input: Receives the time window data structure (Ωα) with sensor activations
and the ADL ontology (ADL-O)
Output: A vector of text strings representing the likely activity labels, A.

DO-ONTOLOGICAL-AR (Ωα, ADL-O)
Derive property assertions
Derive activity description, A_desc
Perform equivalency and subsumption reasoning with A_desc to determine
underlying activity (ies), activity-1, activity-2,.., activity-n
If only one activity, e.g. activity-n, is obtained Then
  Check whether activity-n is abstract or specific
  If activity-n is specific Then
    Report activity-n is successfully identified
    If shrinkable Or shrinkable-and-expandable Then
          ATTEMPT-SHRINK (Ωα, ADL-O, activity-n)
          If window is not shrunk Then
              Report that more sensor data are needed
           ATTEMPT-EXPANSION (Ωα, ADL-O, activity-n)
   End
  End
  Add the activity-n label to vector,A
  Else
   Report that more sensor data are needed
    If shrinkable-and-expandable Then
          ATTEMPT-EXPANSION (Ωα, ADL-O, activity-n)
    End
Else
  Add all activity labels to vector, A
  Obtain generic activity label, parent-activity, from A
  If shrinkable-and-expandable Then
     ATTEMPT-EXPANSION (Ωα, ADL-O, parent-activity)
  End
End
Return (A)
```

Figure 4: Ontological reasoning algorithm (adapted from [4])

```
Input: Receives the time window data structure (Ωα) with sensor activations,
the ADL ontology (ADL-O), and activity label (activity-n)
Output: A truncated time window Ω'α

ATTEMPT-SHRINK (Ωα, ADL-O, activity-n)
If all properties of activity-n are asserted Then
   Shrink the window
Else If activity-n duration exhausted Then
   Shrink the window
Else
      If time-needed-to-complete-activity-n >= pending-window-length Then
       ATTEMPT-EXPANSION (Ωα, ADL-O, activity-n)
    End
End
Return (Ω'α)
```

Figure 5: listing to shrink a time window

4.4.3 Algorithm for expanding time window

A time window can be expanded under two conditions. Firstly, given that a leaf activity has already been identified but the pending time window length is inadequate to complete the activity description, the window is expanded to allow additional activated sensors to be obtained. Secondly, given that a leaf activity has not been identified, information about the currently identified generic activity is used to determine how much additional time would be needed to recognize its subclass that has the longest duration. The pseudo-code in Fig. 6 depicts this scenario.

```
Input: Receives the time window data structure (Ω_α) with sensor activations,
the ADL ontology (ADL-O), and activity label (activity-n)
Output: An expanded time window Ω'_α
ATTEMPT-EXPANSION (Ω_α, ADL-O, activity-n)
If activity-n is specific Then
 If some properties of activity-n are missing And needed time to complete ADL
 >=
    pending time window length Then
      Expand the window
  End
Else
      If some activations have been obtained Then
       Obtain subclasses of activity-n
        Derive maximum duration from the durations of these subclasses
        Obtain remaining duration to complete longest subclass, remainderADL
        If remainderADL >= pending time window length Then
           Expand the window
         End
       End
End
Return (Ω'_α)
```

Figure 6: listing to expand a time window

5 IMPLEMENTATION AND EVALUATION

We have applied the proposed approach to develop a SH-based activity recognition system. The system is implemented using Java language and a raft of semantic technologies and tools. Specifically, we developed ADL ontologies based on OWL-DL [33] using Protégé editor [39] as shown in Fig. 7. The ADL ontology captures information about ADLs such as ADL concepts, hierarchical relationships among concepts, property restrictions for ADLs and contextual information, and sensor related concepts.

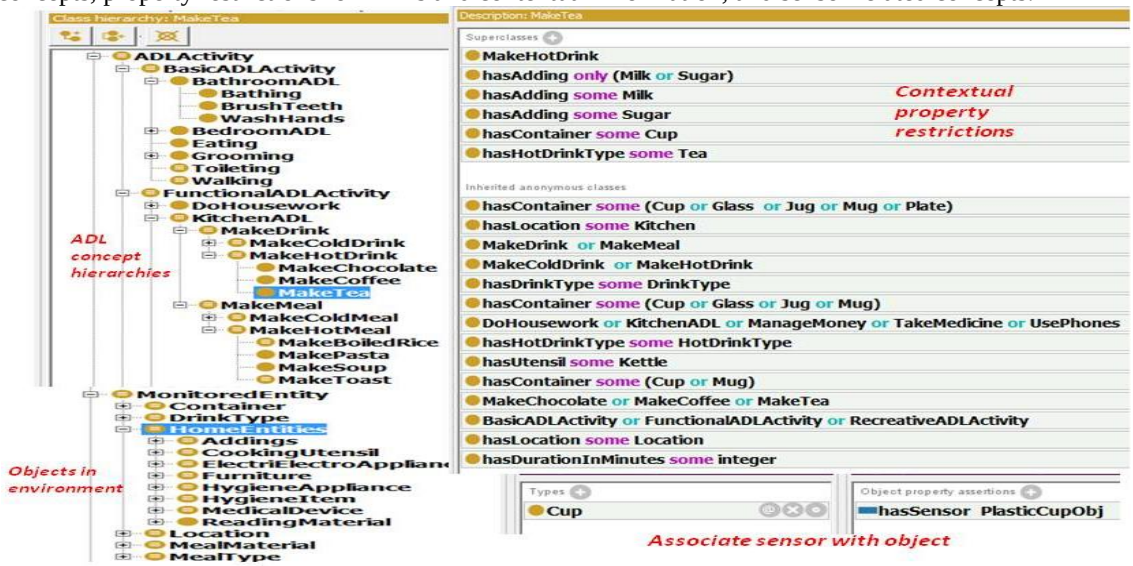


Figure 7: Fragment of ADL ontology

To support ontological reasoning we have used Pellet [40] OWL reasoner, accessed through application programming interfaces (APIs) in Java, to provide reasoning capabilities for activity inference. In addition, we implemented the time window based segmentation model as part of an activity recognition module. Fig. 8 shows four system interfaces of the implemented system. Firstly, on the top-left it shows the interface that displays the set of all sensors that are currently deployed in the environment. Secondly, the top-right of Fig. 8 displays the configuration window that is used to add sensors to the SH environment, set the initial time window parameters, and to initialize the activity monitoring task. Thirdly, the dialog for choosing whether to monitor sensor activations in real-time or to play them back from a file is shown on the bottom-left. Finally, the list of all sensors that have been activated during a particular time window as well as the status of activity recognition is provided at the bottom-right of Fig. 8.

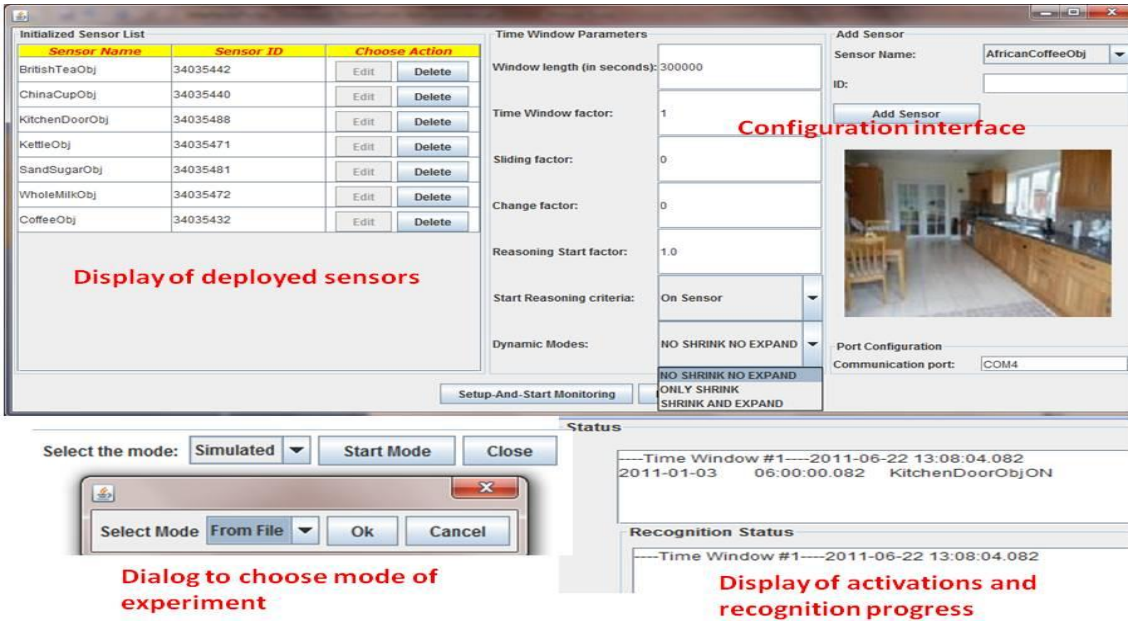


Figure 8: System configuration and status display interfaces

5.1 Experiment Design

To evaluate and demonstrate the feasibility of the proposed approach, we developed a synthetic data generator that can be used to generate synthetic ADL data. This provides ADL data items that possess the necessary temporal information and allows us to quickly evaluate the feasibility of the developed approach before deployment in a real-world environment. Another advantage is that we are able to test the approach on different datasets. To facilitate data generation, we specified 'seed' ADL patterns at the start. Each seed ADL pattern is described by a sequence of ADLs. The synthetic ADL data generator then derives different permutations of these patterns. To select the permutation to use, it uses a random number generator. In this way, each permutation is given an equal chance of being considered an ADL pattern during dataset generation.

To generate the synthetic ADL data, eight typical ADLs related to meals (e.g. MakeTea, MakeCoffee, MakeChocolate, and MakePasta), hygiene (e.g. HaveBath, BrushTeeth, WashHands) and recreation (e.g. WatchTelevision) were used. In addition, to ensure that data is rich with useful temporal information, synthetic ADL data is generated corresponding to three time periods per day: morning (6am-9am), midday (12pm-2pm), and evening (6pm-10pm). To facilitate this, the time period to which the ADL can be performed is specified when adding possible seed ADLs to the synthetic data generator. Similarly, when specifying seed patterns the time period to which an ADL pattern belongs is provided. Finally, the transition time (in seconds) between ADLs is specified for each ADL pattern. For example, the pattern *MakeTea-0, BrushTeeth-600*, implies that MakeTea is the first ADL in the pattern, while the ADL BrushTeeth will occur 600 seconds after MakeTea is completed. Currently, we have made the assumption that only one ADL can be performed at the same time. As a result, no two sensors can be activated concurrently during data generation.

For each ADL considered, we provide one or more patterns of sensor activations. Given that the same ADL may be performed in a variety of ways; these patterns depict the various ways. To incorporate more temporal meaning, each sensor in a pattern is activated after a given amount of time after the immediately preceding activation. By implication, this ensures that duration information of ADLs is included when synthetic ADL data is generated. The text SensorObj@n in a pattern means that the sensor object labelled *SensorObj* is activated ***n*** seconds after the preceding sensor object is activated. As an example, MakeTea is represented by the following patterns of activated sensors.

- **Pattern#1**: KitchenDoorObj@0, KettleObj@20, CupObj@180, TeaObj@20, MilkObj@20, SugarObj@20 (duration=260 seconds)
- **Pattern#2**: KitchenDoorObj@0, KettleObj@20, CupObj@180, TeaObj@20, MilkObj@20 (duration=240 seconds)
- **Pattern#3**: KitchenDoorObj@0, KettleObj@20, CupObj@180, MilkObj@20, TeaObj@20, SugarObj@20 (duration=260 seconds)

Each sensor pattern is captured in a collection data structure; with a collection available for each ADL. Similar sensor activation patterns are specified for the other ADLs too. To select the sensor pattern for any ADL, the synthetic data generator uses a random number generator to randomly select the index of the

sensor pattern for the relevant ADL from the relevant collection. This eliminates bias and gives each pattern a fair chance of being selected.

To test the time window approach and associated algorithms, a simulation tool has been built to mimic the activation of sensors in a dense sensor based deployment. The simulation tool plays back the synthetic ADL data generated described above and feeds the sensor data to the activity recognition system as if the sensor activation is occurring in real-time. As the data is played back, the recognition engine tries to identify the ongoing ADL and displays the status on the interface shown at the bottom-right of Fig. 8.

5.2 Experiments and Results

5.2.1 Time-window model configuration

To carry out the experiments, the duration of the default time window is initially set to a value slightly greater than the longest ADL - in the experiments the longest ADL by duration is MakePasta. The reasoning start mode ( ) is set to zero (0) so that activity inference is attempted each time sensor activation is obtained. The sliding factor ( ) is set to one (1) to indicate that time windows are consecutive and non-overlapping. The time window factor ( ) is set to one (1) so that each time window is by default the same size as the initial window. Finally, the change factor ( ) is computed at runtime during shrinking and expansion operations. Similarly, the other parameter (i.e. start time ( ), end time ( ), sensor data set ( ) and the vector of activity labels ( ) are dynamically computed at runtime.

5.2.2 Ground-true Synthetic ADL data

To facilitate analysis, we generated synthetic ADL data for four weeks based on the eight ADLs above. The dataset contains a total of 154 ADL activities and a summary of the ADLs is provided in Table 1. Three variables are used to describe the dataset. These are: 1) %*in-pattern ADLs*- describes proportion of the ADL that appear in ADL patterns that have at least two ADLs; 2) %*standalone ADLs*- describes the proportion of the ADL that appear in single-ADL patterns and; 3) the *total number of times a given ADL occurs in the dataset*. Generally, ADLs that participate in many ADL patterns (i.e., MakeTea,MakePasta, BrushTeeth, HaveBath and WatchTelevision) have more instances. Conversely, those that appear in just one ADL pattern (i.e., MakeCoffee, MakeChocolate and WashHands) typically have just a few instances. Using the real-time activity recognition system, we played back these synthetic ADL data and present the results in the next section.

ADL Name	**Description of Dataset**		
	% Standalone ADLs	***% In –pattern ADLs***	***Total instances***
MakeTea	10%	90%	41
MakeCoffee	100%	0%	4
MakeChocolate	100%	0%	3
MakePasta	10%	90%	29
BrushTeeth	30%	70%	19
HaveBath	10%	90%	28
WashHands	100%	0%	6
WatchTelevision	10%	90%	24
Num. Of ADLs			**154**

Table 1: Summary of synthetic ADL datasets

5.2.3 Experiment results

In order to evaluate the performance and therefore the feasibility of the approach, we used the metric accuracy. Accuracy measures the correctness of the algorithm, i.e. the ability of the algorithm to return correct results.We compute the accuracy of the recognition performance and provide the results in Table 2. The accuracy is computed from the values of true positive (tp), false positive (fp), true negative (tn), and false negative (fn) using the formula:

$$accuracy=(tp+tn)/(tp+fp+tn+fn) \quad (11)$$

We report results for three experiments and the first experiment was to evaluate recognition performance for static time windows, i.e., given that time windows cannot be shrunk or expanded. The results are shown in Table 2. The second experiment evaluated recognition performance when **shrink-only** is enabled. The results are presented in Table 3. Finally, the third experiment evaluated the performance given that **shrink-and-expand** is selected. The results are shown in Table 4. The second and third experiments relate to dynamically manipulated time windows.

ADL	Values from Dataset				
	TP	FP	TN	FN	Accuracy
MakeTea	39	0	0	2	**0.951**
MakeCoffee	4	0	0	0	**1.000**
MakeChocolate	3	0	0	0	**1.000**
MakePasta	28	0	0	1	**0.966**
BrushTeeth	14	0	0	5	**0.737**
HaveBath	19	0	0	9	**0.679**
WashHands	6	0	0	0	**1.000**
WatchTelevision	10	0	0	14	**0.417**
Average accuracy					0.844

Table 2: Recognition accuracy without shrinking or expansion

ADL	Values from Dataset				
	TP	FP	TN	FN	Accuracy
MakeTea	39	0	0	2	**0.951**
MakeCoffee	4	0	0	0	**1.000**
MakeChocolate	3	0	0	0	**1.000**
MakePasta	26	0	0	3	**0.897**
BrushTeeth	17	0	0	2	**0.895**
HaveBath	24	0	0	4	**0.857**
WashHands	6	0	0	0	**1.000**
WatchTelevision	18	0	0	6	**0.750**
Average accuracy					0.919

Table 3: Recognition accuracy with only shrinking enabled

ADL	Values from Dataset				
	TP	FP	TN	FN	Accuracy
MakeTea	39	0	0	2	**0.951**
MakeCoffee	4	0	0	0	**1.000**
MakeChocolate	3	0	0	0	**1.000**
MakePasta	21	0	0	8	**0.724**
BrushTeeth	17	0	0	2	**0.895**
HaveBath	24	0	0	4	**0.857**
WashHands	6	0	0	0	**1.000**
WatchTelevision	16	0	0	8	**0.667**
Average accuracy					0.887

Table 4: Recognition accuracy with both shrinking and expansion enabled

5.3 Discussion

As can be seen in Table 2, the recognition accuracy of MakeTea, MakePasta, MakeCoffee, MakeChocolate and WashHands is quite encouraging and attests to the feasibility of the presented approach. However, it is important to note that the recognition accuracy for BrushTeeth, HaveBath and WatchTelevision is low compared to the other ADLs. This can be attributed to the fact that these ADLs occur in very few standalone patterns; instead they mostly appear in sequential patterns. Given that the time window does not dynamically vary once created, sensor data belonging to more than one ADL in the pattern may be merged within a time window, thus leading to poor recognition accuracy.

Results in Table 4 show that there is a significant improvement on overall recognition accuracy. However, compared to Table 2, there is a reduction in the accuracy for MakePasta and a corresponding increase for BrushTeeth, HaveBath and WatchTelevision. Similarly, results in Table 3 indicate that the overall recognition accuracy was highest compared to the Table 2 and 4. Just like in Table 4, the

recognition accuracy of MakePasta reduced while that of BrushTeeth, HaveBath and WatchTelevision increased. However, despite the reduced recognition accuracy observed for MakePasta in both in Table 3 and Table 4, there is an overall improved average accuracy. The direct comparison of recognition accuracy per ADL is shown in Fig. 9. In addition, Fig.10 shows a direct comparison of average recognition accuracy.

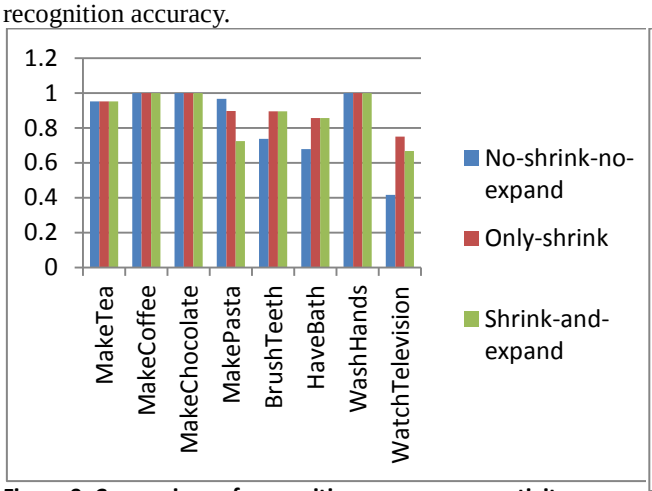


Figure 9: Comparison of recognition accuracy per activity

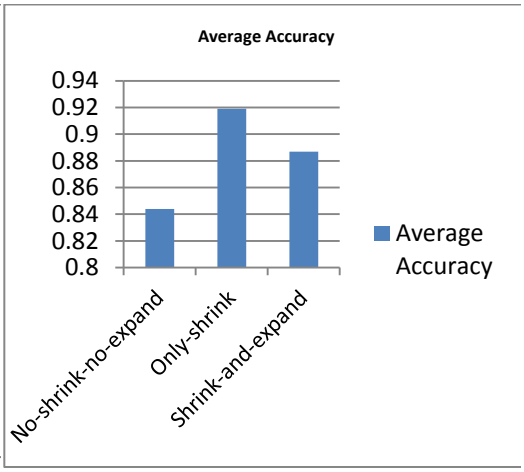


Figure 10: Comparison of average recognition accuracy

In all experiments, the activity recognition system recognized all the instances of all standalone ADLs. The reduced recognition accuracy was only observed regarding the in-pattern instances. As a result, we have reason to believe that the performance of activity recognition regarding BrusthTeeth, HaveBath and WatchTelevision may have been affected by the fact that they occur with other ADLs as indicated in Table 1, and more so as subsequent ADLs in the ADL patterns. In addition, the transition times from one ADL to another in an ADL pattern could also have made it possible for sensor data belonging to two distinct ADLs to be aggregated into one description, resulting in non-recognition. Another reason is the fact that several variants of ADLs were generated in the dataset. Whenever shrinking was allowing, a window could be shrunk before all the sensor activations associated with an ADL are obtained, hence the activations that arrive later could be merged with subsequent activations thus causing recognition failures. In the no-shrink-no-expand case, the recognition failure could be attributed to the chosen time window sizes.

We believe that by handling transitions between adjacent activities the recognition accuracy can be improved. This explains better recognition accuracy when shrinking and expansion are allowed. However, to minimize the failures whenever shrinking and expansion are supported, we believe that explicit relationships between ADLs in a pattern should be defined. This should involve a characterization of additional temporal relationships for the ADLs that could occur in patterns.

An interesting finding from the comparison of experiment results in Fig. 10 is that shrink-only had the best recognition accuracy. The shrink-and-expand case is the next best performer. We attribute lower recognition accuracy of shrink-and-expand compared to shrink-only to the fact that the maximum duration was used to derive time window lengths, thereby favouring activities with longer duration at the expense of those with shorter durations. However, we believe that by incorporating information about how inhabitants perform activities that can be captured through continuous use will improve the recognition accuracy of both shrink-only and shrink-and-expand mode. In future we plan to integrate the activity learning and model evolution approaches described in [41] so that feedback from how an inhabitant performs tasks is used by the recognition system for adaptation.

The average accuracy for all the experiments is above 83% using the provided dataset as well as with other datasets that we generated. This demonstrates that the approach is feasible in supporting real-time activity recognition.

6 CONCLUSION AND FUTURE WORK

This paper presented an approach based on dynamically varied time windows to support sensor data segmentation for use in continuous, real-time activity recognition. It characterises activity recognition and sensor data segmentation from which it formally defines a time window based segmentation model. The paper has detailed the rationale and operation algorithms of the model in the context of knowledge-driven activity recognition. In addition, different scenarios regarding dynamic manipulation of time windows were discussed. The model allows rich temporal information associated with sensor data and activities to

be exploited in real-time activity recognition. The implementation of a prototype to evaluate the approach was also described. The prototype consists of a synthetic ADL data generator, ADL ontology, sensor data simulator for ADL data playback, and a real-time activity recognition system. To establish the feasibility of this approach, this paper has presented evaluation results from experiments. Accuracy was chosen as an evaluation metric and the resulting average accuracy has demonstrated the feasibility of the approach. The average recognition accuracy was lowest, at 84%, when **no-shrink-no-expand mode** was activated. It was highest, at over 91%, when **shrink-only** mode was enabled. This proved that it is beneficial to dynamically manipulate time windows at runtime. In future, we plan to investigate richer models for capturing temporal information to address more complex temporal relationships among sensor data and activities. This would address problems with low recognition accuracy noted especially during the recognition of activities that occur in patterns. In addition, we plan to investigate the use of feedback or user profiles to aid the dynamic manipulation so as to improve performance.

7 REFERENCES

[1] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in Second International Conference, PERVASIVE 2004. Lecture Notes in Comput. Sci. Vol.3001. pp. 1-17.

[2] O. Brdiczka, J. L. Crowley and P. Reignier, "Learning situation models in a smart home," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 39, pp. 56-63, 2009.

[3] D. Sanchez, M. Tentori and J. Favela, "Activity recognition for the smart hospital," IEEE Intelligent Systems, vol. 23, pp. 50-7, 03, 2008.

[4] L. Chen, C. Nugent and H. Wang, "A Knowledge-Driven Approach to Activity Recognition in Smart Homes," Knowledge and Data Engineering, IEEE Transactions on, vol. PP, pp. 1-14, 2011.

[5] S. Knox, L. Coyle and S. Dobson, "Using ontologies in case-based activity recognition," in 23rd International Florida Artificial Intelligence Research Society Conference, FLAIRS-23, may 19, 2010 - may 21, 2010, pp. 336-341.

[6] H. Storf, M. Becker and M. Riedl, "Rule-based activity recognition framework: Challenges, technique and learning," in 2009 3rd International Conference on Pervasive Computing Technologies for Healthcare - Pervasive Health 2009, PCTHealth 2009, April 1, 2009 - April 3, 2009, .

[7] D. J. Patterson, D. Fox, H. Kautz and M. Philipose, "Fine-grained activity recognition by aggregating abstract object usage," in Ninth IEEE International Symposium on Wearable Computers, 2005, pp. 44-51.

[8] 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," Transactions of the Information Processing Society of Japan, vol. 48, pp. 2823-34, 08, 2007.

[9] U. Akdemir, P. Turaga and R. Chellappa, "An ontology based approach for activity recognition from video," in 16th ACM International Conference on Multimedia, MM '08, October 26, 2008 - October 31, 2008, pp. 709-712.

[10] P. Turaga, R. Chellappa, V. S. Subrahmanian and O. Udrea, "Machine recognition of human activities: A survey," IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, pp. 1473-1488, 2008.

[11] Y. Jie, C. Jian and L. Hanqing, "Human activity recognition based on the blob features," in 2009 IEEE International Conference on Multimedia and Expo, ICME 2009, June 28, 2009 - July 3, 2009, pp. 358-361.

[12] T. Huynh, U. Blanke and B. Schiele, "Scalable recognition of daily activities with wearable sensors," in Third International Symposium, LoCA 2007, 2007, pp. 50-67.

[13] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King and R. Friedman, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in Eleventh IEEE International Symposium on Wearable Computers, 2007, pp. 37-40.

[14] T. Huynh and B. Schiele, "Unsupervised discovery of structure in activity data using multiple eigenspaces," in 2nd International Workshop on Location- and Context- Awareness (LoCA 2006), 2006, pp. 151-67.

[15] J. Liao, Y. Bi and C. Nugent, "Evidence fusion for activity recognition using the dempster-shafer theory of evidence," in 9th International Conference on Information Technology and Applications in Biomedicine, ITAB 2009, November 4, 2009 - November 7, 2009.

[16] B. Bouchard, S. Giroux and A. Bouzouane, "A smart home agent for plan recognition of cognitively-impaired patients," Journal of Computers, vol. 1, pp. 10, 08, 2006.

[17] S. Chua, S. Marsland and H. W. Guesgen, "Spatio-temporal and context reasoning in smart homes," in International Conference on Spatial Information Theory, 2009, pp. 9-20.

[18] D. Riboni, L. Pareschi, L. Radaelli and C. Bettini, "Is ontology-based activity recognition really effective?" in Proceedings of CoMoRea'11, 8th IEEE Workshop on Context Modeling and Reasoning, 2011, .
[19] T. R. Gruber, "A translation approach to portable ontology specifications," Knowledge Acquisition, vol. 5, pp. 199-220, 06, 1993.
[20] L. Chen and C. Nugent, "Ontology-based activity recognition in intelligent pervasive environments," International Journal of Web Information Systems, vol. 5, pp. 410-30, 2009.
[21] D. Lymberopoulos, A. Bamis, T. Teixeira and A. Savvides, "BehaviorScope: Real-time remote human monitoring using sensor networks," in 2008 International Conference on Information Processing in Sensor Networks (IPSN 2008), 2008, pp. 533-4.
[22] L.Chen and I.Khalil, "Activity recognition: Approaches, practices and trends," in Activity Recognition in Pervasive Intelligent Environments, 4th ed., L.Chen, C.D.Nugent, J.Biswas and J.Hoey, Eds. Amsterdam-Paris: Atlantis Press, 2011, pp. 1-2-31.
[23] R. Bodor, B. Jackson and N. Papanikolopoulos, "Vision-based human tracking and activity recognition," in Proceedings of 11th Mediterranean Conference on Control and Automation (MED2003), 2003, pp. 6.
[24] L. Fiore, D. Fehr, R. Bodor, A. Drenner, G. Somasundaram and N. Papanikolopoulos, "Multi-camera human activity monitoring," Journal of Intelligent and Robotic Systems: Theory and Applications, vol. 52, pp. 5-43, 2008.
[25] T. L. M. van Kasteren, G. Englebienne and B. J. A. Kröse, "Towards a consistent methodology for evaluating activity recognition model performance," in Pervasive Computing 2010 Workshop on how to do Good Research in Activity Recognition. 2010.
[26] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola and I. Korhonen, "Activity classification using realistic data from wearable sensors," IEEE Transactions on Information Technology in Biomedicine, vol. 10, pp. 119-28, 01, 2006.
[27] Seon-Woo Lee and K. Mase, "Activity and location recognition using wearable sensors," IEEE Pervasive Computing, vol. 1, pp. 24-32, 07, 2002.
[28] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz and D. Hahnel, "Inferring activities from interactions with objects," IEEE Pervasive Computing, vol. 3, pp. 50-57, 2004.
[29] E. M. Tapia, S. S. Intille and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in Proceedings, 2004, pp. 158-75.
[30] L. Liao, D. Fox and H. Kautz, "Extracting places and activities from GPS traces using hierarchical conditional random fields," Int. J. Robotics Res., vol. 26, pp. 119-134, 2007.
[31] L. Chen, C. Nugent, M. Mulvenna, D. Finlay, X. Hong and M. Poland, "Using event calculus for behaviour reasoning and assistance in a smart home," in 6th International Conference, ICOST 2008, 2008, pp. 81-9.
[32] J. C. Augusto and C. D. Nugent, "The use of temporal reasoning and management of complex events in smart homes," in European Conference on Artificial Intelligence, 2004, pp. 778.
[33] I. Horrocks, "OWL: A description logic based ontology language," in 11th International Conference on Principles and Practice of Constraint Programming - CP 2005, October 1, 2005 - October 5, 2005, pp. 5-8.
[34] I. Horrocks, U. Sattler and S. Tobies, "Practical reasoning for expressive description logics," in 6th International Conference, LPAR'99, 1999, pp. 161-80.
[35] T. Van Kasteren, A. Noulas, G. Englebienne and B. Krose, "Accurate activity recognition in a home setting," in 10th International Conference on Ubiquitous Computing, UbiComp 2008, September 21, 2008 - September 24, 2008, pp. 1-9.
[36] X. Hong and C. D. Nugent, "Partitioning time series sensor data for activity recognition," in 9th International Conference on Information Technology and Applications in Biomedicine, ITAB 2009, November 4, 2009 - November 7, 2009.
[37] J. Ortiz Laguna, A. G. Olaya and D. Borrajo, "A dynamic sliding window approach for activity recognition," in 19th International Conference on User Modeling, Adaptation and Personalization, UMAP 2011, July 11, 2011 - July 15, 2011, pp. 219-230.
[38] X. Hong, C. Nugent, M. Mulvenna, S. McClean, B. Scotney and S. Devlin, "Evidential fusion of sensor data for activity recognition in smart homes," Pervasive and Mobile Computing, vol. 5, pp. 236-252, 2009.
[39] Stanford Center for Biomedical Informatics Research, "The Protege Ontology Editor and Knowledge Acquisition System", 2011. *Available from*: http://protege.stanford.edu/.

[40] Clark and Parsia, "Pellet: OWL 2 Reasoner for Java", 2011. *Available from*: http://clarkparsia.com/pellet/.
[41] G. Okeyo, L. Chen, H. Wang and R. Sterritt, "Ontology-enabled activity learning and model evolution in smart homes," in The 7th International Conference on Ubiquitous Intelligence and Computing, Xi'an, China, 2010.