Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Human-Behaviour Study with Situation Lattices

Juan Ye
CLARITY: Centre for Sensor Web Technologies
University College Dublin
Dublin, Ireland
juan.ye@ucd.ie

Simon Dobson
CLARITY: Centre for Sensor Web Technologies
University College Dublin
Dublin, Ireland
simon.dobson@ucd.ie

***Abstract*—Most research in the area of smart environments focuses on improving the accuracy with which human activities can be recognised. Relatively little research has been done into how designers can gain insights into the behaviours their systems are observing, and feed these insights back into improving systems design. We describe a mathematical structure, the *situation lattice*, and show how it can be used to discover knowledge about activities and the way in which they can be sensed. We show how this knowledge can be used to improve activity recognition, using the example of a real-world smart home data set.**

***Index Terms*—Smart home, Human Behaviour, Knowledge Discovery**

I. Introduction

Studying human behaviours in a smart environment has been a popular research area over years. One of the typical topics in this area is *activity recognition*; that is, recognise human activities from raw sensor data. It has been shown that the ability to correctly identify the day-to-day activities of subjects may have significant implications and applications in human-beneficial areas; for example, healthcare [1].

Raw sensor data collected in a smart environment include environmental information (such as temperature or humidity), interaction information between a subject and objects being used or accessed by the subject (such as Radio-frequency IDentification, called RFID), and personal information about the subject (like heart rate). The daily activities are pre-defined with developers' common sense knowledge in a descriptive manner. For example in the PlaceLab data set [2], an activity "using phone" is defined as using a portable phone or the phone on the fax machine in the office; and an activity "grooming" is defined as getting dressed or undressed, or styling hair. Following these definitions, developers (or the third party or the subject) annotate what a subject is doing from recorded video [2] or audio [11], which will serve as the ground truth for the evaluation of an activity recognition technique in terms of its accuracy or efficiency.

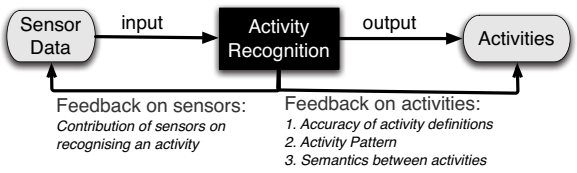

Fig. 1. Activity Recognition

The process of activity recognition is encapsulated in a black box, where sensor data are input and human activities are inferred as output (shown in Figure 1). The main research focuses on how to efficiently and accurately infer activities. Beyond this research, we are also interested in the underlying research questions:

1) *What is an activity pattern from the perspective of sensors*; that is, what do sensors (or sensor data) contribute to determining this activity? This question will build up experience for other smart home set-ups.
2) *Are activities accurately defined*? An accurate descriptive definition of an activity means that the pattern of this activity can be specified in characterised sensor data according to this definition. A daily activity is intrinsically imprecise so it is difficult to be described precisely. In the above example, "grooming" is not accurately defined, since it covers "styling hair", which is hardly captured by sensors and thus is not related to any distinguishable sensor data. The imprecise definition of an activity could make unidentifiable some more specific activities that should be recognisable (for example, getting dressed or undressed). It could also undermine the effect of an activity recognition technique since the technique might try to extract characteristics of an activity that are hardly tractable. Therefore, developers need an approach to evaluate their description on activities.
3) *What activity does a subject usually conduct given particular sensor data (for example, what does the subject do when he is in the living room)*? This question is about deriving the usual pattern of a subject's everyday activities, which will help to spot abnormal behaviours. For example, a subject could be inferred "sleeping" in the kitchen, which violates his activity patterns, so the subject might actually pass out, which is an emergent situation.
4) *What relationships do exist between different activities*; That is, what activities are subsumed by an activity, or what activities the subject cannot conduct simultaneously? This question can improve the accuracy of a recognition process, and can provide guidance on designing activity-aware applications.

This paper will describe *a situation lattice* that can be used to address the above questions by making transparent the knowledge in the activity recognition process. The situation

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

lattice has been used in recognising activities in a smart home environment and the produced evaluation result is promising [3]. This paper will demonstrate the use of the lattice in exploring human activity patterns. The demonstration will use the lattice built on the PlaceLab data set, which is the public data set gathered in the real-world smart home environment [2]. This lattice [3] covers the following sensors: the wireless infrared motion sensors, electrical current, water, and gas flow sensors, switch sensors, RFID, and object motion sensors.

This paper is organised as follows. Section II reviews the literature in the activity recognition area. Section III introduces the basic theoretical concepts in a situation lattice, which are used to extract activity pattern and explore semantics of activities in Section V. Section VI concludes with some future directions.

II. Related Work

Machine learning (or data mining) techniques have been widely used in activity recognition, including decision tree [4], [2], [5], Bayesian inferencing [6], [7], Hidden Markov Models (HMMs) [8], [9], and Conditional Random Fields (CRFs) [10], [11].

Bao *et al* [4], [5] use decision trees to learn user body motions (such as bicycling, shaking hands, or typing) from the raw sensor data provided by accelerators on the human body. Van Kasteren *et al* [7] carry out activity recognition using a Bayesian framework. They use a static Bayesian model to learn the relationship between different sensor data and human activities, and also use a dynamic Bayesian network to model the temporal aspects of activities. Bui *et al* [6] use a multi-layer Bayesian dynamic structure, called an Abstract HMM, to track an object and predict its future trajectory in a wide-area environment. This structure is used to explicitly encode the complex and scalable spatial layout; that is, the hierarchy of connected spatial locations. Trained with coordinate-based location data, it can predict the evolution of the object's trajectory at different levels of detail.

Modayil *et al* [12] use interleaved HMMs to recognise multi-tasked activities where a person switches frequently between steps of different activities such as making a stir-fry, making a jello, and drinking a glass of water. The interleaved HMM records the last object observed from wrist-worn RFID sensors for each activity as a hidden state.

Liao *et al* [10] employ CRFs to construct models of high-level activities such as work, leisure, and visit. They use a person's GPS data to learn his activities over a few weeks, and then determine the relationship between the activities and places that are important to this person.

All these works make effort on improving the accuracy of inferring human activities from a number of advanced sensors (mostly on-body motion sensors). However, the knowledge discovered in these techniques is barely usable other than inferring activities. For example, decision tree is assumed to be able to represent the knowledge into rules, but the rules are sometimes not understandable since they could be contaminated by much noise in the sensor data. Our earlier work has used the C4.5 decision tree in Weka software [13] to learn human activities from raw sensor data [3]. The derived rules do not follow the human natural understanding of the activities. The reason is that decision tree learns the knowledge simply based on the characteristics of data while no domain knowledge is involved. When the data are imperfect, then the knowledge would be inaccurate.

In contrast, the situation lattice will allow developers to represent their domain knowledge; that is, the knowledge about sensors and environments. Incorporating this knowledge in a training process, the lattice can learn more knowledge about sensor data and human activities. Since the newly discovered knowledge is constrained by domain knowledge, it will make more sense and be useful in studying human behaviour patterns.

III. Introduction to Situation Lattices

This section will briefly introduce the basic concepts in a situation lattice, which provides the foundation for the following sections. A more detailed theoretical description can be found in our earlier work [14], [3].

A. Theoretical Model of Situation Lattices

A situation lattice is built on the basic concepts of lattice theory. It is used to study the relationship between sensor data, and relationships between activities and different types of sensor data. Within a situation lattice, sensor data are abstracted in characteristic functions, called *context predicates* in our work. The lattice consists of a set of nodes, each of which represents a logical description of context predicates, which takes context predicates as input and applies the logical conjunction on them. Each node is associated with a set of activities, which implies that when the logical description on this node is satisfied by the current sensor data, a user is possibly conducting any of the activities in this set.

All nodes are organised with a *specialisation* relationship. A node n_i is considered more *specific* than another node n_j, labelled as $n_i \sqsubseteq n_j$, if and only if the logical description on n_i entails that on n_j. The specialisation relationship implies that a node will be activated if and only if all its more general nodes are activated.

Context predicates can have rich relationships, including different levels of abstraction level, conflicting, and overlapping [15]. A situation lattice supports representing these relationships and uses them to explore semantic relationships between human activities.

Given two context predicates p_i and p_j and their corresponding nodes n_i and n_j such that $n_i.l = p_i$ and $n_j.l = p_j$, where $n_i.l$ represents a logical description on a node n_i,

- If p_i is finer grained than p_j, then $n_i \sqsubseteq n_j$. The different levels of granularity is defined as: a context predicate p_i is finer grained than another predicate p_j, if any sensor data that satisfies the former context predicate also satisfies the latter one. For example in Figure 2 (a), the context predicate `inLivingRoom` is finer grained

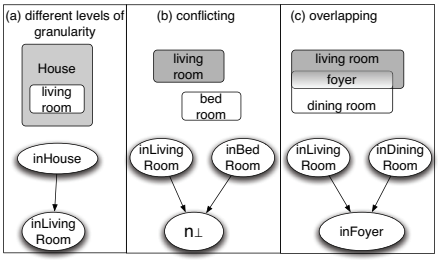

Fig. 2. Semantic relationships between context predicates

than the predicate `inHouse`, since the living room is contained in the house.

- If p_i conflicts with p_j, then $n_i \sqcap n_j = n_\perp$, where $\sqcap$ is the meet operator and the bottom node $n_\perp$ is the unique node whose logical description is the logical contradiction `FALSE` [3]. The conflicting is defined as: two context predicates are conflicting if any sensor data that satisfies one of them cannot satisfy the other. For example in Figure 2 (b), the context predicate `inLivingRoom` is conflicting with the predicate `inBedRoom`, since they are spatially disjoint.
- If p_i is overlapping with p_j, then $n_i \sqcap n_j = n_k$, where $n_k.l$ is the overlapped predicate. The overlapping is defined as: two context predicates are overlapping if they can be satisfied at the same time by certain sensor data, but there exists sensor data that satisfies one of them but not the other. For example in Figure 2 (c), the context predicate `inLivingRoom` overlaps with another predicate `inDiningRoom`, since they share a common location – the foyer. If two context predicates are overlapping, the more specific node under their corresponding nodes is the node with their overlapped logical description; for example, `inFoyer`.

With these predicates and their semantics, a situation lattice can be completed [3]. A built situation lattice organises the context predicates and their combinations in an ordered hierarchy. Each of them is labelled with activities and their occurrence ratio: when a combination of context predicate is satisfied by the current sensor data, then any associated activity can be occurring with a certain likelihood. For example, a set of activities {"watching TV" 0.8, "reading" 0.7} could be associated on the node `inLivingRoom` $\wedge$ `elecCurrentInLivingRoomOn`, which means that when these predicates are satisfied, then the subject could be watching TV with the probability 0.8 and could be reading with the probability 0.7. Figure 3 represents part of a situation lattice built on the PlaceLab data set.

In the lattice, the basic semantics between context predicates are automatically preserved on nodes that contain them. Given two predicates at different levels of granularity; for example, their preliminary nodes $n_i \sqsubseteq n_j$, when combined with another node n_k whose predicate has no relationship with these two, then their compound nodes[1] $n_i \otimes n_k$ and $n_j \otimes n_k$ preserve different levels of granularity: $n_i \otimes n_k \sqsubseteq n_j \otimes n_k$. If two predicates conflict, then any two compound nodes that contains each of them will conflict with each other.

IV. Querying across Multiple Levels of Abstraction

A situation lattice can be used to observe human activities from the perspective of sensors, which can help developers to understand activity patterns of the user and thus spot abnormal activity (like "passing out"). To achieve this, the lattice supports a broad range of queries that involve characteristic sensor data in single or different types and across different levels of abstraction, such as "*what does the subject usually do in the living room*?", "*what does he do at 21:00 in the living room with its light on and its current flow off*?", and even "*what does he do in the foyer (part of the living room)*?".

Situation lattices facilitate answering these queries. The process is similar to the activity recognition process in the lattice [3]. In a query, characteristic sensor data are mapped to a preliminary node, and a compound node whose logical description matches all the sensor data will be located. Developers can check the activities that are associated on the node. For example in the PlaceLab data set, the query "*what does the subject usually do in the living room*?" can be simply answered by checking the preliminary node whose context predicate is `inLivingRoom`. The result will contain activities and their occurrence probabilities; that is, {"watching TV" 0.5, "eating" 0.46, "using phone" 0.23, "using computer" 0.08, "reading" 0.08, "grooming" 0.14 }. This result provides the insight about the subject's activity pattern: when he is in the living room, he is more likely to watch TV, eat, or use phone, but he does not sleep or prepare a meal there.

Take another example of a query "*what does he do at 21:00 in the living room with its light on and its current flow off*?". This query consists of four parts: a time predicate – `21-22`, a location predicate – `inLivingRoom`, and two current predicates – `lightInLivingRoomOn` and `elecCurrentInLivingRoomOff`. A compound node will be located, which represents the conjunction of these four predicates. Its activities are {"using phone" 0.03, "using computer" 0.002, "reading" 0.007, "eating" 0.005}. Combined with the result on the former query, this result uncovers that the subject is impossible to watch TV when the electrical current in the living room is off.

The situation lattice is an effective tool to observe human activity pattern from the view of sensors by supporting these rich queries. A query with a single type of sensor data or multiples types of sensor data will be easily executed on the lattice. It does not need to re-train the structure or execute any other complicated process, while most machine-learning techniques and rule-based system will do.

[1]Given two nodes n_i and n_j, their compound nodes is labelled as $n_i \otimes n_j$ such that $(n_i \otimes n_j).l = n_i.l \wedge n_j.l$.

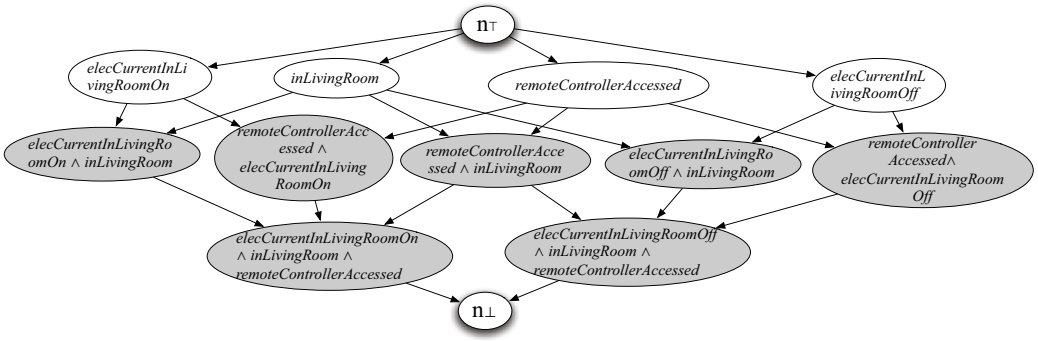

Fig. 3. Part of a situation lattice built on the PlaceLab data set

V. Deriving Specifications and Relationships between Activities

Section IV studies the typical activity recognition process by observing activities from sensor data: what activity does a user usually conduct given the sensor data?. This section will explore activity patterns by observing sensor data from particular situations: how does sensor behave when a user is performing this activity?. A more specific question could be "what time does a user usually prepare a meal?". The activity pattern will be helpful in accumulating experiences on choosing sensors to build a smart home environment, providing feedback on defining activities, and studying semantics between activities.

A. Specification of Activities

A situation lattice supports deriving a specification for an activity. An activity's specification is a logical description that takes context predicates as input and applies logical operators (disjunction, conjunction, and negation) on them. An activity is considered being conducted, if its specification is satisfied by current sensor data.

Given a built situation lattice with N nodes, there exists a set of the most specific nodes (except the bottom node) $N_s \subseteq N$ whose activity set contains a certain activity s. The specification of this activity s is generated by applying the logical operator OR on the logical description of these nodes N_s. That is, a specification of s is $\bigvee_{i=1}^{i=m} n_i.l$, where $\{n_1, n_2, \ldots, n_m\} = N_s$. Algorithm 1 describes the procedure of locating the most specific nodes related to a situation. To check whether a node is the most specific node, we simply check that no other node except the bottom node is more specific than this node. Algorithm 2 applies the OR operator on all the logical descriptions from the most specific node set.

Algorithm 2 works when the size of a situation lattice is small. When the size is relatively large, a specification of an activity can be tedious. For example, the situation lattice built on the PlaceLab data set consists of 27647 nodes, among which 3423 are the most specific nodes. Thus, an activity's specification might contain a disjunction of conjunct context predicates from hundreds of the most specific nodes. Among these nodes, it is possible that a subset of them are all the immediately more specific nodes from a more general node. The specification can be simplified by replacing these subset of nodes with the more general node. For example in Figure 4 (a), for an activity, a subset of its most specific nodes n_1, n_2, $\ldots$, n_i are all the children nodes under the same more general node n_j, then in this activity's specification l, the subset of nodes can be replaced with n_j; while in (b), they cannot since the subset nodes do not cover all the children nodes of n_j.

Algorithm 1: Locating all the most specific nodes for a situation

input : a situation s and a trained lattice $(N, \sqsubseteq)$
output: a set N_s of the most specific nodes that contributes to identify s

```
N_s = ∅
foreach Node n ∈ N do
    if s ∈ n.S then
        IsMostSpecific = TRUE
        foreach Node n' ∈ N do
            if n' ≠ n ∧ n' ≠ n_⊥ ∧ n' ⊑ n then
                IsMostSpecific = FALSE
        if IsMostSpecific then
            N_s.add(n)
    return N_s
```

Algorithm 2: Defining a specification for a situation

input : a set N_s of the most specific nodes that contributes in identifying s
output: a specification of s

```
specification = FALSE
foreach Node n ∈ N_s do
    specification = specification ∨ n.l
return specification
```

Algorithm 3[2] describes the process of refining an activity's specification from its most specific node set.

Algorithm 3: Improvement on defining a specification for a situation

input : a set N_s of the most specific nodes that contributes in identifying s and a trained lattice $(N, \sqsubseteq)$
output: a specification for s

```
repeat
    stop =TRUE
    foreach n ∈ N_s do
        foreach n_p ∈ N such that n !⊑ n_p do
            containsAllChildren =TRUE
            children = ∅
            foreach n' ∈ N such that n' !⊑ n_p do
                if n' ∈ N_s then
                    children.add(n')
                else
                    containsAllChildren =FALSE
                    break
            if containsAllChildren then
                N_s.removeAll(children)
                N_s.add(n_p)
                stop =FALSE
until stop
specification = FALSE
foreach Node n ∈ N_s do
    specification = specification ∨ n.l
return specification
```

Algorithm 3 simplifies the expression of a situation's specification greatly with increased complexity $O(n^3)$, where n is the size of N_s. We use these algorithms on the situation lattice of the PlaceLab data set, and derive specifications of activities. Some of the activities have a relatively small number of nodes in that they have a tractable activity pattern. For example of the activity "hygiene", its specification is listed as follows[3]:

```
hygiene =(inPowderRoom ∧ 18-19
        ∧waterInPowderRoomOff
        ∧lightInPowderRoomOff
        ∧ nothingInPowderRoomAccessed)
        ∨(inBedroom ∧ 23-24
        waterInBathroomOff ∧
        lightInBathroomOn))
```

[2]In this algorithm, $n! \sqsubseteq n_p$ means that n is the immediately child node under n_p.

[3]As stated in [3], the PlaceLab involves a couple of subjects and non person-specific sensors, which introduces extra noise in the data set. Also only the activities on the male subject have been annotated. To compromise the noise, a most specific node is chosen for an activity only if the activity ratio on this node is beyond a certain threshold. Therefore, some nodes could be removed from the specification due to the low occurrence ratio of the activity on them.

Some of the activities are associated with a large number of the most specific nodes. For example, "using phone" has 245 most specific nodes, since the subject could use phone anywhere at any time. Part of its specification is listed as follows:

```
using phone =inLivingRoom ∧
            ((elecCurrentInLivingRoomOff∧
            nothingInLivingRoomAccessed ∧
            16-18) ∨ (lightInLivingRoomOn
            ∧ (((18-19 ∨ 20-21)
            ∧ remoteControlAccessed)
            ∨ (nothingInLivingRoomAccessed
            ∧ 18-20))))
            ...
```

By examining the specifications of the activities, developers can have an intuition on what sensors are useful in recognising an activity. For the activity "hygiene", the most effective sensor is probably a positioning sensor that could detect the subject in the powder room or the bath room. Also the number of most specific nodes on each activity can reflect the accuracy of its descriptive definition; that is, a large number of nodes suggests that either this activity occurs frequently (for example, the activity "using a computer" has 395 most specific nodes, and it occurs in 20.14 hours, 42.9% of the time through the data set.) or this activity is less tractable. Under the latter circumstance, developers might need to refine its descriptive definitions by removing trivial activities (like styling hair) or split the activity into sub activities.

B. Semantics of Activities

Based on specifications of activities, we can explore semantic relationships between activities: *type-of* and *conflicting*. These two relationships can be inferred from the specifications of activities based on their correspondingly most specific node sets. One activity s_i is a type of another activity s_j, if when a subject is conducting s_i he is also considered conducting s_j. It requires that for any of the most specific nodes on s_i, there exists at lease one most specific nodes on s_j such that the node on s_i is more specific than the node on s_j (as presented in Definition 1). Thus if a specification of s_i is satisfied, then the specification of s_j will be satisfied as well.

Definition 1: s_i is a type of s_j, iff $\forall n_k \in N_{s_i}, \exists n'_k \in N_{s_j}$, $n_k \sqsubseteq n'_k$.

One activity s_i conflicts with another activity s_j, if it is impossible for a subject to conduct both activities at the same time. It requires that any of the most specific nodes associated with s_i conflicts with any of the most specific nodes associated with s_j (as presented in Definition 2).

Definition 2: s_i conflicts with s_j, if $\forall n_k \in N_{s_i}, n_l \in N_{s_j}$, $n_k \sqcap n_l = n_\perp$.

If two situations share some of the same most specific nodes, then they are likely to occur at the same time when any of these nodes are activated. However, this leads to another question: if one of these nodes are activated, it is possible that both the situations occur, or one of them is occurring while the

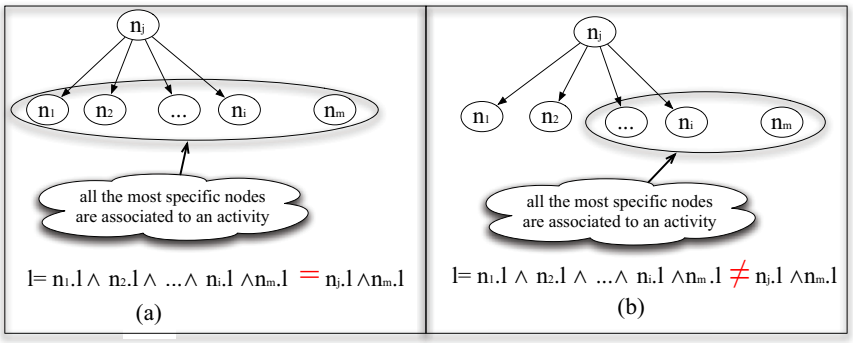

Fig. 4. Examples of refining specification of situations

other is not. For example in the PlaceLab data set, "watching TV" and "eating" can share the same most specific node, whose context predicates are `19-20 ∧ inLivingRoom`. When this node is activated, the subject can be "watching TV" and "eating" simultaneously, or he can be either "watching TV" or "eating". This issue is related to the discernability of sensors in precisely identifying situations, which has been covered in more details in [3].

We apply the above two definitions on the lattice of the PlaceLab data set. There are no specialisation relationships between the activities. Since the location predicates on each individual rooms conflict with each other, the activities that occur in different rooms should conflict, such as "dishwashing" and "hygiene". Due to the imprecision of the positioning sensors, the activities that should be conflicting are not inferred as conflicting; for example, "watching TV" should conflict with "meal preparation".

VI. Conclusions

This paper has demonstrated that the situation lattice lays the foundations of studying human behaviour pattern. The lattice supports a range of rich queries, which is useful in observing typical activities of a subject such as to spot abnormal activities. The specifications of activities derived from the lattice help to uncover the activity pattern of the subject and indicate the sensors (or sensor data) that contribute to identifying an activity. Based on the specification and semantics of predicates, developers can also extract semantic relationships (that is, type-of and conflicting) between different activities. The discovered knowledge will provide guidance for other developers on choosing sensors and defining activities in building a smart home environment. In the future, we will conduct a full evaluation on the generality and effectiveness of using situation lattices in studying human behaviour patterns.

Acknowledgment

This work is partially supported by Science Foundation Ireland under grant numbers 07/CE/1147 "Clarity, the centre for sensor web technologies" and 05/RFP/CMS0062 "Towards a semantics of pervasive computing".

References

[1] H. Zheng, H. Wang, and N. Black, "Human activity detection in smart home environment with self-adaptive neural networks," in *Proceedings of IEEE International Conference on Networking, Sensing and Control (ICNSC 2008)*, April 2008, pp. 1505–1510.

[2] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. S. Intille, "A long-term evaluation of sensing modalities for activity recognition," in *Proceedings of Ubicomp 2007*, Innsbruck, Austria, September 2007, pp. 483–500.

[3] J. Ye, L. Coyle, S. Dobson, and P. Nixon, "Using situation lattices in sensor analysis," in *Proceedings of PerCom 2009*, Mar. 2009, pp. 1–11.

[4] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Proceedings of the second International Conference on Pervasive Computing*, Vienna, Austria, Apr. 2004, pp. 1–17.

[5] K. Kunze and P. Lukowicz, "Dealing with sensor displacement in motion-based onbody activity recognition systems," in *Proceedings of UbiComp '08*. ACM, 2008, pp. 20–29.

[6] H. H. Bui, S. Venkatesh, and G. West, "Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model," *Hidden Markov models: applications in computer vision*, pp. 177–196, 2002.

[7] T. van Kasteren and B. Krose, "Bayesian activity recognition in residence for elders," in *Proceedings of the third IET International Conference on Intelligent Environments*, Sep. 2007, pp. 209–212.

[8] M. K. Hasan, H. A. Rubaiyeat, Y.-K. Lee, and S. Lee, "A hmm for activity recognition," in *Proceedings of ICACT 2008*, vol. 1, 2008, pp. 843–846.

[9] D. Minnen, T. Starner, J. Ward, P. Lukowicz, and G. Troster, "Recognizing and discovering human actions from on-body sensor data," in *Proceedings of ICME 2005*, July 2005, pp. 1545–1548.

[10] L. Liao, D. Fox, and H. Kautz, "Extracting places and activities from gps traces using hierarchical conditional random fields," *International Journal on Robotics Research*, vol. 26, no. 1, pp. 119–134, 2007.

[11] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *Proceedings of UbiComp '08*. New York, NY, USA: ACM, Sep. 2008, pp. 1–9.

[12] J. Modayil, T. Bai, and H. Kautz, "Improving the recognition of interleaved activities," in *Proceedings of UbiComp '08*, 2008, pp. 40–43.

[13] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[14] J. Ye, L. Coyle, S. Dobson, and P. Nixon, "Representing and manipulating situation hierarchies using situation lattices," *Revue d'Intelligence Artificielle*, vol. 22, no. 5, pp. 647–667, 2008.

[15] J. Ye, S. McKeever, L. Coyle, S. Neely, and S. Dobson, "Resolving uncertainty in context integration and abstraction," in *Proceedings of the international conference on Pervasive Services*. New York, NY, USA: ACM, July 2008, pp. 131–140.