Semantics-Driven Multi-user Concurrent Activity Recognition

Juan Ye and Graeme Stevenson*

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

Abstract. This paper presents a novel knowledge-driven approach to recognising multi-user concurrent activities in smart home environments. Capturing these concurrent activity patterns is challenging in that it usually requires detailed application-/user-specific specifications, or needs a large amount of data to build sophisticated models. The proposed approach is founded upon the use of a generic ontology model to represent domain knowledge, which is independent of particular sensor deployment and activities of interest. It leverages the hierarchical structure of domain concept ontologies and applies well-established hierarchy-based techniques to enable automatic segmentation of real-time sensor traces and supports matching finely grained sensor data to coarsely constrained activities. We empirically evaluate our approach using a large-scale real-world dataset, achieving an average accuracy of 86%.

1 Introduction

Recognising human activities in a sensor-instrumented environment is an important task in ambient intelligent environments. Earlier works mainly focus on identifying activities for a single user [20], however there are often multiple residents living in the same environment, performing different tasks *concurrently*. Distinguishing these activities is essential to the development of customised context aware applications in a real-world setting [5].

Recognising multi-user concurrent activities is challenging due to interwoven sensor traces and the numerous, flexible, and complicated ways of human conducting activities simultaneously. Sophisticated data-driven techniques have been designed to capture concurrent activity patterns [10], which usually leads to heavy computation and a craving for training data, and thus is unsuitable for real-time recognition. These learned models are also at risk from being overfitted to the training data, which is very likely to be different from the data at later real-time running. Few knowledge-driven techniques [18] have been applied, 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 unsupervised approach for recognising multi-user concurrent activities. We think from a new direction:

* This work has been supported by the EU FP7 project "SAPERE - Self-aware Pervasive Service Ecosystems" under contract No. 256873.

J.C. Augusto et al. (Eds.): AmI 2013, LNCS 8309, pp. 204–219, 2013.

we consider one of the most important and pre-requisite processes toward recognising concurrent activities to be the automatic segmentation of a real-time *sensor trace* – composed of a sequence of raw sensor events – into semantically meaningful parts, each of which is assumed to correspond to one of the concurrent activities. Then we recognise each activity from segmented fragments by treating it as the single activity recognition problem.

Our approach falls into the knowledge-driven category, however, the main difference from existing techniques is that we use the generic ontologies (e.g., WordNet and the top-level ontology [21]) and necessary conditions to constrain activities which will result in more certain knowledge and require much less knowledge engineering effort. The top-level ontology model formally explores the universal hierarchical structure in all types of domain concepts, inspired by which we adapt two well-established hierarchy-based techniques to enable sensor trace segmentation and activity recognition. First of all we utilise the Wu's conceptual similarity measure [19] to study the similarity between sensor events based on their semantics annotated in the ontological model and use the similarity to segment a sensor trace. To recognise activities, we employ the Pyramid Match Kernel (PMK) which is an effective technique in supporting approximate matches between two sets of hierarchical concepts and has been widely used in recent image-based object detection and matching studies [22]. We extend the PMK for matching sensor events to activities, because the original PMK formula does not suit mapping sensor events to ontological activity description; e.g., the activity profile specified in the activity ontology and raw sensor events might not share the same level of abstraction and the domain concept ontologies are not uniformly hierarchical. We validate our segmentation and recognition algorithms using a large scale real-world dataset. Through comprehensive experimental studies, we demonstrate both the effectiveness and flexibility of our proposed algorithms.

The rest of the paper is organised as follows. Section 2 reviews the recent techniques in recognising interleaved and concurrent activities. Section 3 presents a generic ontological model to represent sensor events, objects, and locations, based on which we explore the semantics between sensor events and segment a sensor trace into fragments. Section 4 introduces an activity ontology that describes necessary conditions on an activity, and applies the PMK technique to support activity recognition. We evaluate our technique on a real-world data set in Section 5 and conclude the paper with future research directions in Section 6.

2 Related Work

Activity recognition, one of the main research topics in ambient intelligence [2], aims to identify users' daily activities from observed sensor readings. Different data-, knowledge-driven, and hybrid techniques are proposed in the literature, among which Hidden Markov Models (HMMs) have not only demonstrated promising accuracies in recognising single-user sequential activities [9] but also are presented as one of the most popular techniques in recognising interleaved and concurrent activities.

Patterson et al. [16] employ the HMM to recognise interleaved activities of a morning routine using RFID data. Having experimented with various HMMs, 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. [13] propose an *interleaved HMM* that aims to capture inter- and intra-activity dynamics. It recognises activities based on the last object used by the subject. This technique performs well in recognising certain interleaved activities and can, to a certain degree, deal with sensor data noise.

Gong et al [3] developed a dynamically multi-linked HMM 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. [14] 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. [8] propose a novel probabilistic framework for multi-goal recognition where both concurrent and interleaving goals can be recognised. The technique used is skip-chain conditional random fields (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. As mentioned in [10], 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 of craving for training data.

Helaui et al. [7] build composite activity models using the Markov Logic Network, a statistical relational approach to incorporate common sense background knowledge. Gu et al. [5] 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. Our approach also uses feature relevance to segment the boundary of adjacent activities, however we apply a more formal knowledge-driven approach to segment real-time sensor traces solely based on the semantic similarity of raw sensor events, without the need for extra activity knowledge.

In knowledge-driven approaches, ontologies are one of the most popular [6,17,15]. Chen et al. [15] present an ontological mode to represent smart home activities and relevant context. The approach is motivated by the observations that ADLs 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 dataset collection and training. More recently, Saguna et al. [18] combine ontological and spatiotemporal modelling and reasoning to recognise interleaved and concurrent activities. However their approach requires detailed understanding and thorough investigation of the activity related specifics 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.

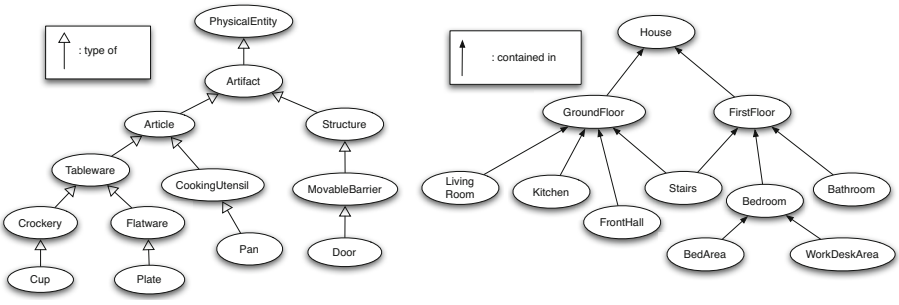
3 Sensor Semantics and Segmentation

This section describes the sensor and the associated domain ontologies based on which we introduce a semantic measure on evaluating the similarity between raw sensor events. Following on, we illustrate an algorithm to segment a real-time sensor trace into coherent fragments.

3.1 Domain and Sensor Ontologies

Central to our technique is a general ontological model that consists of four main components: `Object`, `Location`, `Sensor`, and `Activity`. Among them, the object and location ontologies are general (also called *domain ontology* in this paper) in that they represent concepts of an application domain or in a certain environment. By linking the sensor and activity ontologies to these two, we can make sense of raw sensor data and infer activities from them.

The object ontology (OO) describes the `type-of` relationships between household objects. For example, a *cup* is a type of *crockery*, denoted $cup \sqsubseteq crockery$. To make the OO as general as possible, we extract it from WordNet [12], which



(a) The OO example: a hierarchy of household objects from WordNet (b) The LO example: a hierarchy of rooms in a house

Fig. 1. The domain Object and Location ontologies

is a hierarchical lexical system where words are organised by semantic relations in terms of their meaning. Figure 1 (a) shows a part of the OO[1].

The location ontology (LO) describes the `containment` relationships between locations in terms of their spatial layout. Figure 1 (b) presents an example of room layouts in an ordinary house in the form of a lattice. A house consists of two floors, connected by a set of stairs.

The sensor ontology (SO) represents sensors and sensor events. A sensor event is usually represented in a tuple $se = (st, sId, val)$, indicating at the time st a sensor whose id is sId reports a value val. The sensor id refers to a sensor, which has a type (e.g., energy, object use, or temperature) and can be linked to an attached object and an installed location. Both objects and locations are instances defined in OO and LO. The sensor value can be abstracted into high-level terms (such as the status of the object like 'open' or 'close', or a particular property of an environment 'hot' or 'cold').

3.2 Semantic Similarity and Segmentation

Based on the conceptual models described in the above section, we will discuss *semantic similarity* with which to automatically segment a *sensor trace* that is composed of a sequence of sensor events.

Definition 1. Let se_i and se_j be two sensor events. The **semantic similarity** between them is defined as

$$sim(se_i, se_j) = (sim_T(st_i, st_j), sim_S(sId_i, sId_j)), \tag{1}$$

$$sim_S(sId_i, sId_j) = \frac{sim_C(so_i, so_j) + sim_C(sl_i, sl_j)}{2}, \tag{2}$$

where

- st_i (st_j), sId_i (sId_j), so_i (so_j), and sl_i (sl_j) are respectively the timestamp, sensor id, object that the sensor is attached to, and the location where the sensor is located.
- The sim_T is the time similarity function that compares the time distance between the sensor events.
- The sim_S is the sensor similarity function that normalises the conceptual similarity functions sim_C on their objects and locations.

The conceptual similarity function is built on the algorithm proposed by Wu et al. [19]. The idea is to find the least common subsumer (LCS) of the two input concepts and compute the path length from the LCS up to the root node. The LCS is the most specific concept that these two concepts share as an ancestor.

[1] For brevity, we omit intermediate terms.

Definition 2. Let c_1 and c_2 be two concepts organised in one hierarchy. The **conceptual similarity** measure between them is calculated as:

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

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 equals 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 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 1, the conceptual similarity between a *cup* and a *plate* is $\frac{2*3}{2+2+2*3} = 0.6$, while the similarity between *cup* and *door* is $\frac{2*1}{3+4+2*1} = 0.22$.

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

$$sim_T(st_1, st_2) = min(1, 1 - \frac{|st_2 - st_1|}{T_{max}}), \tag{3}$$

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 coarse-grained form is calculated as Definition 2. In this paper, we take the fine-grained form, because the time hierarchy can be application specific, which violates the generality principle. However, we note this as a future research direction.

Example 1. In this example, we calculate the semantic similarity between the sensor events se_1, se_2 and se_3 in Figure 2. The location and object concepts refer to the LO and OO in Figure 1.

- se_1 and se_2: their time similarity is $1 - \frac{10}{86400} = 1$ using Equation 3; their sensor similarity is $(0.22 + 0)/2 = 0.11$ using Definition 1.
- se_2 and se_3: Similarly, their time similarity is $1 - \frac{25}{86400} = 0.9997$ and their sensor similarity is $(0.22 + 0)/2 = 0.11$.
- se_1 and se_3: their time similarity is $1 - \frac{35}{86400} = 0.9996$ and their sensor similarity is $(0.6 + 1)/2 = 0.8$ where the location similarity between these two sensors is 1.

In the above example, based on the similarity measures between these three sensor events, we can intuitively observe that the sensor events se_1 and se_3 should be grouped together, and se_2 should be partitioned from the former two.

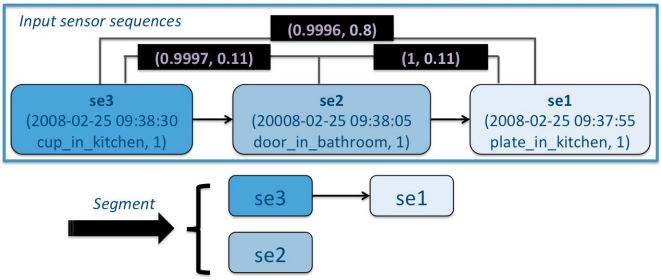


Fig. 2. Segmenting a sequence of interwoven sensor events based on their semantic similarity

This sequence could suggest that there are two users: one is cooking in the kitchen, while the other is using the bathroom. This example introduces our idea of *automatic segmentation* of sensor traces that record concurrent activities; that is, dividing a sequence of raw sensor events into segments, each of which is composed of sensor events that are semantically similar, intuitively corresponding to one of the concurrent activities.

Algorithm 1 illustrates the segmentation process. During the process, we maintain two lists: a *confirmed* list L_c that stores all the segmented sequences that are distant from the time of the current sensor event and will no longer be concatenated with other sensor events, and a *tentative* list L_t that records the unfinished sequences that are likely to be concatenated with new sensor events.

Algorithm 1. Automatic segmentation of a real-time sensor trace

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 threshold pair
Result: L_S: a list of segmented sequences
initialise(L_c);
initialise(L_t);
foreach $se \in SEQ$ **do**
  found = false;
  if *(!L_t.isEmpty)* **then**
    for $i = 1$ **to** $L_t.size$ **do**
      $se_i = L_t.get(i).last$;
      $(sim_T, sim_S) = sim(se, se_i)$;
      if $sim_T \leq \theta_T$ **then**
        $L_c.add(L_t.get(i))$;
        $L_t.remove(i)$;
      else if *(!found &&* $similarityCheck(sim_T, sim_S, \theta_T, \theta_S)$) **then**
        $L_t.add(se)$;
        $found = true$;
  if !$found$ **then**
    $formList(se)$;
    $L_t.add(se)$;
return L_c;

Given an incoming sensor event, se, we calculate the similarity measure between it and the previous sensor event in each sequence in the tentative list L_t; that is, $(sim_T, sim_S) = sim(se, se_i)$, $1 \leq i \leq m$, where m is the size of L_t, and se_i is the last sensor event in the ith sequence. If there exists a sequence i such that the similarity $sim(se, se_i)$ satisfies the given time and sensor thresholds θ_t and θ_s, the new event se will be joined with the sequence i as $\langle \ldots, se_i, se \rangle$. If the current se is different from all the sequences in L_t, then a new list $\langle se \rangle$ will be formed and 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.

During the process we also check whether any list i in the tentative list should be moved to the confirmed list by assessing the time similarity sim_T. If the similarity is too small, the list is distant from the current time and will be unlikely to be further linked with any new sensor events; in this case it will be moved to the confirmed list L_c. In the end, the segmentation result is $L_c = \{\langle se_1, se_2, \ldots, se_{k^1} \rangle, \langle se_{k^1+1}, \ldots, se_{k^2} \rangle, \ldots, \langle se_{k^{m-1}+1}, \ldots, se_n \rangle\}$, where $k^i (1 \leq i \leq m-1)$ is the last index in the ith segmentation. Since the input sensor sequence records the concurrent activities, it is highly likely that there exists two sequences that are temporally contained or overlapping.

4 Activity Ontology and Recognition

After segmenting real-time raw sensor traces into fragments, we need to infer an activity for each fragment. This section describes an activity ontology that defines a profile with *time*, *location*, and *object* constraints. The activity recognition process is to find the activity whose profile best matches the current sensor fragment using the Primary Match Kernel (PMK) technique.

4.1 Activity Ontology

In the activity ontology (AO) we constrain each activity with *time*, *object*, and *location* conditions. Generally there are two types of conditions to constrain activities: *sufficient* and *necessary*. A sufficient condition is the most common type that has been used in knowledge-driven activity recognition techniques. If a sufficient condition is satisfied, then an activity is occurring. An example rule on the activity 'sleep' is:

$$person\_in\_bedroom \wedge bed\_is\_accessed \wedge time\_is\_night$$
$$\wedge\, light\_in\_bedroom\_low \wedge door\_in\_bedroom\_closed$$
$$\Rightarrow user\_is\_sleeping.$$

However, sufficient conditions are usually user-specific, over-detailed and do not always hold. In the above example, a user could sleep while leaving the bedroom's door open, or leaving the light on. Contrastingly, a necessary condition restricts what must hold when a user performs an activity, meaning that if an

activity is occurring, then its necessary condition must hold. In other words, if the condition does not hold, then it is impossible that the user is performing activity. In the AO, only the necessary conditions are specified on activities.

We consider two types of *time* conditions: *occurring* time – when the activity usually occurs and *duration* – how long this activity typically lasts. For example, we place the constraint that the activity 'prepare breakfast' should occur in the morning (6am - 12am), and that the activity 'bath' should last more than 5 minutes. The *object* and *location* condition specifies what object the user must access to in order to perform this activity and where this activity must occur. To note that the necessary conditions that we specify here should be general knowledge in the sense that most people will execute the activity in this way. A summary of necessary conditions on the 'prepare breakfast' activity is presented as follows.

Activity	Occurring Time	Duration	Location	Object
prepare breakfast	[6am - 12am]	30min	kitchen	{cooking utensil, tableware}

As we can see, compared to the sufficient condition, the necessary condition is more certain and concise. The object and location conditions are defined on concepts in the domain ontologies OO and LO. Thanks to the existence of the OO and LO hierarchies, we can specify the conditions using *more general* concepts rather than listing each specific concept. For example, we could directly constrain the objects for the 'prepare breakfast' activity on cooking utensil or tableware, without the need of enumerating each concrete entity like a cup, plate or pan.

4.2 Pyramid Match Kernel and Activity Recognition

Activity recognition is done by finding an activity whose necessary condition *best matches* the currently segmented sensor traces. Here we employ the PMK technique [4], 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 [22].

The principle of PMK is described as follows [11]: let X and Y be two sets of vectors in a D-dimensional feature space. Pyramid matching works by placing a sequence of increasingly coarser 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, so that $H_X^l(i)$ and $H_Y^l(i)$ 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 penalise 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: $K^L(X,Y) = \mathcal{I}^L + \sum_{l=0}^{L-1} \frac{1}{2^{L-\ell}}(I^\ell - I^{\ell+1})$.

Given the above, we next adapt PMK to support semantic matching between the conditions in activities and sensor data. To prepare, we translate each activity's necessary condition into a D-dimensional vector V, where each dimension represents a dimension of constraint (i.e., Location, Object, and Time). 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.

Instead of employing uniform grids for each feature space, we use the conceptual hierarchies of the OO and LO presented in Figure 1. Thus, the match degrees at each resolution level ℓ will be penalised by a new weight $\frac{1}{c_l{}^{L_d-\ell}}$, where c_ℓ is the maximum width at ℓ. The final match degree between a sensor trace and an activity profile is the summed degree at each resolution level in each dimension, which is formally defined in Definition 3.

Definition 3. Let V be a D-dimensional activity profile and S be a set of D-dimensional vectors extracted from a segment of sensor traces. The matching degree between V and S is calculated in the following formula.

$$pmk\_match(V,S) = \frac{\sum_{d=1}^{D}(\mathcal{I}^{L_d} + \sum_{l=0}^{L_d-1} \frac{1}{c_\ell{}^{L_d-\ell}}(I^\ell(d) - I^{\ell+1}(d))))}{|S| \times D};$$

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

$$\delta(i, S_j(d), V^\ell(d)) = \begin{cases} 1, \text{ if both } S_j(d) \text{ and } V^\ell(d) \text{ fall into the cell } i \\ 0, \qquad\qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

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.

Example 2. We present a concrete example of calculating the match degree between a sensor segment and the 'sleep' activity. We extract *time*, *location*, and *object* data from a 3-length sensor segmentation as follows: T = (23:12:38, 23:13:02, 23:13:04), $L = (BA, WDA, BA)$, $O = (Bed, Chair, Bed)$, where BA and WDA represent a bed and work desk area in a bedroom (as shown in Figure 1). The matching degree at these dimensions are calculated as follows.

Dimension	Constraint	PMK Degree
Occurring time	[8pm, 12pm], [0am, 8am]	3
Location	BA	2.5
Object	Bed	2.5

For the occurring time, we assume there is no hierarchy for temporal concepts, so we simply match the sensing time to the range, and the matching degree here is 3. Given the location constraint as the bed area (BA), the unpenalised match degrees from the coarsest level 0 to the finest level 3 respectively are 3, 3, 3, and 2. Then we penalise the match degrees by the weight and get the final result: $(0 + 0 + 1 * 1/2 + 2) = 2.5$. The match degree on the object dimension can be similarly calculated as 2.5. Then the total normalised match degree for this sensor fragment to the 'sleep' activity is $(3+2.5+2.5)/(3*3) = 0.89$.

The recognition of an activity is done by matching the extracted feature set from a sensor segment to each activity's conditions. The activity that achieves the highest score is the inferred result.

5 Experiment and Evaluation

We evaluate our *segmentation* and *recognition* algorithms on a well-known real-world dataset. To the best of our knowledge, there are few available multi-user datasets that are well annotated in the smart home community. After careful selection[2], we adopted the 'Interleaved ADL Activities' (IAA) dataset from the CASAS smart home project [1]. This dataset 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 statuses of doors and lights, consumption of water and electrical energy, and temperature, resulting in $2,804,812$ sensor traces. With the variety of sensors, number of sensor traces, length of the collection period, and the availability of the environment knowledge, we consider IAA as suitable to evaluate our technique.

The apartment housed two people, *R1* and *R2*, who performed their normal daily activities during the collection period. Within the AO described in the earlier section, each activity is constrained by the time, location and object conditions. In this dataset, 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 confident conditions when specifying an activity, we do not place any temporal constraints on the activities. The only objects used in this dataset is the equipment in the kitchen (e.g., the burner), which is only relevant to the 'meal preparation' activity. Therefore, we focus our constraints on the location. For example, the activity 'R1_Wander' is constrained to be in resident R1's bedroom, the activity 'R1_Sleep' is constrained to be in resident R1's bed area, and the activity 'Meal_Preparation' is constrained to the kitchen and using the burner object. The annotated activities along with their recorded occurrence time and location constraints are summarised as follows.

[2] Lists of Home Datasets: `http://boxlab.wikispaces.com/List+of+Home+Datasets`

Activity	Time(in hour)	Location Constraint
R1_Sleep	1053.97	R1's bed area
R1_Work	123.44	R1's work desk area
R1_Wander	1.47	R1's bedroom
R2_Sleep	1335.57	R2's bed area
R2_Work	99.87	R2's work desk area
R2_Wander	0.62	R2's bedroom
Meal (Preparation)	34.05	kitchen
Hygiene	83.83	bathroom
Bath	14.84	bathtub
Home	2.3	fronthall

In the end, the object ontology contains 20 instances and the location ontology contains 15 instances, on which we define 83 sensors, and 10 activities. 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.

We note that in this paper we do not intend to distinguish behaviour for multiple users if they are semantically ambiguous; for example, 'R1_Meal_Preparation' and 'R2_Meal_Preparation' are treated together as one activity 'Meal_Preparation', while 'R1_Sleep' and 'R2_Sleep' are considered individually because they have explicit location semantics. The reason is that capturing the behaviour patterns for each individual user in performing the same activity is very challenging, which not only poses 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 [14]. One of our future goals is to combine our technique with such techniques to distinguish finer-grained interleaved activities. This paper focuses on demonstrating the effectiveness of using a very limited amount of more certain and less subjective knowledge in detecting multi-user concurrent activities with explicit semantic implications.

5.1 Parameter Selection

In terms of segmenting a real-time sensor trace, we have mentioned two parameters in Algorithm 1: *sensor* and *time* semantic thresholds. We set the semantic similarity threshold to be 0.5. In terms of the time threshold, we set a time distance d and the threshold $\theta_T = 1 - \frac{d}{T_{max}}$, and if $sim_T(st_1, st_2)$ 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. Here we propose another way to set the time threshold, called a Tuned Time threshold Configuration mechanism (TTC). The principle of TTC is to tune the time distance by taking the sensor similarity into account:

$$d'(sim_S) = \frac{d}{10^{1-sim_S}}.$$

Compared to setting the uniform time threshold configuration (UTC), TTC is a more flexible way to balance both the time and sensor thresholds. The intuition is that the sensor events reported in close proximity should be tied up, 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 on R1's bedroom that spatially contains both the bed and work areas, which 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 were spread over the room but whose timing gap was close, potentially implying that R1 is wandering. On the other hand, as the sensor threshold is roughly set, we could use this increasingly tighter time constraint to dissect sensor events whose semantics are not very close. For example, we could dissect sensor traces for the sleeping and working activities if the timing gap between the traces was not close enough.

5.2 Evaluation of Segmentation

We consider a segmentation algorithm to work well if it can partition the whole sensor trace into a much smaller number of fragments while still able to detect wherever there is a boundary between interleaved and concurrent activities. Therefore the performance of segmentation is measured using two parameters: (1) *partition percentage* – the percentage of the number of the segmentations over the total size of sensor traces and (2) *accuracy* – the percentage of boundaries between activities that is successfully detected. More specifically, a detection is considered successful and counted if the time at the detected segmentation (either start or end time) dst is within the range of the boundary time (again, either start or end time of an activity) rst; i.e., $dst \in [rst - tt, rst + tt]$, where tt is the time tolerance that is set to be 2 minutes in our experiment.

The evaluation result is presented in Figure 3, where we compare the performances of the above TTC and UTC mechanisms. As presented in Figure 3, our segmentation algorithms can partition a large sequence of sensor events into a much smaller number of fragments (whose partition percentages vary from 25% to 2%). The detection accuracy is close to 1, indicating we can detect the boundary successfully. Combined with both these measures, we can conclude that our algorithm is able to select and combine sensor events that resemble one activity and separate sensor events that correspond to different activities.

5.3 Evaluation of Recognition

After segmentation we use the recognition algorithm to infer the most likely activity for each segment. Figure 4 presents the overall accuracies of recognising single and concurrent activities. In the IAA dataset, there is 44% of the time that more than one activities are being simultaneously performed by different

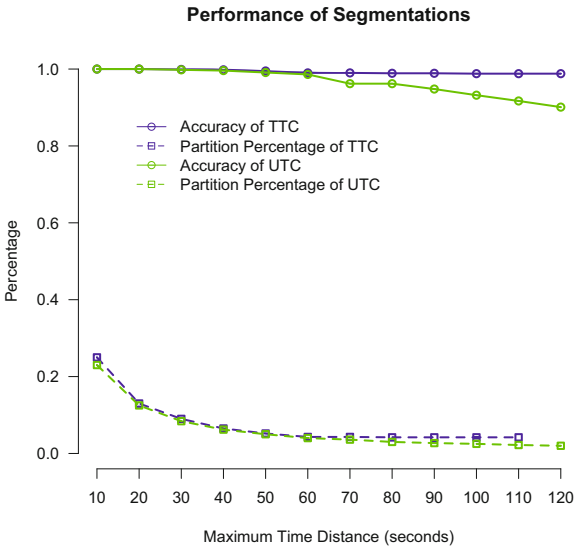


Fig. 3. Partition percentages and accuracies of segmentation

users, and the accuracy of recognising these activities is 95%. The reason that the accuracy of recognising concurrent activities is higher than that on single activity is that most of time we recognise more than one activities. There are times that the activities have been inferred while there is no corresponding activity recorded in the ground truth.

Single Activity		Concurrent Activity		Overall Accuracy
Proportion	Accuracy	Proportion	Accuracy	
56%	82%	44%	95%	86%

Fig. 4. Proportion and accuracies of single- and concurrent-activities in *IAA*

Figure 5 presents the confusion matrix of recognising the activities in *IAA*, where each figure is read as the percentage of predictions over actual occurrences. The result shows that the recognition algorithm can accurately recognise the activities with explicit constraints. For example, the 'Meal', 'Bath', 'Home' activities have distinguished location constraints that are exclusive from any other activities, so they are very well recognised. In contrast, the 'Wander' and 'Hygiene' activities are less well recognised since they are specified with coarser-grained constraints.

	R1_Sleep	R2_Sleep	R1_Wander	R2_Wander	R1_Work	R2_Work	Meal	Hygiene	Bath	Home
R1_Sleep	0.931	0.103	0.074	0	0	0	0	0	0	0
R2_Sleep	0	0.672	0	0.125	0	0	0	0	0	0
R1_Wander	0.069	0.164	0.852	0	0	0	0	0	0	0
R2_Wander	0	0.017	0	0.75	0	0	0	0	0	0
R1_Work	0	0.014	0.074	0	1	0	0	0	0	0
R2_Work	0	0	0	0.125	0	1	0	0	0	0
Meal	0	0	0	0	0	0	1	0	0	0
Hygiene	0	0.011	0	0	0	0	0	0.962	0	0
Bath	0	0.017	0	0	0	0	0	0.038	1	0
Home	0	0	0	0	0	0	0	0	0	1

Fig. 5. Confusion matrix for recognising the activities in *IAA*

6 Conclusion and Future Work

This paper presents a novel approach to combining ontologies with recent advances in semantic matching techniques to recognise daily human activities. The technique benefits from generality, low engineering effort, and no requirement for training data. We have demonstrated the effectiveness of our technique in segmenting sensor traces online, and recognising multi-user concurrent activities. The technique works well on activities that have explicit semantics, but is limited in its ability to distinguish coarsely constrained activities from similar but more finely constrained activities. In the future, we will explore other derivative PMK functions to tackle this problem.

Also the current segmentation algorithms work well on object use sensors such as motion sensors or RFID, 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. Additional areas of future research will focus on distinguishing the individual user participating in a common activity, and extending our ontological model with time series analysis techniques and machine learning techniques; that is, using the frequency and gap of sensor events to characterise the temporal features of different users performing the same activity.

References

1. Cook, D., Schmitter-Edgecombe, M.: Assessing the quality of activities in a smart environment. Methods of Information in Medicine 48, 480–485 (2009)
2. Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 5(4), 277–298 (2009)
3. Gong, S., Xiang, T.: Recognition of group activities using dynamic probabilistic networks. In: ICCV 2003, Washington, DC, USA, pp. 742–749 (2003)
4. Grauman, K., Darrell, T.: The pyramid match kernel: Efficient learning with sets of features. J. Mach. Learn. Res. 8, 725–760 (2007)
5. Gu, T., Chen, S., Tao, X., Lu, J.: An unsupervised approach to activity recognition and segmentation based on object-use fingerprints. Data Knowl. Eng. 69(6), 533–544 (2010)

6. Gu, T., Wang, X.H., Pung, H.K., Zhang, D.Q.: An ontology-based context model in intelligent environments. In: CNDS 2004, pp. 270–275 (January 2004)
7. Helaoui, R., Niepert, M., Stuckenschmidt, H.: Recognizing interleaved and concurrent activities: A statistical-relational approach. In: PERCOM 2011, pp. 1–9. IEEE Computer Society, Washington, DC (2011)
8. Hu, D.H., Yang, Q.: Cigar: concurrent and interleaving goal and activity recognition. In: Proceedings of the 23rd National Conference on Artificial Intelligence, AAAI 2008, vol. 3, pp. 1363–1368. AAAI Press (2008)
9. Kasteren, T.L.M., Englebienne, G., Kröse, B.J.A.: Human activity recognition from wireless sensor network data: Benchmark and software. In: Chen, L., Nugent, C.D., Biswas, J., Hoey, J. (eds.) Activity Recognition in Pervasive Intelligent Environments. Atlantis Ambient and Pervasive Intelligence, vol. 4, ch. 8, pp. 165–186. Atlantis Press, Paris (2011)
10. Kim, E., Helal, S., Cook, D.: Human activity recognition and pattern discovery. IEEE Pervasive Computing 9(1), 48–53 (2010)
11. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR 2006, pp. 2169–2178. IEEE Computer Society, Washington, DC (2006)
12. Miller, G.A.: Wordnet: a lexical database for english. Commun. ACM 38(11), 39–41 (1995)
13. Modayil, J., Bai, T., Kautz, H.: Improving the recognition of interleaved activities. In: Proceedings of the 10th International Conference on Ubiquitous Computing, UbiComp 2008, pp. 40–43. ACM, New York (2008)
14. Nguyen, N.T., Venkatesh, S., Bui, H.: Recognising behaviours of multiple people with hierarchical probabilistic model and statistical data association. In: Proceedings of the British Machine Vision Conference, pp. 126.1–126.10 (2006)
15. Okeyo, G., Chen, L., Wang, H., Sterritt, R.: Dynamic sensor data segmentation for real-time knowledge-driven activity recognition. Pervasive and Mobile Computing (2012)
16. Patterson, D.J., Fox, D., Kautz, H., Philipose, M.: Fine-grained activity recognition by aggregating abstract object usage. In: ISWC 2005, pp. 44–51. IEEE Computer Society (2005)
17. Riboni, D., Bettini, C.: Context-aware activity recognition through a combination of ontological and statistical reasoning. In: Zhang, D., Portmann, M., Tan, A.-H., Indulska, J. (eds.) UIC 2009. LNCS, vol. 5585, pp. 39–53. Springer, Heidelberg (2009)
18. Saguna, Zaslavsky, A., Chakraborty, D.: Recognizing concurrent and interleaved activities in social interactions. In: DASC 2011, pp. 230–237 (December 2011)
19. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: ACL 1994, Stroudsburg, PA, USA, pp. 133–138 (1994)
20. Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: a review. Pervasive and Mobile Computing 8, 36–66 (2012)
21. Ye, J., Stevenson, G., Dobson, S.: A top-level ontology for smart environments. Pervasive and Mobile Computing 7, 359–378 (2011)
22. Yu, T.-H., Kim, T.-K., Cipolla, R.: Real-time action recognition by spatiotemporal semantic and structural forests. In: Proceedings of British Machine Vision Conference, Aberystwyth, UK, pp. 1–12. British Machine Vision Association (2010)