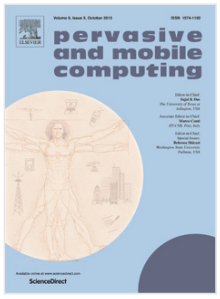
Accepted Manuscript



KCAR: A knowledge-driven approach for concurrent activity recognition

Juan Ye, Graeme Stevenson, Simon Dobson

PII: S1574-1192(14)00029-7
DOI: http://dx.doi.org/10.1016/j.pmcj.2014.02.003
Reference: PMCJ 485

To appear in: *Pervasive and Mobile Computing*

Received date: 1 August 2013
Revised date: 7 February 2014
Accepted date: 14 February 2014

Please cite this article as: J. Ye, G. Stevenson, S. Dobson, KCAR: A knowledge-driven approach for concurrent activity recognition, *Pervasive and Mobile Computing* (2014), http://dx.doi.org/10.1016/j.pmcj.2014.02.003

KCAR: A knowledge-driven approach for concurrent activity recognition

Juan Ye, Graeme Stevenson, and Simon Dobson

School of Computer Science, University of St Andrews, UK
E-mail: juan.ye@st-andrews.ac.uk

Abstract

Recognising human activities from sensors embedded in an environment or worn on bodies is an important and challenging research topic in pervasive computing. Existing work on activity recognition is mainly concerned with identifying single user sequential activities from well-scripted or pre-segmented sequences of sensor events. However a real-world environment often contains multiple users, with each performing activities simultaneously, in their own way and with no explicit instructions to follow. Recognising multi-user concurrent activities is challenging, but essential for designing applications for real environments. This paper presents a novel Knowledge-driven approach for Concurrent Activity Recognition (*KCAR*). Within KCAR, we explore the semantics underlying each sensor event and use semantic dissimilarity to segment a continuous sensor sequence into fragments, each of which corresponds to one ongoing activity. We exploit the *Pyramid Match Kernel*, with a strength in approximate matching on hierarchical concepts, to recognise activities of varying grained constraints from a potentially noisy sensor sequence. We conduct an empirical evaluation on a large-scale real-world data set that is collected over one year and consists of 2.8 millions of sensor events. Our results demonstrate that KCAR achieves an average recognition accuracy of 91%.

Keywords: Ontologies, Smart home, Concurrent activity recognition, Semantics, Domain knowledge, Pyramid match kernel

1. Introduction

Recognising human activities is an important task in a pervasive environment. It is particularly beneficial for application areas such as healthcare, home care for the elderly and disabled, emergency detection, and leisure. Earlier work mainly focuses on identifying activities for a single user [48], however there are often multiple residents living in the same environment, performing different tasks simultaneously. Having the ability to distinguish these *concurrent* activities is an essential step towards realising activity-aware applications for real-world settings [12].

Recognising multi-user concurrent activities is challenging due to many factors, among which the most critical are the interwoven sensor sequences and the myriad ways that activities may be simultaneously conducted. Sophisticated data-driven techniques have been designed to capture concurrent activity patterns [22], and they usually rely on heavy computation based around large volumes of training data, rendering them unsuitable for real-time recognition. The different ways in which concurrent activities may overlap lead to many possible patterns of sensor data, and the greater the number of permutations, the less feasible it becomes to collect sufficient training data. As a result, learned models may suffer from over-fitting, making them less effective in recognising activities. Few knowledge-driven techniques have attempted [38], however the detailed specifications might only be specific to certain environments or users, thus not scalable to a wide range of deployment.

To address the above problems, we propose a general knowledge-driven approach to support online multi-user concurrent activity recognition, called *KCAR*. We novelly consider one of the most important and pre-requisite processes toward recognising concurrent activities to be the ability to segment a continuous sensor sequence into fragments, each of which corresponds to a single ongoing activity. Then we turn the multi-user concurrent activity recognition problem into single user sequential activity recognition.

KCAR is designed to recognise activities conducted by multiple users simultaneously in a smart home environment, which benefits from the integration of an existing commonly-agreed knowledge base and of the statistical techniques in similarity measure and approximate matching. More specifically, the contributions of KCAR are listed as follows:

- KCAR is built on a top-level ontology model that explores the universal hierarchical structure in all dimensions of information [49], forming the theoretical foundation of our approach. Rather than devise an *ad hoc* model, we construct domain ontologies from an existing large knowledge base (e.g., WordNet [30]), which facilitates reuse and share of the ontologies across different platforms and environments. With the help of this knowledge, KCAR does not need to learn correlations between sensor events from training data.

- KCAR enables online segmentation on continuous sensor sequences by evaluating the *semantic similarity* between individual sensor events. We follow the methodology of the *semantic sensor web* [39] to extract the semantic features of a sensor event into *spatial*, *temporal*, and *thematic* aspects. We employ a hierarchy-based similarity measure [45] to quantify the similarity on these aspects in their corresponding ontologies.

- KCAR recognises activities by matching segmented sensor sequences to ontological activity profiles. One of the main concerns in concurrent activity recognition is how to distinguish an *unexpected* sensor event as either a sensor noise or another concurrent activity. To tackle this issue, we employ and extend the recently devised image matching technique – Pyramid Match Kernel (PMK) [11] – to accommodate and balance the sensor noise in activity recognition.

- We evaluate the performance of KCAR's segmentation and recognition algorithms on a large-scale real-world smart home data set. The results demonstrate that (1) compared to classic segmentation techniques not only can KCAR segment the sensor sequence more accurately, but it can also produce segments that more closely resemble activities being performed in a real-world environment, and (2) KCAR can recognise concurrent activity recognition to a high accuracy.

The rest of the paper is organised as follows. Section 2 introduces the background and existing work in activity recognition and identifies the scope of this paper. Section 3 describes the modelling of domain ontologies in a smart home environment, based on which we discuss semantic similarity between sensor events and perform segmentation on interwoven sensor sequences in Section 4. To perform concurrent activity recognition, we introduce the principle of the PMK and demonstrate how we extend and apply it in Section 5. The performance of segmentation and activity recognition is evaluated on a large-scale real-world data set, which is detailed and discussed in Section 6. The paper concludes in Section 7.

2. Background and Literature Review

A smart home environment can be defined as a regular dwelling that is configured with a number of sensors and actuators. The sensors perceive the state of the residents and their ambient environment, based on which the actuators infer the residents' present activities and as well as provide activity-aware services [9]. Examples of the prototyped smart homes include the Georgia Tech Aware Home project [21], MIT PlaceLab [27], University of Amsterdam smart houses [20], and the CASAS project [6], to name a few. Activity recognition is a key element in realising the smart home vision, which links the gap between high-level applications and low-level sensor data. A general process of activity recognition is composed of collecting sensor data, segmenting the collected sensor data into fragments, and recognising the activity from each fragment. In the following, we detail the process and identify the scope of this paper in terms of what types of sensor data are usually collected in a smart home environment, what types of activities are of interest, how the data is collected, why we need to segment sensor data and how, and what the existing techniques for recognising activities are.

2.1. Sensor Data

Pervasive sensing technologies have progressed significantly towards the design and development of small, lightweight, low cost sensors with long battery life. There are five types of sensors that are popular in existing smart home environments [7, 48]: *cameras* to closely watch the activities in the environment, *accelerometers* that are worn on a human body to detect movement and postures, *environment and resource sensors* to monitor the environmental property (e.g., temperature or humidity) and resource (e.g., electricity or gas) consumption, *positioning sensors* that are installed in rooms to detect the inhabitant's whereabout, and *object sensors* that are attached to an everyday object to detect interactions between the object and the inhabitant.

Early activity recognition approaches relied on processing features extracted from camera video streams [2, 40]. Such solutions are challenged because of the potential for the violation of user privacy, the difficulty of extracting robust and informative features to infer high-level activities, and the computational overhead [42, 44]. Accelerometers, worn by users on their thighs or arms, have proved useful in detecting running, walking, or jogging [28, 50]. It is usually necessary to combine this information with other sensors to accurately recognise higher-level activities such as cooking or exercising. Environmental and resource sensors are often considered as a contextual factor that affects users' activities or an outcome of their activities, rather than as a direct evidence on inference.

Positioning sensors like passive infrared sensors [20] or motion detectors [7, 27] are widely used in smart homes. Location has significant importance in identifying human daily activities in a smart home [27, 47]; e.g., cooking in the kitchen, sleeping in the bedroom, or taking shower in the bathroom. To distinguish activities occurring in the *same* location, we need to use other types of sensors. Object sensors detect user interactions with everyday objects, which include RFIDs, pressure mats, binary-state sensors developed in the University of Amsterdam [20], binary switches (called *MITes*) in MIT [27], and binary discrete motion sensors used in CASAS [7]. Compared with the other types of sensors, object sensors exhibit the advantages of unobtrusiveness, ease of processing, potential to be deployed on a wide range of objects, and direct implication of a user's current activities. KCAR attempts activity recognition mainly based on these *positioning and object* sensors.

2.2. Activities of Interest

Varying with different application goals, a smart home environment can have a wide range of activities of potential interest. For example, an elderly healthcare application may be concerned about users' movements or fall detection, while a general purpose smart home application may be interested in daily routines such as cooking, personal hygiene, working, or entertaining. According to the interactions between the performance of these activities, they can be classified into four categories [14]:

- *single user sequential* activities – a single user performs activities one by one;
- *multi-user sequential and simultaneous* activities – multiple users perform the same activity together, e.g., two users are drinking coffee together;
- *multi-user collaborative* activities – multiple users perform different activities in a cooperative manner to achieve the same goal; e.g., two users are cooking together: one is retrieving the ingredients while the other is stirring the pan;
- *multi-user concurrent* activities – multiple users perform different activities independently and aim for different goals; e.g., one user is cooking while the other is working on the computer.

KCAR is founded on semantics underlying the sensor events and activities, and hence it aims to recognise the fourth class of activities – *detecting coarse-grained concurrent activities* that have explicit and distinguishable semantic profiles; for examples, activities being performed in different locations; or activities involving everyday objects in completely different categories such as `Stove` in the `Cook` activity, and `Computer` in the `Work` activity.

2.3. Data Collection

Most activity recognition techniques require a data set to build an activity model and evaluate the recognition algorithm. A data set usually consists of a temporally ordered sequence of sensor events that are annotated with activity labels. An example of a data set is shown in Listing 1[1].

Listing 1 An example of a data set that consists of timestamped sensor events with annotated activities [5]

```
        Timestamp              Sensor          Value      Annotated Activity
-------------------------------------------------------------------------------------
2009-08-24 20:23:43.054643 Kitchen_Stove         ON       R2_Prepare_Dinner begin
2009-08-24 20:23:56.086543 WorkArea1_Computer    ON       R1_Work_In_Room begin
2009-08-24 20:23:57.002937 Kitchen_Fridge        ON
2009-08-24 20:24:19.034964 WorkArea1             ON
2009-08-24 20:24:21.061429 WorkArea1             ON
2009-08-24 20:24:22.078563 Kitchen_Door          ON
2009-08-24 20:24:23.077674 WorkArea1             ON
2009-08-24 20:24:25.095411 BedArea1              ON
2009-08-24 20:24:26.077674 WorkArea1             ON       R1_Work_In_Room end
```

Early approaches to collecting such data sets relied on asking subjects to perform a set of scripted activities, one at a time, in a laboratory setting [23]. This usually produces a clean data set with well segmented sensor sequences and highly predictable activity patterns. In recent years, researchers are more interested in collecting data in a more realistic manner; that is, they instrument many sensors of various types in a normal house where subjects will live in for a long period of time and perform activities as normally as possible. The subjects' activities can be manually recorded by hand [46], reported using a microphone [20], or recorded on video [27]. An offline annotation process will be taken to mark the sensor data with the recorded activities. In these cases, data is often noisy due to the environmental interference, dislodge of the sensors, activities being performed in a more diverse manner, or errors in recording the performed activities, and consequently segmenting sensor sequences and recognising activities on these data are more challenging. We propose KCAR as a technique to perform activity recognition on unsegmented sensor sequences that are collected in this type of environments.

2.4. Segmentation

Segmenting a continuous sensor sequence is usually a prerequisite process for activity recognition [33]. The most common approach is to partition a sensor sequence into a fixed time interval; e.g., one second [12] or one minute [43]. Each of the resulting partitions shares the same time interval, known as the static sliding window technique [1, 18]. Another type of static sliding window techniques is the fixed-size approach, where each window has the same number of sensor events [23]. We refer to the former as *fixed time* segmentation and the latter as *fixed size* segmentation in this paper and we will compare the performance of KCAR with these two techniques in Section 6.

The above static sliding window techniques are impractical in real-world scenarios as the duration of activities typically varies anywhere from 30 seconds to over 10 hours. Such segmentation often gathers sensor events that are spread over the boundary of two activities into one segment, which adds extra noise to the inference process. Additionally, for a long-duration activity like cooking, sensor traces within a short time window are often insufficient to arrive at an accurate classification.

In contrast, dynamic sliding window methods enable varying sizes of sliding windows at runtime based on different features, such as the duration of activities [29, 33, 41], change of sensor states [24], or change of location context of consecutive sensor data [16]. Krishnan et al. [23] explore both static and dynamic sliding window approaches, with the incorporation of the time decay and mutual information of sensor events within a window; e.g., the occurrence ratio of two sensors occurring consecutively in the entire sensor

[1]For the sake of readability, we replace the sensor IDs with an object or location symbol indicating to which object the sensor is attached, or where it is this sensor is installed.

stream. Rashidi et al. [35] extend the tilted-time window approach to discover activity sequential patterns over time; that is, using temporally parameterised support counts to find frequent patterns over streaming sensor data. All these techniques work well on single user sequential activity data sets and none of them yet targets at segmenting concurrent activity data sets, which KCAR aims to address.

Gu et al. [12] present an unsupervised technique based on *emerging patterns* with sliding time windows to recognise interleaved activities. This technique calculates complex activity scores based on mined activity-feature sets as well as correlation scores between the activities. The main difference between KCAR and the emerging patterns in segmentation is that instead of relying on any training data we use the ontological reasoning to evaluate the semantic similarity between consecutive sensor events.

2.5. Activity Recognition

Activity recognition techniques can be grouped into data- and knowledge-driven approaches [3, 48]. Data-driven approaches are built on machine learning and data mining techniques, which learn an activity model and estimate parameters from a set of training data. Among all the existing techniques, Hidden Markov Models (HMM) have not only demonstrated promising accuracies in recognising single user sequential activities [20] but also are presented as one of the most popular techniques in recognising interleaved and concurrent activities.

Patterson et al. [34] employ the HMM model to recognise interleaved activities of a morning routine using RFID data. Having experimented with various HMM models, they conclude that using a HMM with a single state for each activity performs best and increasing model complexity does not necessarily improve the recognition accuracy. Modayil et al. [31] propose an *interleaved HMM* model that aims to capture inter- and intra-activity dynamics. It recognises the interleaved activities that are performed by a single user. It is evaluated on a data set that is collected in a laboratory performed by subjects on scripted activities. The results show that it is an efficient mechanism to recognise potentially confusable activities like *make oatmeal* and *make eggs*.

Gong et al. [10] develope a dynamically multi-linked HMM model to interpret group activities from video data involving multiple objects in a noisy outdoor scene. The model is based on the discovery of salient dynamic interlinks among multiple events using dynamic probabilistic networks. Nguyen et al. [32] employ the hierarchical HMM (HHMM) in a general framework to recognise primitive and complex behaviours of multiple people. A unified graphical model is constructed to incorporate a set of HHMMs with data association.

Hu et al. [17] propose a novel probabilistic framework for multi-goal recognition where both concurrent and interleaving goals can be recognised. The technique used is the Skip-Chain Conditional Random Field (SCCRF), within which concurrent and interleaved goals are derived by adjusting inferred probabilities through a correlation graph. The SCCRF is computationally expensive when a large number of skip edges are involved. To prevent the recognition accuracy from deteriorating, every partial model of the interleaved activities has to be observed during the training phase. Hence the SCCRF requires a large amount of training data because there are many different ways to interrupt and resume an ongoing activity. Also as acknowledged in [22], both HMMs and CRFs are more suitable for purely sequential activities. To recognise interleaved and concurrent activities, these techniques need to be extended or integrated with other techniques, which usually encounters the problems of heavy computation and craving for training data.

Helaui et al. [15] build composite activity models using the Markov Logic Network, a statistical relational approach to incorporate common-sense background knowledge; that is, the sequential relationships between activities; e.g., the activity of setting the table must precede the activities of eating breakfast and clearing the table.

Different from the above data-driven approaches, KCAR is knowledge-driven in that we do not need any training data to build a potentially complex model and thus we avoid the computational complexity and the problems of over-fitting. Knowledge-driven techniques follow a description-based approach to model the relationships between sensor features and activities [3]. Ontological reasoning is a representative example as ontologies provide a formal way to represent sensor data, context, and situations into well-structured

terminologies, which makes them understandable, sharable, and reusable by both humans and machines [4, 13, 37]. In ontology-based approaches, activities are described with a number of properties linking to constraints on sensor events. The activity recognition process is to match the conditions of each activity against the input sensor sequence. If all the conditions are satisfied, the reasoner will infer the activity.

Chen et al. [4] present an ontological model to represent smart home activities and relevant context. The approach is motivated by the observations that activities are daily routines full of common-sense knowledge providing rich links between the environment, events, and activities. The domain and prior knowledge is valuable in creating activity models, avoiding the need of large-scale data set collection and training. More recently, Saguna et al. [38] combine ontological and spatio-temporal modelling and reasoning to recognise interleaved and concurrent activities. However their approach requires detailed understanding and thorough investigation of the activity-related specifics so as to successfully build long-term solutions. Although our approach shares the same motivation, we use the more certain must-have knowledge rather than provide a complete specification for each activity; thus, we can reduce the amount of knowledge engineering effort and as well as the bias from experts.

Moreover, a pure knowledge-based reasoning is insufficient to deal with sensor noise [48] in that if a sensor sequence contains contradictory sensor events, then the reasoner might arrive at conflicting inference results. To resolve this problem we employ the Pyramid Match Kernel (PMK) technique to enable approximate match between sensor sequences and activity profiles. This technique is rooted in image-based object detection, with a well proved advantage in supporting match between hierarchical concepts [25]. In KCAR we adapt the PMK algorithm to accommodate sensor noise that sparsely occurs in the collected data set.

2.6. Overview of KCAR

KCAR is able to recognise multi-user concurrent activities, as illustrated in Figure 1. KCAR takes as input a continuous sensor sequence that consists of raw sensor events and processes the input sequence by linking each sensor event with the concepts defined in our ontological models. The segmentation process groups or partitions sensor events into fragments based on the similarity measure that is calculated between the ontological concepts linked with these events.

A PMK-based recogniser performs the matching between each fragment and the ontological description of activities, and infers the activity that is most likely to be occurring. The inferred result is evaluated to determine whether the fragment for this activity is complete or not. If so, the recogniser outputs the result; otherwise, the fragment is returned to the segmentation algorithm to be concatenated with new sensor events; this process is performed recursively. The produced result is a sequence of temporally bound activities. We describe each part of the process in the following sections, starting with the ontological model that forms the foundation for the segmentation and recognition algorithms.

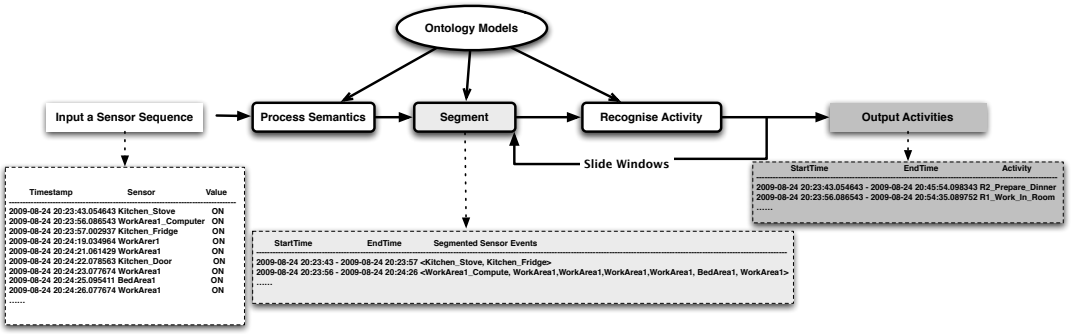


Figure 1: An overview of KCAR on segmentation and recognition

3. Ontological Modelling

Central to our technique is a generic ontological model that consists of three components: sensor, domain, and activity ontologies. This is illustrated in Figure 2. Built upon well-founded domain ontologies, we can evaluate the semantic similarity between sensor events in the sensor ontology and as well as relate the sensor and activity ontologies so that we can perform activity recognition.

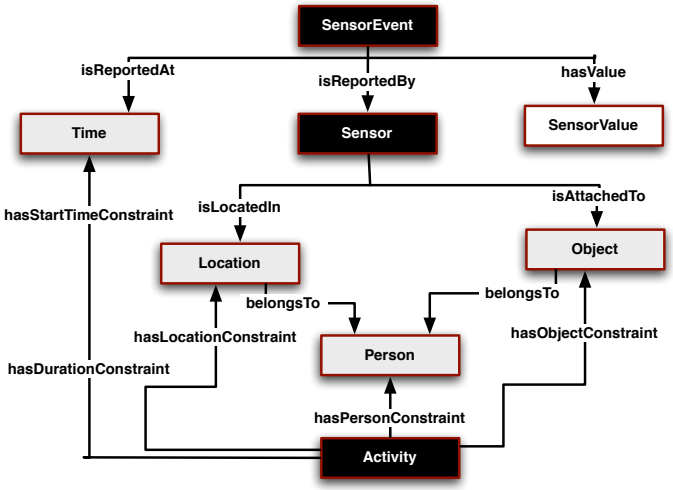


Figure 2: An overview of a general structure of the ontological model

To specify domain knowledge, we use OWL – Web Ontology Language – a Description Logic [52] based markup language, as it supplies decidability and computational completeness of reasoning procedures at the expense of some modelling expressiveness. The core elements in the formalism are concepts, instances, and relationships. A concept corresponds to a collection of instances and a relationship relates two instances. In Description Logic there are (1) the *TBox*, *Terminological Box*, that contains the vocabularies on defining concepts and relationships, which are reusable across different environments, and (2) the *ABox*, *Assertional Box*, that contains the vocabularies on defining instances and their relationships about a particular real-world domain.

3.1. Domain Ontology

Rather than provide a full formal conceptualisation, the domain ontologies are referred to as vocabularies, which classify and organise conceptual terms in a hierarchy. The object ontology (OO) describes the `type-of` relationships between household objects. For example, a `Fridge` is a type of `WhiteGood`, denoted as `Fridge` $\sqsubseteq$ `WhiteGood`, and represented in the following statement in the TBox of the OO.

Listing 2 An example of declaring household object concepts

```
# Declare household object concepts in the TBox of OO
obj:WhiteGood a owl:Class.
obj:Fridge a owl:Class;
    rdfs:subClassOf obj:WhiteGood.
```

The vocabulary in the OO is extracted from the WordNet [30], which is a hierarchical lexical system where words are organised by semantic relations in terms of their meaning. The use of the existing knowledge

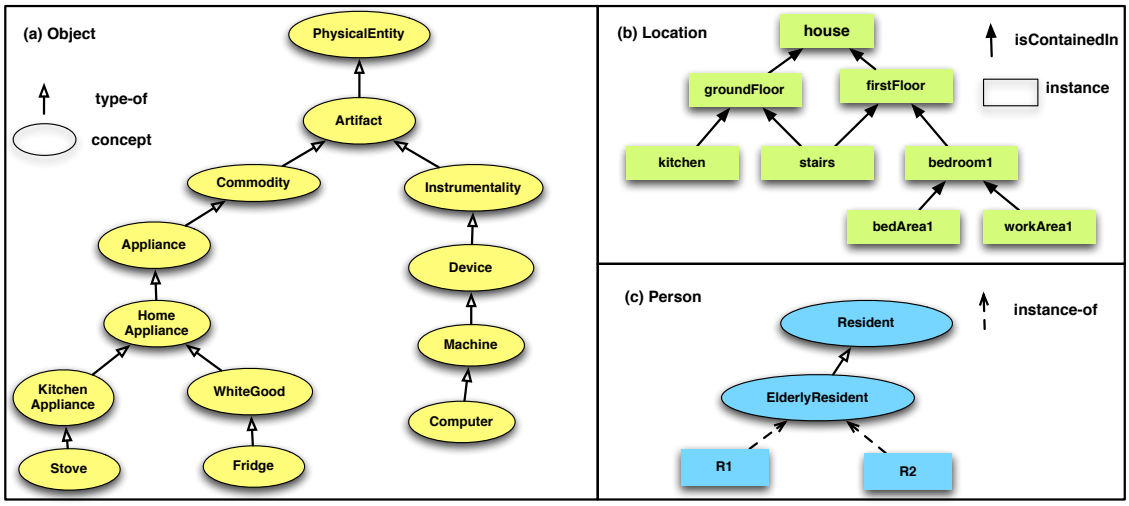


Figure 3: Examples of domain ontologies in Object, Location, and Person

repository can make the OO as general as possible, which encourages share and reuse of the ontologies. Figure 3 (a) shows a part of the OO[2].

The location ontology (LO) describes location concepts common in most home settings, which is complemented by defining location instances and `containment` relationships in terms of their spatial layout. Figure 3 (b) presents an example of room layouts in a particular house in the form of a lattice. A house consists of two floors, connected by a set of stairs, whose ontological statements are described in Listing 3.1.

Listing 3 An example of defining location concepts, instances, and relationships

```
# Declare location concepts in the TBox of LO
loc:House a owl:Class .
loc:Floor a owl:Class .
loc:Stairs a owl:Class .

#  Define the containment relation (to be used on instances) in the TBox of LO
loc:contains a owl:ObjectProperty .

# Define location instances in the ABox of LO
ex:house a loc:House .
ex:groundFloor a loc:Floor .
ex:firstFloor a loc:Floor .
ex:stairs a loc:Stairs .

# Define spatial relationships between location instances in the ABox of LO
ex:house loc:contains ex:groundFloor .
ex:house loc:contains ex:firstFloor .
ex:groundFloor loc:contains ex:stairs .
ex:firstFloor loc:contains ex:stairs .
```

We similarly construct the person ontology (PO) that classifies the residents in a house into groups; for example in Figure 3 (c): there are two elderly resident instances $R1$ and $R2$. Both the PO and OO focus on relations at the class level, while the interpretation of the LO is subject to relations between its

[2]For brevity, we omit intermediate terms.

individuals (the spatial layout of the environment). We manually create these relations by following the guideline of a top-level ontology [49]: gradually abstracting concepts from the ground values in each domain step by step. This allows us to build domain ontologies in a more systematic and tractable manner, which not only facilitates reasoning on relationships between concepts but also enables share and reuse with other ontologies across more environments [49].

3.2. Sensor Ontology

The sensor ontology (SO) describes sensors and sensor events. A sensor event is usually denoted as a tuple (t, s, v), indicating that at the time instant t a sensor s reports a value v, as shown in Listing 1. To explore the semantics of a sensor event, we borrow the semantic annotation terminology from the Semantic Sensor Web, that describes *temporal*, *spatial*, and *thematic* semantics [39]. The temporal dimension refers to the time instant t when the sensor event is reported or an interval during which the event is valid; the spatial dimension refers to the location where the sensor s is installed; and the thematic dimension refers to a real-world state extracted from the sensor event. As we are concerned with object sensors, the thematic dimension refers to objects to which a sensor is attached.

Definition 1 defines the semantics of an object sensor event as a 3-dimensional tuple, denoted as $se = (t, l, o)$, each of which refers to an instant in domain ontologies including the `Time`, `Location`, and `Object` ontologies.

Definition 1. Let TO, LO, and OO be the Time, Location, and Object ontologies that organise concepts of each domain in a hierarchy. The semantics of a sensor event is represented as $se = (t, l, o)$, where

- $\tau : se \rightarrow TO$, mapping the reported time of se to an instance t in `TO`;
- $\iota : se \rightarrow LO$, mapping the location of the sensor to an instance l in `LO`;
- $\delta : se \rightarrow OO$, mapping the object associated with the sensor to an instance o in `OO`.

Different from the Semantic Sensor Web community, the sensing semantics in a smart home environment, especially in a multi-user context, involve another important dimension – `Person`, indirectly indicating *who* a sensor event is about via the association of person semantics with the location or object of the sensor; that is, who owns or has the access right to this location or object. Person semantics are indirectly related to a sensor event and can be used to infer which particular person is performing an activity. For example, if we infer the current activity is sleeping and the detected location is a bedroom belonging to a resident $R1$, then we can further derive that it is `R1` who is sleeping.

3.3. Activity Ontology

The activity ontology (AO) represents the structural properties; that is, the hierarchy of activity types. We constrain activities by associating each with *time*, *object*, *location*, and *person* conditions. We constrain activity classes and instances by associating each with conditions on properties; for *object*, *location*, and *person*, activity classes restrict the type of values that each property can relate. The *object* and *location* condition specifies what object a person must access in order to perform this activity, and where this activity must occur. The use of a hierarchy allows us to specify conditions using *more general* concepts rather than listing each specific concept for each instance. For example, we can directly constrain the objects for the activity of preparing dinner on cooking utensil or tableware, without the need to enumerate every concrete entity like a cup, plate, or pan.

We note that what we specify against more general activity classes should be general knowledge in the sense that most people will execute the activity in this way. Against activity instances, the *person* constraint indicates the current activity profile is about a specific user; that is, one user performs this activity differently from another. General conditions are defined on concepts in the domain ontologies (e.g., sleeping takes place in a bedroom), while specific constraints can be placed on instances (e.g., `R1_Sleep` involves sleeping in `R1`'s bedroom). Listing 3.3 specifies this relationship.

Classes and instances may also be paired with *time* conditions: *occurring* time – when the activity usually occurs and *period* – how long the activity typically lasts. For example, we place the constraint that the activity of preparing dinner should start in the evening and last more than 5 minutes. Where this information is present in an instance definition, we treat it as overriding any temporal restrictions associated with the class.

Listing 4 An example of defining activity concepts and instances

```
# Define the general activity of sleeping in the TBox of A0
ex:Sleep rdfs:subClassOf act:Activity ;
   act:time [act:hasStartTimeConstraint [ ex:activityStartDescription
                                          a time:DateTimeDescription ;
                                          time:unitType time:unitHourMinute ;
                                          time:hour 22^^xsd:int] ;
             act:hasDurationConstraint "PT35M"^^xsd:duration ] ;
   rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty act:object ;
        owl:allValuesFrom obj:Bed
      ] ;
   rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty act:location ;
        owl:allValuesFrom loc:Bedroom
      ] ;
   rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty act:person ;
        owl:allValuesFrom per:Person
      ] .

# Define activity instances in the ABox of A0
ex:R1_Sleep a ex:Sleep;
    act:object ex:bed1 ;
    act:location ex:bedroom1 ;
    act:person ex:R1 .
```

4. Semantic Similarity and Segmentation

In this section, we use the generic ontological model introduced above to partition a continuous real-time sensor sequence into fragments, each of which corresponds to one (or part of an) activity that might be occurring simultaneously with other activities, as illustrated in Figure 4. The segmentation is based on sensor and activity semantics; that is, using semantic dissimilarity between sensor events to segment for each user, and using activity temporal knowledge to control the time interval of fragments. In the following we introduce how to quantify the similarity measures between sensor events.

4.1. Semantic Similarity

The main assumption of our approach is that there exists semantic similarity between sensor events, with which we can segment sensor sequences into semantically integral partitions. Deriving from the semantic interpretation of a sensor event, we present how to quantify the similarity between two sensor events from their ontological conceptual terms.

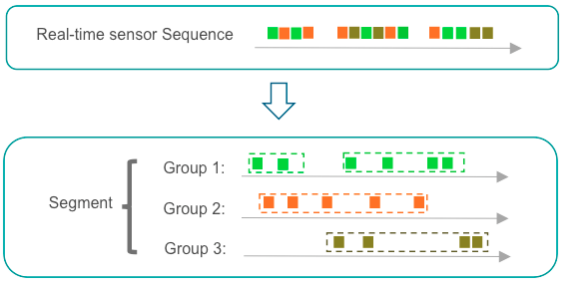


Figure 4: Online segmentation of a continuous sensor trace

Definition 2. Let se_i and se_j be semantic representations of two sensor events. The **semantic similarity** between them is defined as

$$sim(se_i, se_j) = (sim_T(se_i, se_j), sim_S(se_i, se_j)) \quad (1)$$

$$sim_S(se_i, se_j) = (sim_C(\iota(s_i), \iota(s_j)) + sim_C(\delta(s_i), \delta(s_j))/2 \quad (2)$$

where

- ι and δ are the location and object mapping functions introduced in Definition 1.
- sim_T is the time similarity function that compares the temporal similarity between the sensor events;
- sim_S is the sensor similarity function on Location and Object concepts that are associated with sensors;
- sim_C is the conceptual similarity function between hierarchical concepts in each domain.

The conceptual similarity function is built on the algorithm proposed by Wu et al. [45]. The approach works by finding the least common subsumer (LCS) of the two input concepts and computing the path length from the LCS up to the root node. The LCS is the most specific concept that both concepts share as an ancestor. The literature contains many other similarity measures; for example, the Leacock Chodorow matcher is based on counting the number of links between two concepts in the hierarchy [26]. The principle of this method is similar to what we use here; however, the similarity value in [45] is more intuitive to understand. Some works use the information content of the concepts by exploiting the frequency of concept occurrences in a given text corpus, such as the Resnik matcher [36] and the Jiang Conrath matchers [19]. Compared to them, [45] only needs a hierarchy of conceptual terms, which can be easily accessed through the standard lexical system like WordNet.

Definition 3. Let c_1 and c_2 be two concepts organised in one hierarchy. The **conceptual similarity** measure between them is calculated as (as shown in Figure 5):

$$sim_C(c_1, c_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3},$$

where N_1 (N_2) is the path length between c_1 (c_2) and the LCS node of c_1 and c_2, and N_3 is the path length between the LCS and the root.

When c_1 is equal to c_2, their LCS node is itself and the similarity is 1.0. When c_1 is semantically far from c_2, their LCS node can be close to the root in the hierarchy, which makes N_1 and N_2 are large and N_3 small, so the similarity is close to 0. Therefore, the larger the similarity measure, the closer the two concepts. Taking an example from Figure 3, the conceptual similarity between a `Stove` and a `Fridge` is $\frac{2*4}{2+2+2*4} = 0.67$, while the similarity between `Stove` and `Computer` is $\frac{2*1}{5+4+2*1} = 0.18$.

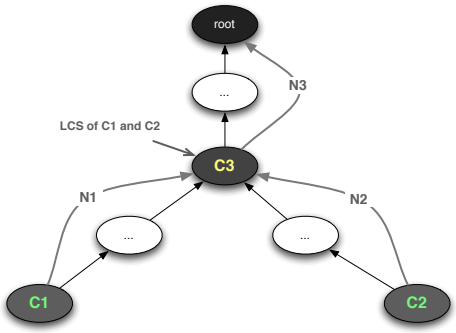


Figure 5: The concept similarity measure

The time similarity function sim_T can exist in two forms: numeric and symbolic. Since the timestamps on each sensor event can be represented in milliseconds, its numeric form is calculated as

$$sim_T(se_i, se_j) = max(0, 1 - \frac{|\tau(se_j) - \tau(se_i)|}{T_{max}}), \tag{3}$$

where T_{max} is the maximum range of the time under consideration. For example, if we consider daily activities we set the T_{max} to be 24 hours (equally 86,400 seconds). If there exists a hierarchy of temporal concepts similar to the object or location concepts, then its symbolic form is calculated as Definition 3. In this paper, we take the numeric form, because sensors in a smart home environment are usually frequently sampling and the symbolic form of temporal concepts is not particularly useful.

Example 1. In this example, we calculate the semantic similarity between the sensor events se_1, se_2 and se_3 in Figure 6, which are extracted from the Listing 1. The location, and object concepts refer to the LO and OO in Figure 3.

- se_1 and se_2: their time similarity is $1 - \frac{13}{86400} = 1$ using Equation 3. In terms of the sensor similarity, their location similarity between kitchen and work area 1 is 0, and object similarity between `Stove` and `Computer` is 0.18. Thus the sensor similarity is normalised to be 0.09 (=(0+0.18)/2) using Definition 2.
- se_2 and se_3: Similarly, their time similarity is $1 - \frac{1}{86400} = 1$ and their sensor similarity is 0.09.
- se_1 and se_3: their time similarity is $1 - \frac{14}{86400} = 1$ and their sensor similarity is $(1 + 0.67)/2 = 0.83$ where the location similarity is 1.0 and the similarity between `Stove` and `Fridge` is 0.67.

In the above example, based on the similarity measures between these three sensor events, we can intuitively derive that the sensor events se_2 should be partitioned from the events se_1 and se_3, for se_2 is more likely to correspond to an activity different from the activity corresponding to the latter two. For example, this sequence could suggest that there are two semantically different activities occurring at the same time: cooking in the kitchen, and using the computer in the bedroom.

This example introduces our idea of *automatic segmentation* of sensor sequences that record concurrent activities; that is, dividing a sequence of sensor events into segments, each of which is composed of sensor events that are semantically similar, intuitively corresponding to one of the ongoing activities. As a coarse-grained concurrent activity recognition approach, KCAR only separates the sensor events that cannot be grouped together. It cannot finely distinguish concurrent activities that are semantically similar. For example in Figure 6, it cannot detect how many users are involved in the cooking activity, or further discriminate which user is performing a different cooking task such as stirring a pan or retrieving an ingredient.

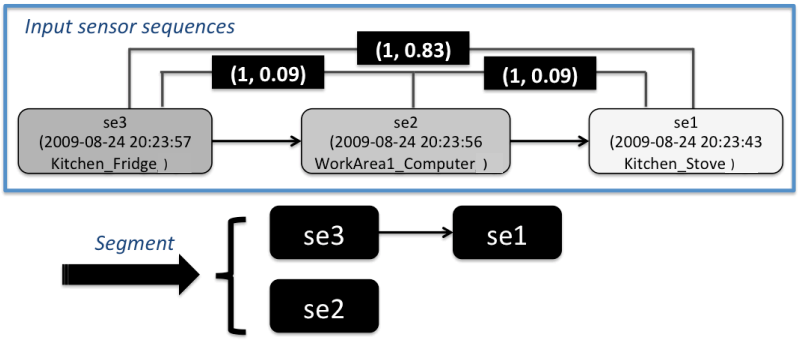


Figure 6: Segmenting a sequence of input sensor events based on their semantic similarity

4.2. Online Segmentation

Algorithm 1 presents the segmentation process. During the process, we maintain two lists: a *tentative* list L_t and a *confirmed* list L_c. L_t records a list of pairs (l, a), where l is an unfinished sequence that is possibly concatenated with new sensor events, and a is an activity that is inferred from l.

Algorithm 1: Automatic segmentation of real-time sensor sequences

Data: $SEQ = \langle se_1, se_2, \ldots, se_n \rangle$: a sequence of incoming sensor events
(θ_T, θ_S): the time and sensor similarity threshold pair
λ: the maximum time distance to separate any two adjacent sensor events
AR: an activity recognition reasoner that takes a sequence of sensor events and outputs an activity
Result: L_S: a list of segmented sequences
initialise(L_c);
initialise(L_t);
foreach $se \in SEQ$ **do**
  $found = false$;
  if $(L_t.isNotEmpty)$ **then**
    foreach $(l, a) \in L_t$ **do**
      $(sim_T, sim_S) = sim(se, l.last)$;
      if $dist(\tau(se) - \tau(l.last)) \geq \lambda$ **then**
        $L_c.add((l, a))$;
        $L_t.remove((l, a))$;
      if $similarityCheck(sim_T, sim_S, \theta_T, \theta_S)$ **then**
        $a' = AR(\langle l; se \rangle)$;
        if $timeSpan(\langle l; se \rangle) \leq retrieveDurationConstraint(a').upperBound$ **then**
          $L_t.update((\langle l; se \rangle, a'))$;
          $found = true$;
          $break$;
  if $!found$ **then**
    $a' = AR(\langle se \rangle)$;
    $L_t.add((\langle se \rangle, a'))$;
if $L_t.isNotEmpty$ **then**
  $L_c.addAll(L_t)$;
return L_c;

Given an incoming sensor event se, the algorithm calculates the similarity measure between it and the last sensor event in each sensor sequence l in L_t. If they pass the similarity check, then the event se is appended to the list l as $\langle l; se \rangle$. The algorithm performs activity recognition on the new list and updates the inferred activity a'. If the time span over the new list satisfies the upper bound of the period constraint

on a', then the new list is validated and updated to L_t. If the current event se cannot concatenate with any sequence in L_t, then a new list $\langle se \rangle$ is formed and an activity a' is informed from this singular list. The new pair $(\langle se \rangle, a')$ will be added to L_t. That is, each sequence in L_t suggests one of the concurrent activities and the number of sequences in L_t implies *how many* concurrent activities are ongoing.

To examine whether a pair (l, a) in L_t should be removed to the confirmed list, we consider the principle that inactive sensor events over a long period suggest discontinuity of activities. That is, if the time distance between the most recent event in l and the current event (i.e., $\tau(se)$) is greater than the maximum time gap, then l is regarded as *out of date* and cannot be joined with incoming sensor events.

In the end, the segmentation result is a m-sized list $L_c = \{(\langle se_1, se_2, \ldots, se_{k^1} \rangle, a_1), (\langle se_{k^1+1}, \ldots, se_{k^2} \rangle, a_2), \ldots, (\langle se_{k^{m-1}+1}, \ldots, se_n \rangle, a_m)\}$, where $k^i (1 \leq i \leq m-1)$ is the last index in the ith segmentation. Since the input sensor sequence records the interleaved activities, it is highly likely that there exist two sequences that are temporally contained or overlapping, as depicted in Figure 4.

In this algorithm, we consider the reasoner AR can be any technique for recognising sequential (rather than overlapping) activities of a single user, no matter it is from a logical reasoning engine of a knowledge-driven technique or from a trained model of a data-driven technique. In the next section, we will introduce our activity recognition algorithm – matching segmented sensor sequences to activity profiles specified in the AO.

5. PMK and Activity Recognition

Activity recognition is done by finding an activity whose condition *best matches* a sensor sequence. Here we employ the PMK technique [11], which is used to find an approximate correspondence between two sets of hierarchical concepts. PMK has been successfully utilised in recent image-based object detection and matching studies [51].

5.1. PMK and Preliminaries

The principle of PMK is described as follows [25]: let X and Y be two sets of vectors in a D-dimensional feature space. Pyramid matching works by placing a sequence of increasingly finer grids over the feature space and taking a weighted sum of the number of matches that occur at each level of resolution. At any fixed resolution, two points are said to match if they fall into the same cell of the grid; matches found at finer resolutions are weighed more heavily than matches found at coarser resolutions. More formally, let H_X^ℓ and H_Y^ℓ denote the histograms of X and Y at a resolution ℓ, and $H_X^\ell(i)$ and $H_Y^\ell(i)$ are the numbers of points from X and Y that fall into the ith cell of the grid. The number of matches at ℓ is given by the histogram intersection function:

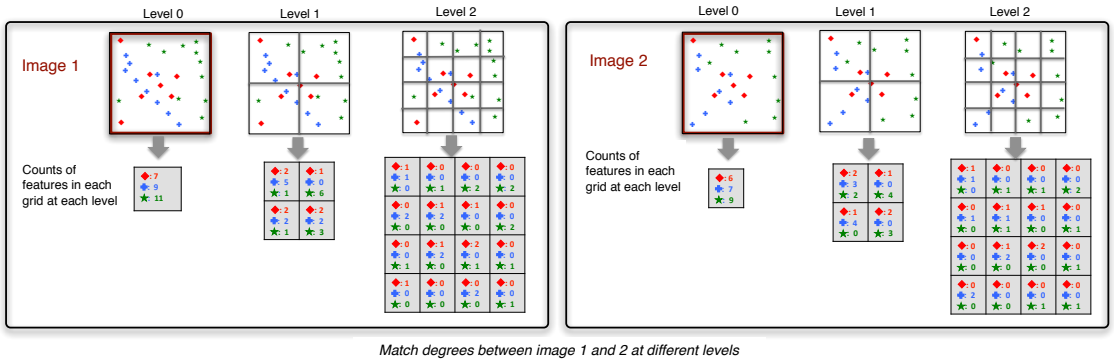
$$\mathcal{I}^\ell = \mathcal{I}(H_X^\ell, H_Y^\ell) = \sum_{i=1}^{N} min(H_X^\ell(i), H_Y^\ell(i)),$$

where N is the total number of cells at a level ℓ along each dimension, which is $N = 2^{D\ell}$.

Since the number of matches found at a coarser resolution ℓ includes all the matches found at the finer level $\ell+1$, the matches at coarser levels involve increasingly dissimilar features. To address this issue, PMK penalises matches at the coarser level by associating the matches at each level ℓ with a weight $\frac{1}{2^{L-\ell}}$, which is inversely proportional to the cell width at that level. Thus we come to the formula:

$$K^L(X, Y) = \mathcal{I}^L + \sum_{l=0}^{L-1} \frac{1}{2^{L-\ell}} (I^\ell - I^{\ell+1}).$$

To illustrate the PMK algorithm more concretely, we develop an example from [25] to calculate the match degree between two images. Figure 7 presents two images: *Image 1* and *2*, each of which has three feature types: red diamonds, blue crosses, and green stars. The original image is considered as *Level 0*, which has one cell. We count the occurrences of each feature in each image at this level and the match degree is the



Level 0	Level 1	Level 2	Penalised degrees
◆: 6 ✚: 7 ★: 9 Total: 22	◆: 6 ✚: 5 ★: 8 Total: 21	◆: 6 ✚: 5 ★: 7 Total: 18	18 + (21 – 18) /2 + (22 – 21) /4 = 19.75

Figure 7: A PMK image matching example

sum of the minimum matches of each feature, which is $min(6,6) + min(9,7) + min(9,9) = 22$. Then we place the first level of grids on these two images respectively, resulting in a regular 2×2 grid. Similarly we count the occurrence of each feature in the four finer-grained cells. Take the red diamond feature as an example. We count the match degrees in the cells from the left to the right and then from the top to the bottom: $min(2,2) + min(1,1) + min(1,1) + min(2,2) = 6$. Then we sum the degrees of the three features to be 18 (=6+5+7).

Repeatedly we place the second level of grids, resulting in a 4×4 grid, and calculate the match degrees at this level cell by cell and feature by feature, whose values have been presented in the tables of Figure 7. If necessary, we can place finer-grained levels of grid over the images, however for the purpose of demonstration, we stop at the second level. The final result is a penalised score on the degrees measured at each level; that is, the sum of the match degrees at the finest grained level and gradually weighted difference degrees at adjacent levels; that is, 18 + (21 -18)/2 + (22 - 21)/4 = 19.75. The score represents the similarity between two images, and the larger the score, the higher the similarity. If we have more images to compare against, we can find out which image best matches a target image by comparing their match degrees. In the following we demonstrate how to extend this image matching principle to perform activity recognition; that is, which activity profile best matches a given sensor sequence.

5.2. PMK-based Activity Recognition

After introducing the basics of PMK, we now describe how to adapt it to support semantic matching between the conditions in activity and sensor events. To prepare, we translate each activity's condition into a D-dimensional vector V, where each dimension represents a type of constraint (i.e., Location, Object, Start Time, and Period). Correspondingly, we construct a set S of D-dimensional vectors extracted from a sensor segment. For a cell i at a resolution level ℓ in a dimension d, we consider a value falling into the cell if the value is more specific to the value corresponding to the cell. Since each element in an activity vector could be composed of a set of values, we call the activity element falling into a cell if any of its values falls into the cell. If the activity element v is more general than the value c representing the cell, the *falling* is quantified as the similarity measure between them; that is, $sim_C(v, c)$.

Instead of employing uniform grids on each feature space, we use the conceptual hierarchies of the OO and LO presented in Figure 3. Thus, the match degrees at each resolution level ℓ will be penalised by a new weight w_ℓ, where w_ℓ is the width under the ℓth level. The final match degree between a sensor sequence

and an activity profile is the summed degree at each resolution level in each dimension, which is formally defined in Definition 4.

Definition 4. Let V be a D-dimensional activity profile and S be a set of D-dimensional vectors extracted from a segment of sensor sequences. The matching degree $\mathcal{K}$ between V and S is calculated in the following formula.

$$\mathcal{K}(V,S) = \sum_{d=1}^{D} \mathcal{I}^{L_d} + \sum_{l=0}^{L_d-1} \frac{1}{\prod_{i=\ell}^{L_d-1} w_i} (\mathcal{I}^{\ell}(d) - \mathcal{I}^{\ell+1}(d)));$$

$$\mathcal{I}^{\ell}(d) = \mathcal{I}(S^{\ell}(d), V^{\ell}(d)) = \sum_{i=1}^{N(\ell)} \sum_{j=1}^{size(S^{\ell}(d))} \delta(i, S_j(d), V^{\ell}(d))$$

$$\delta(i, S_j(d), V^{\ell}(d)) = \begin{cases} min(1, max_{v \in V^{\ell}(d)}(sim_C(v, cell(i))), & \text{if } cell(i) \sqsubseteq S_j(d) \text{ and } cell(i) \sqsubseteq V^{\ell}(d) \\ 0, & \text{otherwise.} \end{cases}$$

where L_d is the number of resolution levels in a dimension d, $N(\ell)$ is the number of cells, and $size(S^{\ell}(d))$ is the size of values in the sensor segment S at a level ℓ in the dimension d.

5.3. Case Study

In this section we present two examples: one as a work-through example of computing the PMK match degree on an unevenly structured hierarchy of the object ontologies using Definition 4; and the other to illustrate how to use the modified PMK to infer activities from a sensor sequence. In the second example, we will also demonstrate the advantage of dealing with sporadic sensor noise and inferring activities with constraints at varying granularities.

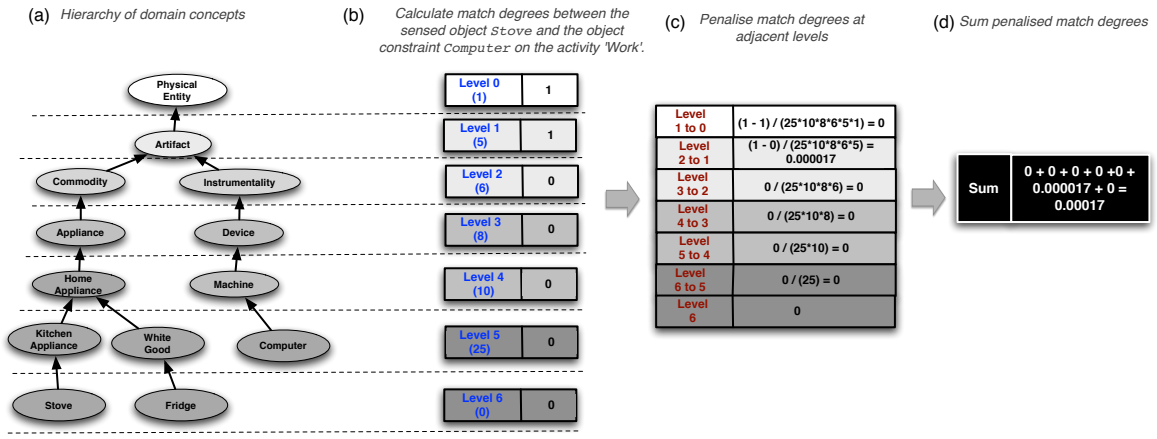


Figure 8: An example of calculating PMK match degree on the object ontology

We start from the simplest example: given one sensor event whose object feature is `Stove` and an activity of `Work` whose object constraint is `Computer`, we calculate their PMK match degree on the object ontology of Figure 3. The OO is assumed to have 7 levels and the widths (i.e., the number of nodes) under each level are respectively 1, 5, 6, 8, 10, and 25. Following a procedure similar to that introduced in the example of Figure 7, we start from the top concept and move down to the finer-grained concepts. That is, at level 0, we count how many times both `Stove` and `Computer` are more specific than the root concept `PhysicalEntity`, which here is 1. At each of the following levels, we count how many times both of these concepts are more specific than any concept on that level. The calculation is illustrated in Figure 8. Then we compute the difference between each adjacent level and penalise by the width at all the finer levels ($\prod_{i=\ell}^{L_d-1} w_i$); for

example, the difference from level 5 to 4 is 0 (=0 - 0) and the penalised weight is 1/(25*10), where 25 and 10 are the widths under levels 5 and 4 respectively. The final PMK match degree is the sum of these penalised differences at each level. In this example the match degree between `Stove` and `Computer` on the object ontology is $0 + 1/(25*10*8*6*5) + 0 + 0 + 0 = 0.000017$.

We have illustrated how to use Definition 4 to calculate the PMK match degree in the simplest form of matching a pair of concepts on an unevenly structured hierarchy in a top-down manner. In the following, we will use a slightly more complicated example to show how we use the PMK match degree to perform activity recognition and deal with sporadically occurring sensor noise.

Example 2. For the sake of the space, we focus our example on the location dimension only. Using the previous example in Listing 1, we extract *location* features from one segmented sensor sequence: $L = (WA, WA, WA, WA, BA, WA)$, where WA and BA represent `workArea1` and `bedArea1` in `bedroom1` (as shown in Figure 3).

We consider three activities whose constraints are specified as: `Sleep` – at the bed area BA, `Work` – at the work area WA, and `Wander` – in the bedroom BR. The `Wander` activity is most coarsely defined, which entails the constraints on the other activities; pure ontological reasoning might not be able to distinguish them. For example, the above sensor segment matches the constraint of the `Wander` activity; however intuitively it suggests the `Work` activity while the single occurrence of BA seems more likely to be a sensor noise.

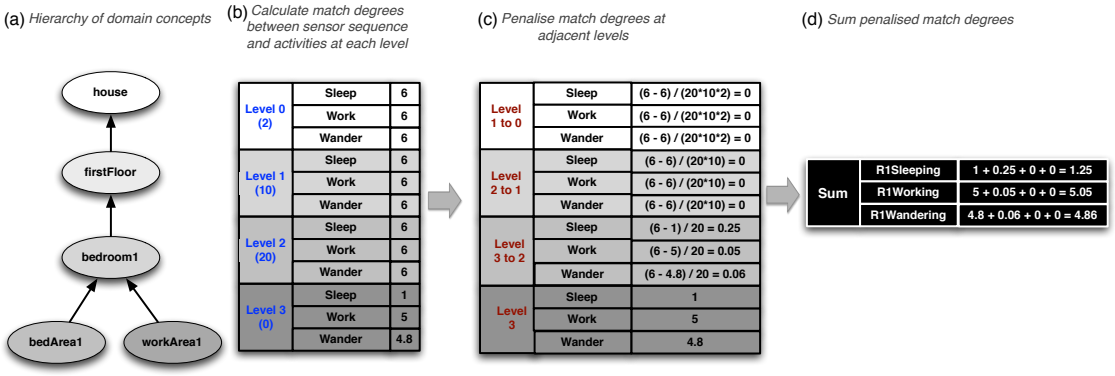


Figure 9: A working example of using the extended PMK to calculate match degrees between a sensor sequence and ontological activity profiles

Through Figure 9 we illustrate the processes of calculating match degrees at each level of the hierarchy, penalising the match degrees at coarser-grained levels, and summing the degrees up to the final score. Figure 9 (a) represents a part of a hierarchy that contains relevant location concepts from Figure 3, which forms the basis of the calculation. We consider the whole house as level 0, the floor as level 1, the rooms in each floor as level 2, and the specific areas contained in each room as level 3. In this example, we assume the widths under level 0, 1, and 2 are 2, 10, and 20 respectively.

The match degree of the sensor sequence and one activity at each level is the count of how many sensor events in this sequence agree with the activity profile in each cell at level of this location feature. For example at level 3, only one sensor event matches with the location constraint of the `Sleep` activity on the specific area BA, so the match degree with `Sleep` at level 3 is 1. Similarly the match degree with `Work` at this level is 5. For the `Wander` activity, the location constraint BR is more general than both BA and WA, both of which are assumed to be the only areas contained in the bedroom. Given that conceptual similarity between BR and BA (WA) is 0.8, the match degree between the sensor sequence and `Wander` activity at level 3 is the sum of the minimum counts on these two areas; that is, $min(1, 0.8) + min(5, 0.8 * 5) = 4.8$.

The next step is to penalise the match degrees at the coarser level. For the `Sleep` activity, the difference

of the degrees between levels 3 and 2 is 5 (= 6 - 1), and the penalisation on the difference is to divide by the width under level 2; that is, 5/20 = 0.25. Moving up the hierarchy, we penalise the degrees level 1 and 0 to be 0 and 0 respectively. We perform the similar process on the `Work` and `Wander` activities. Finally, we sum up the penalised degrees; e.g., adding the penalised degrees from level 3 up to 0 on the `Wander` activity: 1 + 0.25 + 0 + 0 = 1.25. By comparing the final scores between this sensor sequence and each of these activities, we can conclude that the most likely activity is `Work`.

The above example demonstrates that the extended PMK algorithm can accommodate the small amount of sensor noise and infer the correct result. In the following we will illustrate how the PMK can balance the sensor noise with the inference of a coarsely-specified activity.

We slightly change L by replacing one WA with BA. As a result, the match degree on the `Work` activity becomes: 6, 6, 6, and 4, leading to the penalised degree to be 4.1, which is lower than the match degree on the `Wander` activity. Thus we will infer the `Wander` activity as the final result. Now we reflect one step back: what if we use the original PMK formula; that is, we simply count the matching cells rather than use the similarity measure? For the new L, the match degree in the location dimension on the `Wander` activity would be 0.3, from the unpenalised match degrees: 6, 6, 6, and 0. In fact we will not be able to infer the `Wander` activity at all, only `Sleep` or `Work`.

In summary, the underlying assumption of this modified PMK is that when a sensor sequence consists of sensor events that are spread out in the range of a coarser constrained activity, then this method will promote the chance of this activity being inferred. However, if the sensor events indicating a more strongly constrained activity are accumulated then we will still be able to infer the more specific activity.

5.4. Activity Recognition Algorithm

The activity recognition algorithm starts by extracting from an input sensor sequence the semantic features in time, location, object, and person. For each candidate activity, the algorithm checks whether its person constraint is satisfied by the person feature. For example, given the person feature as the user `R1`, if the cooking activity is set to be `Resident`, then we consider this activity's person constraint is satisfied; `R1` is an instance of `Resident` in Figure 3. For all the activities whose person constraint is satisfied, we perform PMK-based matching algorithm on the time, location, and object features. The algorithm will produce the match degrees, indicating how close the input sequence matches to the activity profile. The activity that achieves the highest match score is the inferred result.

6. Experiment and Evaluation

We evaluate the *segmentation* and *recognition* algorithms of KCAR on a well-established real-world data set. To the best of our knowledge, there are few available multi-user data sets that are well annotated in the smart home community. After a careful selection[3], we adopte the 'Interleaved ADL Activities' (IAA) data set from the CASAS smart home project [5]. This data set was collected in a smart apartment testbed hosted at Washington State University during the 2009-2010 academic year. The apartment was instrumented with various types of sensors to detect user movements, interaction with selected items, the states of doors and lights, consumption of water and electrical energy, and temperature, resulting in 2,804,812 sensor events. The majority of sensors in this data set are still positioning sensors, which might restrict the semantics analysis on sensor events and thus limit the effectiveness of KCAR in segmentation and activity recognition. However, with the size of collected data, the length of the collection period, and the availability of the environment knowledge, we still consider IAA as the best available data set to evaluate our technique.

The apartment housed two people, `R1` and `R2`, who performed their normal daily activities during the collection period. The annotated activities along with their recorded occurrence time and location constraints are summarised in Table 1. In this data set, we find that these activities do not have explicit temporal features; e.g., the sleeping activity is rather than defined as a proper night sleep, but as the user lying on the bed even for a few seconds. To be consistent with our knowledge engineering principle: only consider

[3]Lists of home data sets: http://boxlab.wikispaces.com/List+of+Home+Datasets

confident conditions when specifying an activity, we do not place any temporal constraints on the activities. The only objects used in this data set is the equipment in the kitchen (e.g., the burner), which is only relevant to the cooking activity. Therefore, we focus our constraints on the location. For example, the wandering activity is constrained to be in a bedroom, the sleeping activity is constrained to be in a bed area, and the cooking activity is constrained to the kitchen and using the burner object.

The person constraint is used to distinguish who is performing the same activity; for example, `Wander` is defined as an instance of the activity `Wandering`, with a person constraint on the resident R1. We note that in this paper we do not intend to distinguish behaviour for multiple users if they are semantically ambiguous; for example, `R1_Cook` and `R2_Cook` are treated together as one activity `Cook`. This paper focuses on demonstrating the effectiveness of using a very limited amount of more certain and less subjective knowledge in detecting coarse-grained multi-user concurrent activities with explicit semantic implications; e.g., recognising the situation where one user is cooking while the other is working. How to further distinguish individual users' behaviour patterns on the same type of activities is out of the scope of this paper. The reason is that capturing the behaviour patterns for each individual user in performing the same activity is very challenging, which does not only pose the risk of under- or over-specifying due to expert bias but also requires expressive logical language to specify and computationally expensive inference engine to process. Such knowledge is usually better discovered through sophisticated data mining and machine learning techniques [32]. One of our future goals is to combine our technique with such techniques to distinguish finer-grained interleaved activities.

We manually construct the object, location, person, and activity ontologies. We take the spatial layout of the map, identify the critical areas where sensors are deployed, and gradually abstract the areas to rooms, floors, and to the house level. We extract the object ontology from WordNet and manually check the noun terms that are closest to object instances that are associated with the sensors. The person ontology only contains 2 concepts (`Resident` and `ElderlyResident`) and 2 instances as there are only two residents. After construction, the object ontology contains 56 concepts and 5 instances and the location ontology contains 8 concepts and 32 instances, on which we manually define 83 sensor and 13 activity instances from the 10 activity concepts. We note that these ontologies are quite small for recognising activities in a real-world smart home environment. Complement to such small size of knowledge in activity recognition is a novel use of hierarchy-based similarity measure and pattern recognition.

Table 1: The occurrence time and constraints on activities recorded in the IAA data set

Activity	Recorded Time (in hours)	Location Constraint
R1_Sleep	1053.97	R1's bed area
R1_Work	123.44	R1's work area
R1_Wander	1.47	R1's bedroom
R2_Sleep	1335.57	R2's bed area
R2_Work	99.87	R2's work area
R2_Wander	0.62	R2's bedroom
Hygiene	83.83	basin area
Bathing	14.84	bathtub
Housekeep	1.01	living room
Eat	29.1	dining table area
WatchTV	103.03	tv area
Cook	34.05	kitchen
Home	2.30	front hall

6.1. Evaluation Methodology

KCAR is designed as a knowledge-driven activity recognition technique, where we manually specify the activity profiles in ontologies using common-sense knowledge. We do not use any training data to build the

model, so the testing is performed on the whole data set.

We consider a segmentation algorithm to work well if it can detect the boundary between concurrent activities and produce partitions into a small number of fragments that well resemble real activities. That is, the algorithm will be measured in three parameters: *accuracy* – the percentage of boundaries between activities that are successfully detected; *partition percentage* – the percentage of the number of the segmentations over the total size of sensor events; and *resembling scores* – the Chi-Squared scores between the produced segmentations and the actual activity instances recorded in the diary to measure how similar the segmentations are to what really happened.

To evaluate the performance of activity recognition, we use a standard technique – time-slice accuracy [14, 43], which represents the percentage of correctly labelled time slices. The length of the time slice is set to one minute in our experiment. The accuracy is evaluated using two parameters: *precision* and *recall*. Precision is the ratio of the times that an activity is correctly recognised to the times that it is inferred. Recall is the ratio of the times that an activity is correctly inferred to the times that it actually occurs. As an overall measure of accuracy, we use the F-measurement that is a balanced measure of precision and recall together.

We compare the evaluation measurements of KCAR with the other two static sliding window techniques discussed in Section 2.4: (1) *fixed time segmentation* (FTS) where a sensor trace is divided into equal size time intervals, which are set to be 30 and 60 seconds, labelled as FTS-30 and FTS-60 respectively; and (2) *fixed size segmentation* (FSS) where a sensor trace is divided into chunks each of which contains an equal number of sensor events, which are set from 1 to 60, labelled as FSS-1 to FSS-60 respectively. To quantify the improvement of performance (i.e., segmentation and recognition accuracies), we employ the Welch's t-test, which is a standard technique to test the statistical significance of the difference between classifiers [8].

6.2. Parameter Selection

In terms of segmentation, we have mentioned two parameters in Algorithm 1: λ – the maximum time distance threshold, and (θ_T, θ_S) – the semantic threshold pair. λ is set to be 30 minutes, which we consider is enough for a dense sensing environment.

The sensor threshold θ_S is set to separate activities with the furthest semantic distance. The semantic distance between activities is measured on the semantic similarity between sensors that are mapped to these activities. The mapping process is described in Definition 5; that is, if a domain concept used to characterise a sensor (i.e., object and location) is more specific to the corresponding domain concept constrained in an activity, then the sensor is considered as a *key* sensor to the activity. Using this process, each activity will have a collection of key sensors.

Definition 5. For each activity in AO, its object and location constraint is represented as a collection of objects C_O and locations C_L. A sensor s is **mapped** to an activity if there exists an object $o \in C_O$ and a location $l \in C_L$ such that $\delta(s) \sqsubseteq o$ and $\iota(s) \sqsubseteq l$.

For any two activities A, A', we compute the semantic similarity measures between their mapped sensors S, S' and choose the largest value as the *maximum similarity measure* (MSM) between A and A'. The MSM $msm(A, A')$ is calculated as follows:

$$msm(A, A') = max(sim_S(s_i, s_j)), \forall s_i \in S, s_j \in S'.$$

The sensor threshold θ_S is the smallest non-zero value among the MSMs between any two activities. Figure 10 presents the MSM between each activity in the IAA data set, and the semantic threshold is chosen as 0.222.

The time threshold θ_T is set to separate sensor events that are not temporally close. Here we discuss two strategies of setting the threshold: *uniform time threshold* (UTT) and *tuned time threshold* (TTT). In UTT, we set a time distance d and the threshold $\theta_T = 1 - \frac{d}{T_{max}}$. If $sim_T(se_1, se_2)$ in Formula 3 is below θ_T (i.e., their time distance is over d), then we dissect the corresponding sensor events se_1 and se_2. The principle of TTT is to tune the time distance by taking the sensor similarity into account:

$$d'(sim_S) = \frac{d}{e^{\alpha(1-sim_S)}}.$$

	R1_Sleep	R1_Work	R1_Wander	R2_Sleep	R2_Work	R2_Wander	Hygiene	Bath	Housekeep	Eat	Watch_TV	Cook	Home
R1_Sleep	1	0.6	1	0.444	0.444	0.5	0.444	0.444	0.25	0.222	0.222	0.25	0.25
R1_Work		1	1	0.444	0.444	0.5	0.444	0.444	0.25	0.222	0.222	0.25	0.25
R1_Wander			1	0.571	0.571	0.667	0.571	0.571	0.333	0.286	0.286	0.333	0.333
R2_Sleep				1	0.75	1	0.5	0.5	0.286	0.25	0.25	0.286	0.286
R2_Work					1	1	0.5	0.5	0.286	0.25	0.25	0.286	0.286
R2_Wander						1	0.571	0.571	0.333	0.286	0.286	0.333	0.333
Hygiene							1	0.75	0.286	0.25	0.25	0.286	0.286
Bath								1	0.286	0.25	0.25	0.286	0.286
Housekeeping									1	1	1	0.667	0.667
Eat										1	0.75	0.571	0.571
Watch_TV											1	0.571	0.571
Cook												1	0.667
Home													1

Figure 10: Maximum similarity measures between activities in the IAA data set

Compared to the UTT strategy, TTT is more flexible in balancing both the time and sensor thresholds. The intuition is that the sensor events reported in close proximity should be integrated, even though they are not highly similar; this is useful in detecting activities that have coarser constraints. For example, the `R1_Wander` activity is constrained in `R1`'s bedroom that spatially contains both the bed and work areas. These two areas are defined as the location constraints on the `R1_Sleep` and `R1_Work` activities respectively. Using this tuned threshold, we could gather sensor traces that are spread over the room but whose timing gap is close, potentially implying that R1 is wandering. On the other hand, as the sensor threshold θ_S is meant to distinguish the activities whose semantic similarity is furthest, we could use this increasingly tighter time constraint to dissect sensor events whose semantics are not very close. For example, we could dissect sensor sequences for the sleeping and working activities if the timing gap between the events is not close enough. Therefore, in the following we use the TTT strategy to set θ_T and study the impact of difference combinations of the parameters d and α.

6.3. Evaluation of Segmentation

In the following, we will evaluate the three segmentation measurements: segmentation accuracies, partition percentage, and resembling scores.

6.3.1. Segmentation Accuracies

First of all, we measure the accuracy of segmentation. A detection is considered successful if the detected segmentation (dst, ded) is within the range of the recorded segmentation (rst, red); i.e., $dst \in [rst-tt, rst+tt]$ and $ded \in [red-tt, red+tt]$, where tt is the time tolerance.

Figure 11 presents KCAR's accuracies of segmentation on different configurations of the time distance d and α factors in the above TTT strategy. The results show that the larger the α value is, the higher the accuracy; and the larger the d value is, the lower the accuracy. This observation is mainly to do with the intense sensing frequency in this data set; that is, the gap between any adjacent sensor events is as small as 1 second. Therefore, we choose the distance threshold 10 seconds and α value 8 for the following evaluation.

Figure 12 compares the boundary detection accuracies between KCAR and FTS / FSS techniques. The results show that KCAR outperforms these two techniques at different time tolerances. The boundary detection accuracies on the FTS techniques do not vary much at different time intervals (e.g., 30 or 60 seconds). The difference in the evaluation results is mainly to do with the effect of the time tolerance to be evaluated on the time intervals that are used to partition, rather than with the actual segmentations. For example, when the time tolerance is not greater than 30 seconds, then the FTS-30 technique achieves better segmentation accuracies than the FTS-60 technique; and when the tolerance is greater than 30 seconds, the FTS-60 technique takes over.

The boundary detection accuracies on the FSS techniques in the bottom of Figure 12 are not monotonous. As the number of sensors increases from a very low number (e.g., 1), the accuracies go up. When the number of sensors reaches to a certain point (e.g., 12), the accuracies start decreasing, suggesting that the majority of the activity instances recorded in the diary include around 12 sensor events.

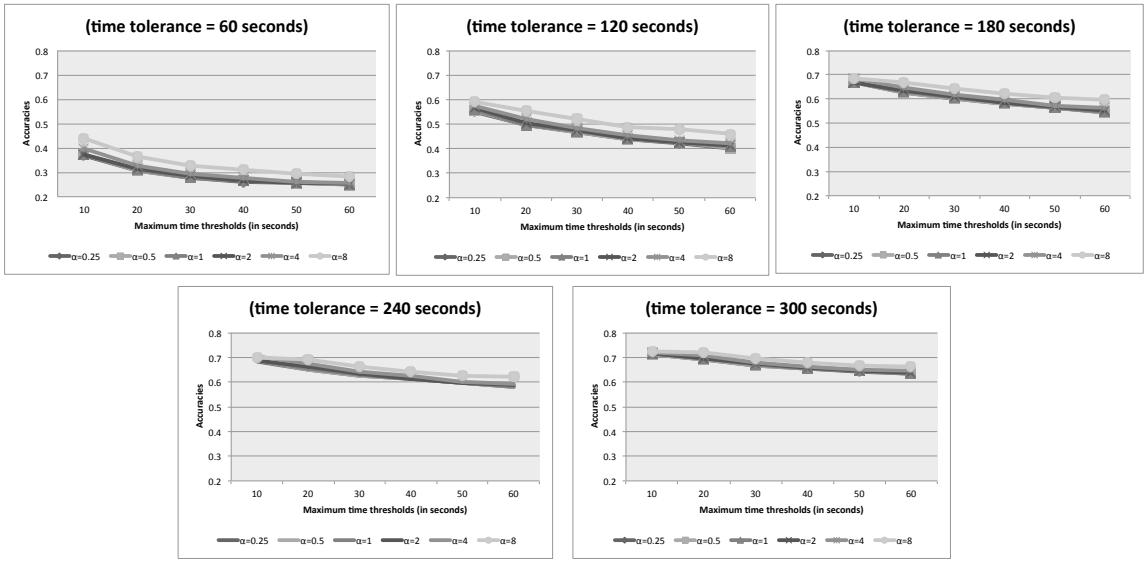


Figure 11: Accuracies of semantic segmentation with various combinations of d and α in TTT

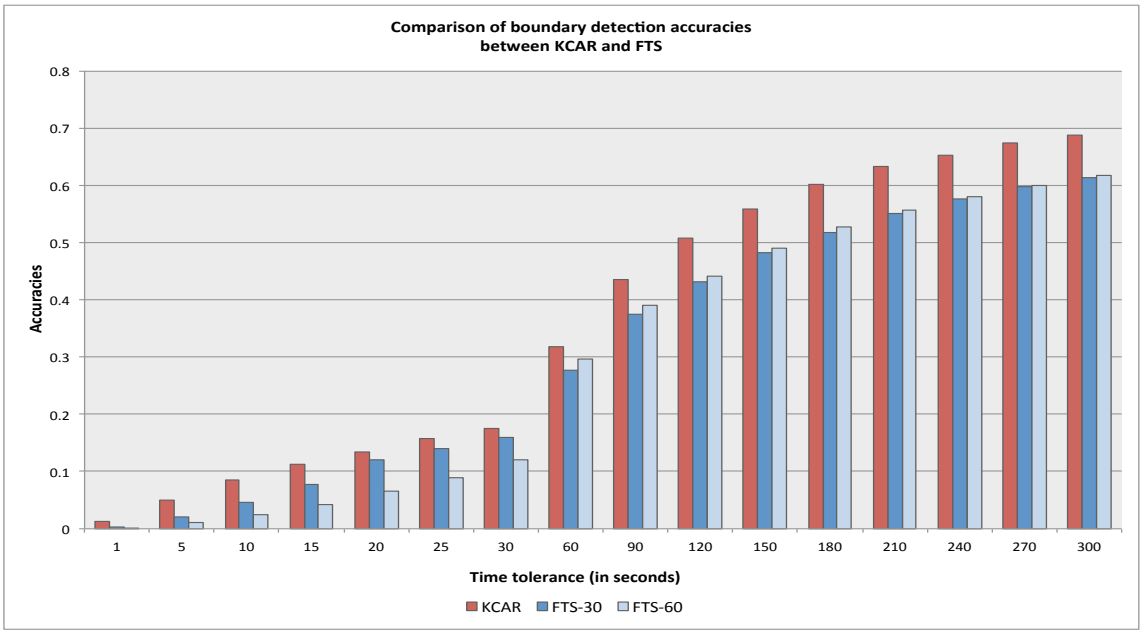
We use Welch's t test to quantify the improvement of the segmentation accuracies. We take the segmentation accuracies of KCAR at different time tolerances, and compare them with the accuracies acquired from each of FTS and FSS techniques. With a null hypothesis H0 of KCAR providing no improvement in accuracy over FTS-30 in segmenting the data set, and an alternative hypothesis H1 of KCAR displaying improvement, we select a standard significance level of 95% for the test, meaning that if the *p-value* in the test result is smaller than 0.05, we reject the null hypothesis and accept that there is a statistically significant improvement. The result shows that KCAR has significantly improved the segmentation accuracies over the FTS techniques and the majority of the FSS techniques, except when the sensor size is set between 5 and 16. The choice on the sensor size might depend on the density of sensor deployment, sampling frequencies of sensors, and the nature of activities of interest. Even though the accuracies on certain FSS techniques are very close to KCAR, in order to set the right size on the FSS technique developers need to acquire a comprehensive understanding of the above knowledge. However, KCAR does not need such heuristic knowledge and can flexibly adapt the sensor sizes without loosing the segmentation accuracies to the FSS techniques; therefore, we argue that KCAR is better than the FSS techniques in segmentation.

6.3.2. Partition Percentage

We expect that KCAR will combine more relevant sensor events into one segment, leading to fewer fragments. This is evaluated in the *partition percentage* — the percentage of the number of the segmentations over the total size of sensor events. Since the FSS techniques divide the whole trace proportionally (e.g., 50% for FSS-2), we only compare the partition percentage with the FTS techniques. The results in Table 2 show that KCAR is more effective in gathering sensor events together.

Segmentation Technique	Number of Segments (Percentage)
KCAR	80,515 (7.76%)
FTS-30	148,048 (14.26%)
FTS-60	96,713 (9.32%)

Table 2: Comparison of partition percentages between KCAR and FTS techniques



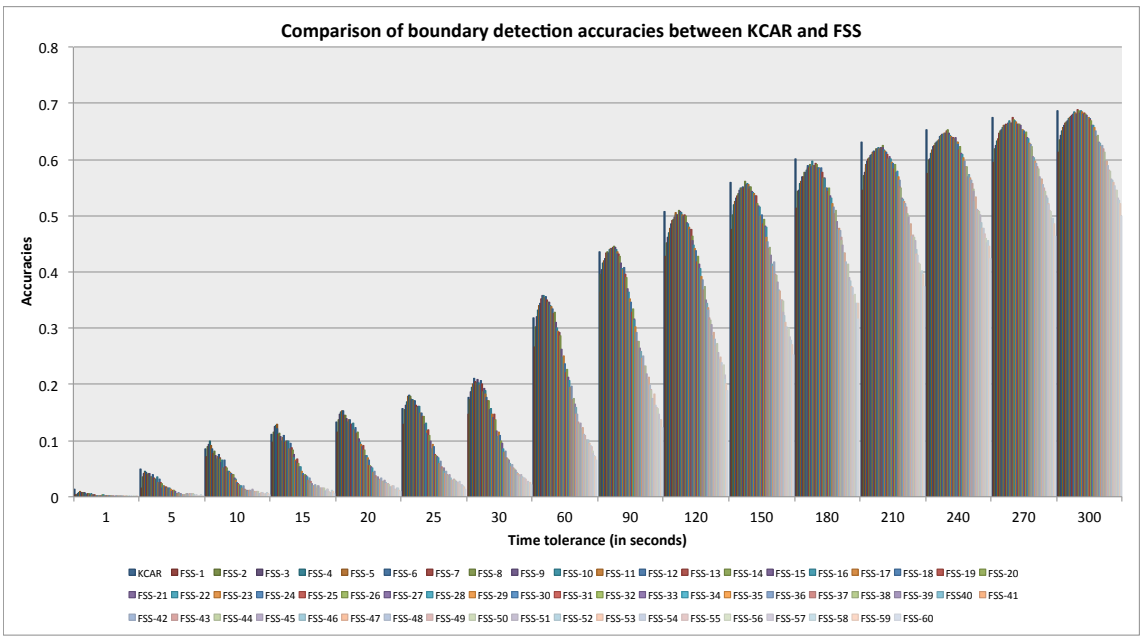


Figure 12: Boundary detection accuracies between KCAR and FTS/FSS techniques

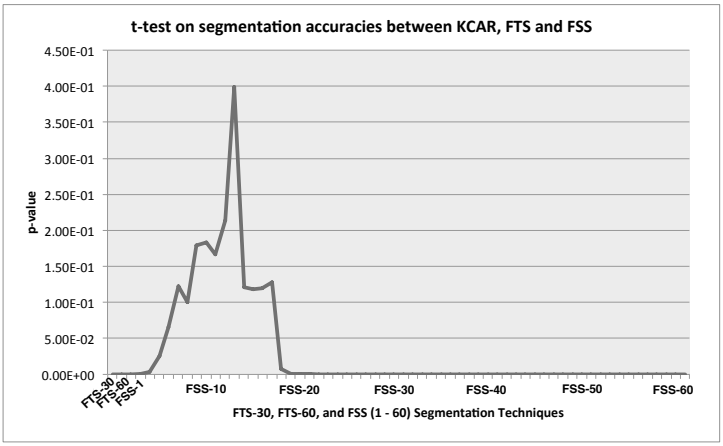


Figure 13: Statistical significance test on boundary detection accuracies between KCAR, FTS, and FSS techniques

6.3.3. Resembling Scores

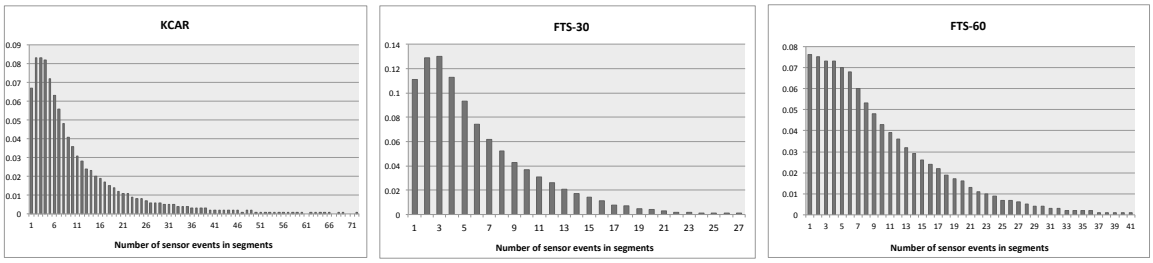


Figure 14: Distribution of numbers of sensor events in segments between KCAR and FTS techniques

The length and complexity of the activities vary; that is, some activities last a short period, while others last longer and trigger more sensor firings. We assume that a diverse distribution in the number of sensor events and the temporal period of fragments resembles more closely real activities. Figure 14 compares the distribution of the number of sensor events in segments between KCAR and FTS algorithms that are configured with 30 and 60 second windows respectively. We can see that the long tail distribution observed for KCAR has a much wider range of the numbers of sensor events contained in each segment and thus exhibits better diversity.

Figure 15 shows the ratio of various time intervals of segments acquired from the FSS algorithms. Due to space limitations, we present the results of the FSS techniques from 10 to 60. We want to evaluate how closely the fragments resemble what happens in the real-world. As we cannot get the ground truth of how many sensor events really contribute to a concurrent activity, we only compare the time interval distribution of the recorded activities with that of the fragments produced by the KCAR and FSS techniques. Here we borrow a standard statistical method, *Chi-Squared* scores, to perform the comparison on a pair of histograms, where the smaller the score, the closer the histograms.

In Figure 16 we present the Chi-Squared scores between each time interval distribution of the recorded activities in the IAA data set and the distributions of the segments resulting from the KCAR and FSS-1 to FSS-60 techniques. The first two figures in Figure 16 show the time interval distribution on the segments produced by the KCAR algorithm, and the interval distribution of the recorded activities in the IAA data set. The results in the rightmost figure show that KCAR has gained the smallest score, suggesting that the

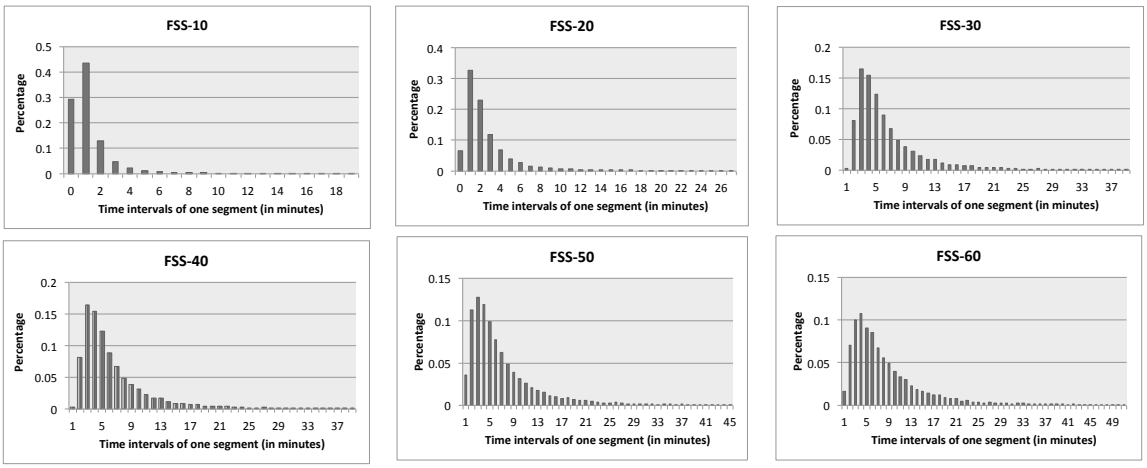


Figure 15: Distribution of time intervals of segments in FSS techniques

segmentations produced by KCAR better resemble the time interval distributions of the true activities. The Chi-Squared scores of the FSS techniques gradually decrease with the increasing size of sensors, and reach to the lowest value (0.52) on FSS-48, which is very close to the score on KCAR (0.51). This implies that the time interval distributions across such a number of sensor events best match the interval distribution of the activity instances, which could indicate why, in this smart environment, the interval across a large number of sensor events matches the interval of the majority of activity instances. However, the segmentation of the FSS-48 technique is much less accurate than KCAR, as shown in Figure 12 and 13.

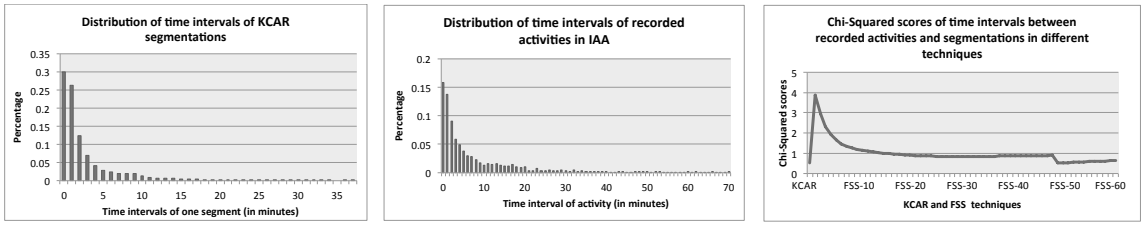


Figure 16: Distribution of time intervals of recorded activities and *Chi-Squared* scores between it and distributions of KCAR and FSS techniques

By combining the comparisons on the three segmentation measurements, we conclude that KCAR can partition a real-time sensor sequence with interwoven, concurrent activities to a higher accuracy than the FTS/FSS techniques. It can also produce a smaller number of segments that better resemble real-world situations, rather than the uniformity offered by the FTS/FSS techniques.

6.4. Evaluation of Activity Recognition

Figure 17 presents the confusion matrix, where each cell is read as the percentage of correct predictions over actual occurrences. The results show that KCAR can accurately recognise the activities with explicit constraints. For example, the `Work`, `Sleep`, `Watch TV`, `Hygiene`, `Cook`, and `Home` activities imply staying in distinguishable location constraints that are exclusive from any other activities, so they are very well recognised. Contrarily, activities like `Wander` and `Eat` that involve movement in a coarsely constrained location are less recognised. One underlying reason could be that the current data set is positioning sensor

	R1_Sleep	R1_Work	R1_Wander	R2_Sleep	R2_Work	R2_Wander	Hygiene	Bath	Housekeep	Eat	Watch_TV	Cook	Home
R1_Sleep	0.915	0.133	0.375	0.059	0.01	0	0.006	0.017	0.003	0.018	0.008	0.002	0
R1_Work	0.02	0.839	0.255	0.023	0.001	0	0.003	0.019	0.003	0.006	0.002	0	0.001
R1_Wander	0.005	0.003	0.319	0.01	0.004	0.05	0.003	0.006	0.003	0.003	0.004	0.002	0.002
R2_Sleep	0.011	0.006	0.051	0.809	0.046	0.442	0.004	0	0	0	0.003	0	0.001
R2_Work	0.006	0.007	0	0.007	0.921	0.292	0.004	0.014	0	0	0.004	0.001	0
R2_Wander	0.002	0.002	0	0.001	0.006	0.2	0.002	0	0	0.001	0.001	0	0.001
Hygiene	0.004	0.001	0	0.004	0.001	0.008	0.937	0	0	0	0.001	0	0
Bath	0.004	0	0	0.002	0	0	0.029	0.905	0	0.001	0	0	0
Housekeep	0.012	0.004	0	0.036	0.003	0.008	0.002	0.027	0.933	0.313	0.155	0.014	0
Eat	0.005	0	0	0.014	0	0	0	0.004	0	0.288	0.07	0.019	0.002
Watch_TV	0.009	0.002	0	0.013	0.001	0	0.001	0.007	0.053	0.311	0.719	0.003	0
Cook	0.004	0.002	0	0.008	0.001	0	0.006	0	0	0.055	0.023	0.951	0.012
Home	0.004	0.001	0	0.015	0.004	0	0.004	0.002	0.003	0.005	0.009	0.007	0.98

Figure 17: Confusion matrix of KCAR recognising concurrent activities

focused, and we believe that the availability of more informative sensors (e.g., object sensors) will help to increase the recognition accuracy for these activities. For example, object sensors on the bed or on the keyboard of the computers might be able to help further distinguish whether the user is actually sleeping, using the computer, or simply wandering around.

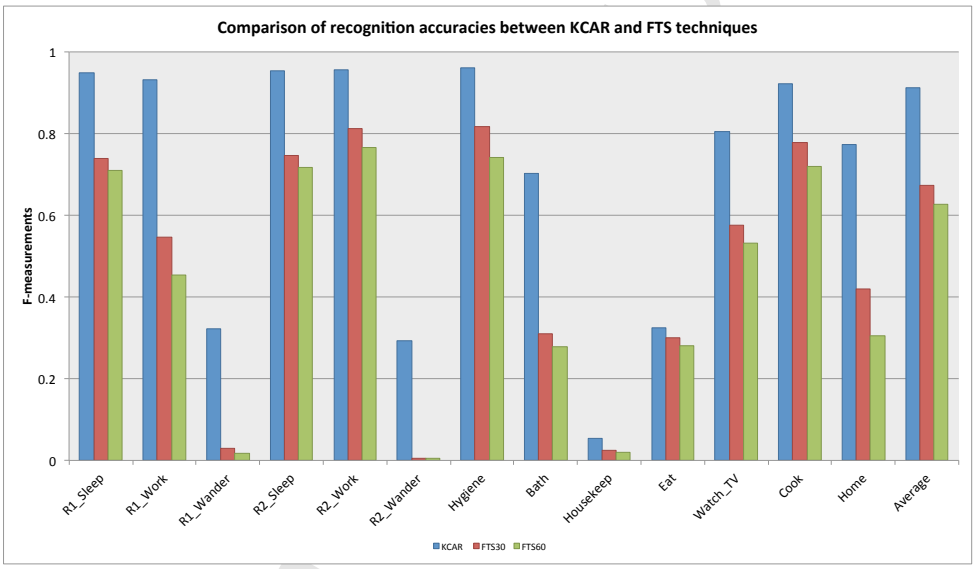


Figure 18: Comparison of F-measurements between KCAR and FTS-30/60 techniques in recognising concurrent activities

To further demonstrate the effectiveness of our segmentation algorithm we compare the recognition accuracy between the KCAR and the above FTS and FSS segmentation algorithms. We perform the same recognition algorithm on the segmented sensor sequences acquired from these algorithms, and their F-measurements of recognising concurrent activities are presented in Figure 18 and Figure 19. The results show that our semantic segmentation technique helps to increase the recognition accuracy compared to these two static sliding window approaches. We also note that the FSS techniques work better than the FTS techniques and the smaller the size of a sensor sequence, the better the recognition accuracy. The phenomena could be relevant to the features of the activities recorded in this data set. By looking at Figure 16 again, we will find that the majority of the activities last a short period (i.e., 1 or 2 minutes), so between these two static approaches, segmentations of a smaller size tend to contain less noisy or interleaving sensor events and thus lead to a better accuracy.

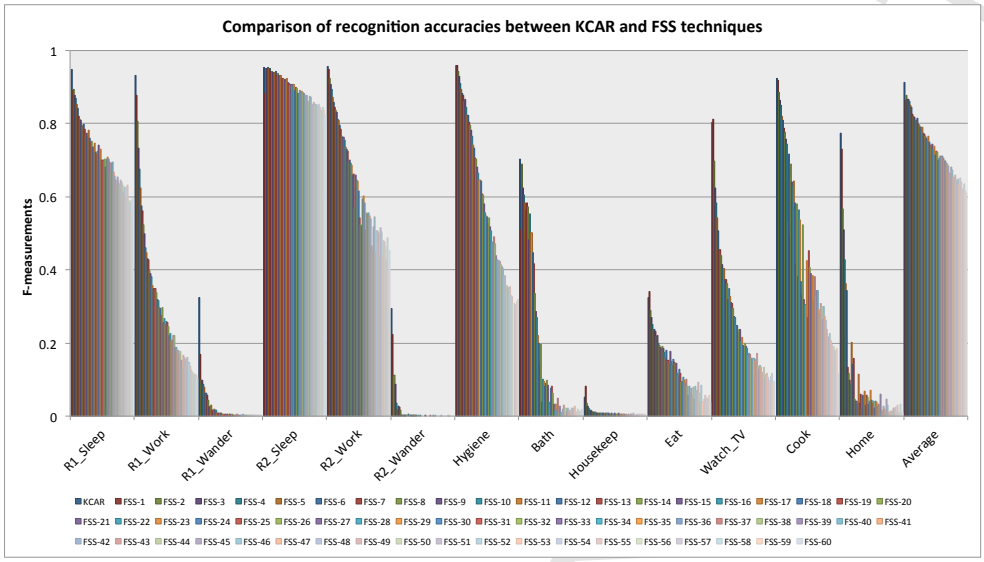


Figure 19: Comparison of F-measurements between KCAR and FSS-1 to -60 techniques in recognising concurrent activities

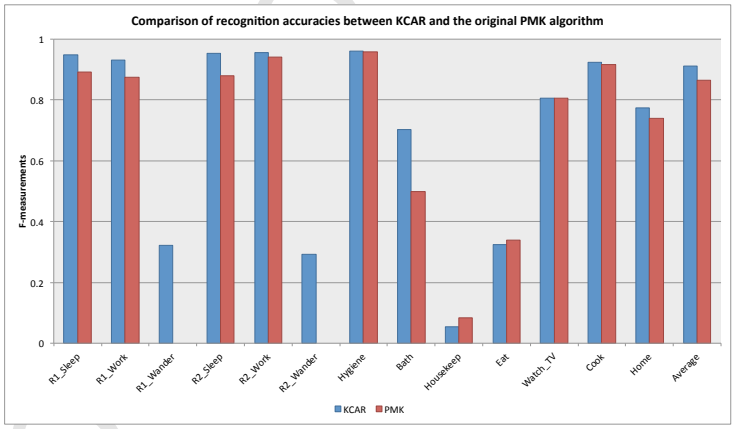


Figure 20: Comparison of F-measurements between KCAR and the original PMK algorithm in recognising concurrent activities

We compare the F-measurement of recognition between KCAR and the original PMK algorithm in Figure 20. The accuracies on the `Wander` activities have been increased from 0 while the accuracies on their counterparts such as the `R1_Sleep` and `R1_Work` activities have not been compromised. This means that KCAR does help to balance sporadic sensor noise in distinguishing coarsely- and finely-specified activities.

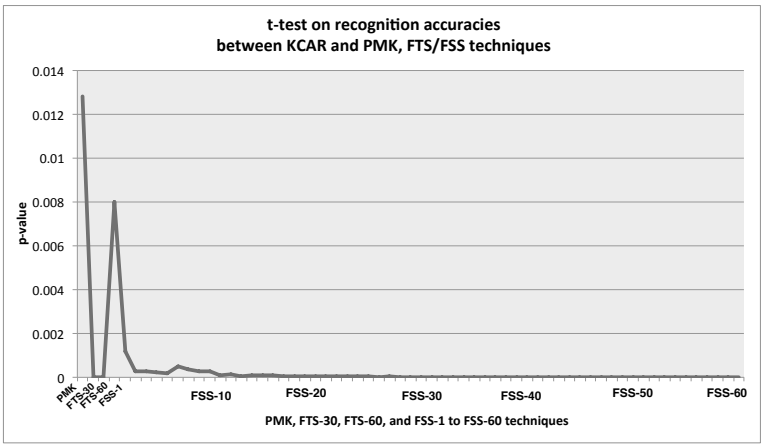


Figure 21: Statistical significance test on recognition accuracies between KCAR, FTS, and FSS techniques

We perform Welch's t-test to measure the improvement of KCAR over the above techniques. We take the F-measurements on each activity of KCAR and compare with the F-measurements resulted from the original PMK, FTS, and FSS techniques. The p-values presented in Figure 21 show that KCAR achieves the statistically significant improvement over these techniques, because all of them are smaller than 0.05, indicating that KCAR provides a statistically significant improvement with 95% certainty. The improvement over the original PMK algorithm is smaller, due to the low occurrences of the `Wander` activities (shown in Table 1).

	Multi-user concurrent activities	Single-user sequential activities
Occurrence ratio	65.9%	34.1%
Recall	93.6%	86.6%

Table 3: Occurrence ratio and recognition accuracies on multi-use concurrent and singular activities

Finally, we focus on the analysis of KCAR in recognising multi-user concurrent activities. Table 3 presents the overall accuracies of recognising single and concurrent activities. More than one activity is being simultaneously performed by different users in 66% of the time in the IAA data set, and the accuracy of recognising these activities is 93.6%. The reason that the accuracy of recognising concurrent activities is higher than that of single activities is that most of the time we recognise more than one activity. There are times where activities have been inferred while there is no corresponding activity recorded in the ground truth.

7. Conclusion

This paper proposes KCAR as a novel approach to recognise multi-user concurrent activities from an unsegmented continuous sensor sequence. By partitioning the interwoven sensor sequence for each ongoing activity, KCAR turns the complex task of concurrent activity recognition into an easier one of single user sequential activity recognition. It leverages strengths both in ontological reasoning and statistical methods

on similarity measure and matching of hierarchical concepts. It benefits from generality, low engineering effort, and no requirement for training data.

By exploring the semantics of sensor events in the ontological model, KCAR can automatically segment a continuous sensor sequence into meaningful partitions. Compared to two classic static sliding window approaches, KCAR can detect the boundary of concurrent activities more accurately, and produce a smaller number of partitions that better resemble real activities. The underlying principle is to only separate sensor events that must be separated, but it does not guarantee the grouped sensor events map to just one user or one activity. In addition, the current segmentation algorithm works well on positioning and objects sensors, and we will look into how to formally define correlations between different types of sensors so that we can segment other types of sensor data.

In terms of activity recognition, KCAR adapts the PMK algorithm to support reasoning on activities that are profiled with constraints of varying granularity and the presence of sporadic sensor noise. The average recognition accuracy achieved for concurrent activities is 93.6%. Currently, KCAR aims towards coarse-grained activities that have very distinguishable semantics from the given domain ontologies; that is, the activities that are performed in different locations, or that involve objects in distinct categories. It cannot further pinpoint activities for individual users. For example, it can only infer `Cook` and `Work` occurring at the same time, but cannot derive how many users are involved in these two activities or determine which user(s) are cooking or working. Neither can it separate finer-grained activities such as stirring a pan or retrieving ingredients. In the future we will look into finer-grained similarity measurements and other machine learning techniques to distinguish individual user's behaviour patterns when they perform the same activities, as identified in Section 6. One possible direction is to use the frequency and gap of sensor events to characterise the temporal features of different users performing the same activity.

8. Acknowledgements

This work is partially supported by the FP7 FET proactive project *SAPERE – Self-aware Pervasive Service Ecosystems*, under grant no. 256873. We thank all the reviewers and the associate editor for their constructive comments and the Systems Group in the School of Computer Science at the University of St Andrews (especially Lei Fang for his advice on statistical significance tests).

References

[1] L. Bao and S. Intille. Activity recognition from user-annotated acceleration data. In A. Ferscha and F. Mattern, editors, *Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2004.

[2] O. Brdiczka, P. Reignier, and J. L. Crowley. Detecting individual activities from video in a smart home. In *Proceedings of the 11th international conference, KES 2007 and XVII Italian workshop on neural networks conference on Knowledge-based intelligent information and engineering systems: Part I*, KES'07/WIRN'07, pages 363–370, Berlin, Heidelberg, 2007. Springer-Verlag.

[3] L. Chen, J. Hoey, C. Nugent, D. Cook, and Z. Yu. Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(6):790–808, 2012.

[4] L. Chen, C. D. Nugent, and H. Wang. A knowledge-driven approach to activity recognition in smart homes. *IEEE Transactions on Knowledge and Data Engineering*, 24:961–974, 2012.

[5] D. Cook and M. Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 48:480–485, 2009.

[6] D. Cook, M. Schmitter-Edgecombe, A. Crandall, C. Sanders, and B. Thomas. Collecting and disseminating smart home sensor data in the casas project. In *Proc. of CHI09 Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*, 2009.

[7] D. J. Cook, J. C. Augusto, and V. R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277 – 298, 2009.

[8] P. Dalgaard. *Introductory Statistics with R*. Statistics and Computing. Springer, 2008.

[9] D.J.Cook. How smart is your home? *Science*, pages 1579–1581, 2012.

[10] S. Gong and T. Xiang. Recognition of group activities using dynamic probabilistic networks. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 742–, Washington, DC, USA, 2003. IEEE Computer Society.

[11] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *J. Mach. Learn. Res.*, 8:725–760, May 2007.

[12] T. Gu, S. Chen, X. Tao, and J. Lu. An unsupervised approach to activity recognition and segmentation based on object-use fingerprints. *Data Knowl. Eng.*, 69(6):533–544, June 2010.
[13] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *CNDS '04*, pages 270–275, January 2004.
[14] T. Gu, Z. Wu, L. Wang, X. Tao, and J. Lu. Mining emerging patterns for recognizing activities of multiple users in pervasive computing. In *MobiQuitous '09: the 6th Annual International conference on Mobile and Ubiquitous Systems: Networking Services*, pages 1–10, 2009.
[15] R. Helaoui, M. Niepert, and H. Stuckenschmidt. Recognizing interleaved and concurrent activities: A statistical-relational approach. In *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications*, PERCOM '11, pages 1–9, Washington, DC, USA, 2011. IEEE Computer Society.
[16] X. Hong and C. Nugent. Partitioning time series sensor data for activity recognition. In *ITAB 2009: the 9th International Conference on Information Technology and Applications in Biomedicine*, pages 1–4, 2009.
[17] D. H. Hu and Q. Yang. Cigar: concurrent and interleaving goal and activity recognition. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI'08, pages 1363–1368. AAAI Press, 2008.
[18] T. Huỳnh, U. Blanke, and B. Schiele. Scalable recognition of daily activities with wearable sensors. In *Proceedings of the 3rd international conference on Location-and context-awareness*, LoCA '07, pages 50–67. Springer-Verlag, 2007.
[19] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*, pages 19–33, Sept. 1997.
[20] T. L. M. Kasteren, G. Englebienne, and B. J. A. Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In L. Chen, C. D. Nugent, J. Biswas, and J. Hoey, editors, *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, chapter 8, pages 165–186. Atlantis Press, Paris, France, 2011.
[21] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. D. Mynatt, T. Starner, and W. Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, CoBuild '99, pages 191–198, London, UK, UK, 1999. Springer-Verlag.
[22] E. Kim, S. Helal, and D. Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1):48–53, Jan. 2010.
[23] N. C. Krishnan and D. J. Cook. Activity recognition on streaming sensor data. *Pervasive and Mobile Computing*, 2012.
[24] J. O. Laguna, A. G. Olaya, and D. Borrajo. A dynamic sliding window approach for activity recognition. In *UMAP '11*, pages 219–230, Berlin, Heidelberg, 2011. Springer-Verlag.
[25] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR '06*, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.
[26] C. Leacock, G. A. Miller, and M. Chodorow. Using corpus statistics and wordnet relations for sense identification. *Comput. Linguist.*, 24(1):147–165, Mar. 1998.
[27] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *Proceedings of the 9th international conference on Ubiquitous computing*, UbiComp '07, pages 483–500, Berlin, Heidelberg, 2007. Springer-Verlag.
[28] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, BSN '06, pages 113–116, Washington, DC, USA, 2006. IEEE Computer Society.
[29] S. McKeever, J. Ye, L. Coyle, and S. Dobson. Activity reasoning using temporal evidence theory. *Ambient Intelligence and Smart Environment*, 2:253–269, 2010.
[30] G. A. Miller. Wordnet: a lexical database for English. *Commun. ACM*, 38(11):39–41, Nov. 1995.
[31] J. Modayil, T. Bai, and H. Kautz. Improving the recognition of interleaved activities. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 40–43, New York, NY, USA, 2008. ACM.
[32] N. T. Nguyen, S. Venkatesh, and H. Bui. Recognising behaviours of multiple people with hierarchical probabilistic model and statistical data association. In *Proceedings of the British Machine Vision Conference*, pages 126.1–126.10. BMVA Press, 2006. doi:10.5244/C.20.126.
[33] G. Okeyo, L. Chen, H. Wang, and R. Sterritt. Dynamic sensor data segmentation for real-time knowledge-driven activity recognition. *Pervasive and Mobile Computing*, 2012.
[34] D. J. Patterson, D. Fox, H. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, ISWC '05, pages 44–51, Washington, DC, USA, 2005. IEEE Computer Society.
[35] P. Rashidi and D. J. Cook. Mining sensor streams for discovering human activity patterns over time. In *ICDM 2010*, pages 431–440, 2010.
[36] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
[37] D. Riboni and C. Bettini. Context-aware activity recognition through a combination of ontological and statistical reasoning. In *UIC '09*, pages 39–53. Springer-Verlag, 2009.
[38] Saguna, A. Zaslavsky, and D. Chakraborty. Recognizing concurrent and interleaved activities in social interactions. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 230–237, Dec.
[39] A. Sheth, C. Henson, and S. Sahoo. Semantic sensor web. *Internet Computing, IEEE*, 12(4):78–83, 2008.

[40] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757, Aug. 2000.

[41] E. Tapia, S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. In A. Ferscha and F. Mattern, editors, *Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science*, pages 158–175. Springer Berlin Heidelberg, 2004.

[42] E. M. Tapia, S. S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive'04*, pages 158–175, 2004.

[43] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 1–9, New York, NY, USA, 2008. ACM.

[44] S. Wang, W. Pentney, A.-M. Popescu, T. Choudhury, and M. Philipose. Common sense based joint training of human activity recognizers. In *IJCAI '07*, pages 2237–2242, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[45] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994.

[46] J. Ye, L. Coyle, S. Dobson, and P. Nixon. Representing and manipulating situation hierarchies using situation lattices. *Revue d'Intelligence Artificielle*, 22(5):647–667, 2008.

[47] J. Ye, L. Coyle, S. Dobson, and P. Nixon. Using situation lattices in sensor analysis. In *PERCOM '09: Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–11, Texas, USA, Mar. 2009.

[48] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: a review. *Pervasive and Mobile Computing*, 8:36–66, Feb. 2012.

[49] J. Ye, G. Stevenson, and S. Dobson. A top-level ontology for smart environments. *Pervasive and Mobile Computing*, 7:359–378, June 2011.

[50] J. Yin, Q. Yang, and J. J. Pan. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1082–1090, 2008.

[51] T.-H. Yu, T.-K. Kim, and R. Cipolla. Real-time action recognition by spatiotemporal semantic and structural forests. In F. Labrosse, R. Zwiggelaar, Y. Liu, and B. Tiddeman, editors, *Proceedings of British Machine Vision Conference*, pages 1–12, Aberystwyth, UK, 2010. British Machine Vision Association.

[52] F. Baader, D. Calvanese, D.-L. McGuinness, D. Nardi, and P. Pate-Schneider. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 0-521-78176-0, 2003.

Juan Ye currently is a lecturer in School of Computer Science at University of St. Andrews, UK. Her research areas are pervasive and ubiquitous computing and wireless sensor network. Her interest resides in designing and developing knowledge- and learning-hybrid techniques in human activity recognition, with speciality in ontology, context modelling and reasoning, and uncertainty resolving techniques. She has published in a number of international journals and conferences including PMC, KER, PerCom, Pervasive, LoCA, and ICPS.

Graeme Stevenson is a Ph.D. candidate in the School of Computer Science at the University of St Andrews, UK. He holds an MPhil. in Computer Science. His core research interests lie in the development of context modelling abstractions, middleware for pervasive sensor systems, and in the development of models to infer human behaviour from noisy sensor-data. He has over 30 publications in international workshops, conferences, and journals.

Simon Dobson is Professor of Computer Science at the University of St Andrews, having returned to the UK in 2009 after 12 years in Ireland. He publishes in top venues in adaptive systems (ACM Trans. Auto. Adap. Sys., IEEE Trans. Sens. Net., IEEE Trans. Inst. Measure., Per. Mob. Comp., UBICOMP, IWSOS, IEEE SMC), and is internationally known for his contributions in the design, analysis and implementation of sensor-driven, pervasive and autonomic systems, with an emphasis on integrating semantic techniques, situation recognition, uncertain reasoning and other techniques into programming languages. He has been PI/coI on grants including Sapere (€2.5M), Clarity: the Centre for SensorWeb Technologies (€6M), NEMBES: Networked Embedded Systems (€11M), and Lero: the Irish Software Engineering Research Centre (€11M). He has supervised 11 PhDs (to completion) and 5 PDRAs, and currently leads a group of 4 PhDs and 1 PDRA. He is an associate editor of ACM Trans. Auto. Adapt. Sys, and was invited expert for the EU strategic initiatives Situated and Autonomic Computing and Beyond the Horizons. He was also the co-founder and CEO of a start-up company, growing it to a valuation of €1.8M. He is a Chartered Engineer and a Fellow of the British Computer Society.

Figure
Click here to download high resolution image

Figure
Click here to download high resolution image

Author Photo

Click here to download high resolution image