A Knowledge-Driven Tool for Automatic Activity Dataset Annotation

Gorka Azkune[1], Aitor Almeida[1], Diego López-de-Ipiña[1], and Liming Chen[2]

[1] Deusto Institute of Technology – DeustoTech, University of Deusto, Bilbao, Spain
{gorka.azkune, aitor.almeida, dipina}@deusto.es
[2] De Montfort University, Leicester, United Kingdom
liming.chen@dmu.ac.uk

Abstract. Human activity recognition has become a very important research topic, due to its multiple applications in areas such as pervasive computing, surveillance, context-aware computing, ambient assistive living or social robotics. For activity recognition approaches to be properly developed and tested, annotated datasets are a key resource. However, few research works deal with activity annotation methods. In this paper, we describe a knowledge-driven approach to annotate activity datasets automatically. Minimal activity models have to be provided to the tool, which uses a novel algorithm to annotate datasets. Minimal activity models specify action patterns. Those actions are directly linked to sensor activations, which can appear in the dataset in varied orders and with interleaved actions that are not in the pattern itself. The presented algorithm finds those patterns and annotates activities accordingly. Obtained results confirm the reliability and robustness of the approach in several experiments involving noisy and changing activity executions.

Keywords: Activity recognition · Knowledge-Driven · Activity Annotation

1 Introduction

Being able to recognize what human beings are doing in their daily life can open the door to a lot of possibilities in diverse areas. Technology is becoming more and more human-centred in order to provide personalized services. In other words, technological services have to adapt to human users and not the other way around.

Following that trend, human activity recognition becomes a natural enabler to such adaptive technologies. Let us consider the case where an intelligent assistive system has to promote a healthy lifestyle. For such a system it is mandatory to recognize the activities carried out by human users in order to analyse what kinds of recommendations would be better to improve the health of users. Many other examples can be found in domains like pervasive and mobile computing, surveillance-based security, context-aware computing or social robotics.

The literature shows multiple ways of approaching activity recognition, based on the sensors to be used and techniques adopted. A good survey by Chen et al. can be found in [1]. There are two main currents inside the activity recognition scientific community: the data-driven current and the knowledge-driven current. It turns out that for both activity recognition currents annotated datasets are needed to ensure a successful development and/or evaluation.

However, the scientific community has already identified a lack of work in methods to get annotated datasets for activity recognition, as noted by Rashidi and Cook in [11]: *"An aspect of activity recognition that has been greatly underexplored is the method used to annotate sample data that the scientist can use to train the activity model"*.

This paper tackles the problem of annotating activity datasets, adopting a knowledge-driven approach. The basic idea is that domain experts have important knowledge about how activities are carried out by users. However, that knowledge is not complete in the sense that every user has different ways to perform activities. The software tool presented in this paper uses expert knowledge to build minimal activity models that contain for each activity, those concepts that always appear in it. For example, to make a coffee every user will use coffee and a liquid container to pour and drink that coffee. Additional actions may be executed by several users, such as adding milk and sugar, but no user will make a coffee without a container and coffee.

A novel algorithm that uses those minimal activity models to annotate activity datasets is presented in this paper, which is called SA^3 (*Semantic Activity Annotation Algorithm*) The input of SA^3 is a time-stamped sensor activation dataset. First, sensor activations are transformed to actions. For example, if a contact sensor installed in a coffee-cup is triggered, the algorithm transforms it to the action *hasContainer(cup)*. Minimal activity models are expressed as a sequence of actions to perform activities. In the second step, the algorithm iterates through the actions dataset and finds all the occurrences of minimal activity models. In this step the algorithm finds all action patterns regardless of the order of actions and the occurrence of any interleaved action. In the third step the set of non-overlapping activity patterns is found, which is used to annotate the original dataset.

The three-step annotation algorithm has been extensively tested. To make those tests more reliable, a synthetic dataset generator has been developed. The synthetic dataset generator allows simulating an arbitrary number of days for a user, specifying the activities performed, several ways to perform those activities and sensor noise. The tool generates a proper ground-truth to test SA^3 in many diverse ways and scenarios. The results obtained are very encouraging.

The paper is structured as follows: in Section 2, the literature about activity dataset annotation will be analysed. In Section 3, the details of the annotation algorithm will be described. Afterwards, Section 4 will explain briefly the usage of the synthetic dataset generator and present all the experiments performed and results obtained. Those results will be discussed in Section 5, to finish the paper with some conclusions and future work in Section 6.

2 Related Work

Dataset annotation for activity recognition is an under-explored area. Most of the researchers rely on manually annotated datasets. Many research works show experimental methodologies where participants have to manually annotate the activities they are performing. Good examples can be found in [12], [10] and [7]. Nevertheless, such system does not allow participants to behave naturally, interfering the experiment itself. In addition, correct annotation cannot be assumed when working with cognitively impaired people.

There are some other research works where the experimenters prepare a script with the activities that participants should perform, as can be seen in [2], [8], [3] and [4]. The problem of such an approach is again that participants cannot behave naturally. In the case of [2], participants cannot even decide how to perform the activities. Those situations are far from being real-world situations.

To finish up with manual annotation methods, Wren et al. show in [13] experiments where an expert had to go through raw sensor data to find activities and annotate them. In general, such an approach is very time consuming and results depend on the ability of the expert.

Although not many, there are some alternative methods to manual annotation. For instance, Kasteren and Noulas present in [6] a novel method that implies the use of a bluetooth headset equipped with speakers in order to capture the voice of the participant. While performing an activity, the participant has to name the activity itself. Speech recognition algorithms are used to automatically annotate the activity. This method is much more comfortable than manual annotation from the point of view of the participant. However, it is invasive and presents the same problems with cognitively impaired people.

A different approach is presented in [5] by Huynh et al. They provide three annotation methods. The first one is a mobile phone application that asks several questions while the participant is performing activities. Depending on the answers, the system decides the label that corresponds to the ongoing activity. The second method is a hand-written diary, i.e. manual annotation. And the third system uses the mobile phone again, but in a different way. Another application has been developed to take pictures regularly. Those pictures are used by experts to annotate the activities. Authors admit that after several experiments where participants could choose among the three options, the vast majority tend to use the manual annotation system.

Finally, Rashidi and Cook show in [11] an activity recognition approach that does not need any annotated dataset. They use an unsupervised approach that discovers frequent patterns in raw sensor data. Models are trained to be able to recognize the extracted patterns, so they do not require any annotated activity. However, the activity models they generate do not have any semantic meaning. Hence, a manual process should be run afterwards to understand activity models and assign a semantic label to them. In that sense, the work of Rashidi and Cook cannot be considered a dataset annotator. What they present is an activity recognition system that overcomes the problem of not having an annotated activity dataset.

The work presented in this paper is novel respect to the state of the art, since an algorithm which uses minimal expert knowledge is developed to automatically annotate activity datasets. Notice that our work cannot be considered as an activity recognition approach, since it works offline, once a whole dataset of activities has been collected. In that sense, the requirements and constraints of activity annotation are softer than real-time activity recognition.

3 The Approach to Automatic Annotation

The annotation algorithm presented in this paper is inspired in knowledge-driven activity recognition methods, specially in [2]. Three important concepts from that work are extracted and used for the presented approach:

- **Domain knowledge allows recognizing activities reliably:** in contrast with data-driven techniques, which use intensively annotated datasets in order to learn activity models for recognition, knowledge-driven techniques use expert domain knowledge to model activities. Those models can be reliably used for activity recognition
- **Sensor activations are linked to user-object interactions:** if a contact sensor installed in a cup changes its state, the object is assumed to be used for an activity. Sensors' information is modelled to allow transformations from sensor activations to user-object interactions or **actions**. Following with the contact sensor of the cup, whenever an activation is received, it is interpreted as an action, which can be named as *hasContainer(cup)*
- **Activities can be modelled as sequences of actions:** even though richer information can be provided as location and time, to keep activity models as simple as possible, we only consider necessary actions. As such, the *MakeCoffee* activity can be modelled as the action sequence {*hasContainer(x), hasCoffee(y)*}.

Real-time activity recognition needs complete activity models to have a reliable recognition performance. However, providing complete activity models is very complicated, since each user executes varied action sequences to perform the same activity. For example, to make a coffee, some users may use milk and sugar, while others may add only some cream. But making coffee will always imply using coffee and having a container, i.e. the action sequence {*hasContainer(x), hasCoffee(y)*}. This prior knowledge is used in SA^3 for activity annotation.

Activity annotation and recognition is not the same thing. Activity annotation can make use of the whole dataset offline, with no time restrictions. On the other hand, activity recognition is required to work while activities are being performed, so only past sensor activations can be used. This key difference makes feasible using incomplete activity models to annotate activities, in contrast with the real-time recognition problem.

3.1 Basic Concepts

- **Sensor activations (SA):** it is assumed that environments are equipped with sensors that can collect information about user-object interactions. Whenever those sensors change their states, a sensor activation is collected. Sensor activations are then composed by a sensor identifier and a time-stamp

$$SA = \{time\text{-}stamp, sensorID\}$$

- **Sensor dataset:** a time-ordered sequence of sensor activations
- **Actions:** actions are the primitives of activities and are directly linked to sensor activations. The link between sensor activations and actions is manually established. Tables are used where several sensor activations are linked to an action. For example, *cupObjSensor* and *glassObjSensor* are linked to the action *hasContainer*. A sensor activation can only be linked to one single action. The transformation function is defined as:

$$Trans(SA) : SA \rightarrow Action$$

- **Action dataset:** a time-ordered sequence of actions
- **Minimal activity models:** activity models are sequences of actions defined by a domain expert. Minimal activity models refer to the minimal number of necessary actions to perform an activity. The objective of such models is to represent incomplete but generic activity models which provide enough information to detect activities. Minimal activity models also have an estimation of the maximum duration of the activity based on a heuristic:

$$Activity_n = \{action_a, action_b, \ldots, max\_duration\}$$

3.2 SA^3: Semantic Activity Annotation Algorithm

Having as inputs a sensor dataset, sensor-action transformation functions and minimal activity models, the objective of SA^3 is to generate an annotated sensor dataset, where each sensor activation will have an activity label or the special label *None*, which represents lack of known activity. Single user - single activity scenarios are considered, so no interleaved activities will be in the sensor dataset. Considering those constraints, a three-step algorithm has been designed:

1. **Sensor-action transformation step:** the first step takes as input the sensor dataset and transformation functions to generate an action dataset. For each sensor activation in the sensor dataset, the transformation function is applied to obtain the corresponding action. In consequence, a sensor sequence such as:

$$\{cupObjSensor, whiteSugarSensor, skimmedMilkSensor\}$$

   will be transformed to:

$$\{hasContainer(cup), hasFlavour(white\text{-}sugar), hasMilk(skimmed\text{-}milk)\}$$

   Since only actions and not objects are relevant for activity annotation, *hasContainer(cup)* will be used as *hasContainer*

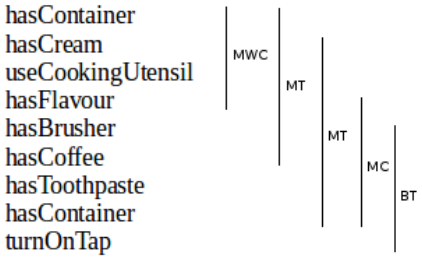

Fig. 1. Illustrative example of the output of the step 2 of SA^3. MWC refers to *MakeWhippedCream*, MT to *MakeTiramisu*, MC to *MakeCoffee* and BT to *BrushTeeth*

2. **Activity sequence finding step:** all the occurrences of minimal activity models are found iterating on the action dataset. For an action pertaining to a minimal activity model following actions are searched. Those actions are considered to describe the activity if two criteria are fulfilled:
   (a) *Completion criterion*: the action sequence has to contain all the actions of the corresponding minimal activity model
   (b) *Duration criterion*: the duration of the action sequence has to be smaller than the duration estimation of the corresponding minimal activity model

   Depending on the activity models, the actions dataset and noise levels, detected activities can overlap each other. Here is an illustrative example, where time-stamps are ignored since duration criterion is satisfied. Imagine we define minimal activity models for *MakeCoffee, MakeTiramisu, MakeWhippedCream* and *BrushTeeth*, where:

$$MakeCoffee = \{hasCoffee, hasContainer, hasFlavour\}$$
$$MakeTiramisu = \{hasCream, hasContainer, hasCoffee\}$$
$$MakeWhippedCream = \{hasFlavour, hasContainer, hasCream\}$$
$$BrushTeeth = \{hasBrusher, hasToothPaste, turnOnTap\}$$

   Let us consider the following action sequence from the action dataset:

$$\{hasContainer, hasCream, useCookingUtensil, hasFlavour, hasBrusher, hasCoffee, hasToothPaste, hasContainer, turnOnTap\}$$

   Figure 1 shows all the activities found applying the completion and duration criterion. Activities overlap each other, because there are several actions that belong to several activities. Notice also that there are some actions that are not in any activity model, which is totally feasible for our approach.
3. **Correct activity sequence fitting step:** having the overlapping activities, the objective of this step is to find the maximum number of activities that do not overlap. This heuristic is derived from the fact that only none interleaved

activities are considered. Applying it, the example of Figure 1 can be solved appropriately. The solution found by the algorithm is that there are only two activities in that action sequence:

$$MakeWhippedCream = \{hasContainer, hasCream, useCookingUtensil, hasFlavour\}$$
$$BrushTeeth = \{hasBrusher, hasCoffee, hasToothPaste, hasContainer, turnOnTap\}$$

Hence, *hasCoffee* is probably due to a faulty sensor activation and *hasContainer* may refer to a glass used in the bathroom to rinse the mouth.

This three-step algorithm is depicted as pseudo-code in Algorithm 1. It has been designed to work with noisy sensor activations, varying order for activity executions and sensor activations that do not belong to any activity model. That flexibility allows using the tool in many different datasets. Section 4 contains many cases to show the performance of the algorithm in several demanding situations.

Algorithm 1 SA^3 algorithm for semantic activity annotation

Require: sensor_dataset, transformation_function, minimal_activity_models
Ensure: annotated_dataset
  $action\_dataset \leftarrow apply\_transform\_function(sensor\_dataset, transformation\_function)$
  for all $action \in action\_dataset$ **do**
    if $action \in minimal\_activity\_models$ **then**
      $activities \leftarrow obtain\_activities(action, minimal\_activity\_models)$
    end if
    for all $activity \in activities$ **do**
      // Use duration and completion criteria
      $detected\_activities \leftarrow find\_proper\_activities(minimal\_activity\_models)$
    end for
  end for
  $annotated\_dataset \leftarrow find\_non\_overlapping\_activities(detetected\_activities)$
  return *annotated_dataset*

4 Evaluation

4.1 Synthetic Dataset Generator

To evaluate the SA^3 algorithm, a ground-truth was required. For that purpose, a synthetic dataset generator algorithm was developed, inspired by the tool described by Okeyo et al. in [9]. The tool offers the following functionalities:

- Define activation patterns: activation patterns are sensor activation sequences linked to an activity. Each activity can have different activation patterns with associated probabilities. Typical time intervals between two successive activations are also provided. Synthetic dataset generator uses Gaussian time intervals, simulating activation patterns more realistically

– Define activity patterns: activity patterns specify sequences of activities that occur in a given time interval. Time intervals between two successive activities are provided. Activity patterns may model that between 7 AM and 9 AM a person usually makes coffee and brushes teeth. Additionally, alterations can also be modelled, where an occurrence probability is assigned to an activity for a given interval. This allows us to model the probability of a person watching TV between 6 PM and 8 PM
– Sensor noise: two kinds of noise can be modelled: (i) missing sensors, where a sensor that should be activated fails and (ii) positive noise, where a sensor that should not be activated, suddenly gets activated. For all sensors, both noise modalities can be provided with probabilistic models

Activation patterns contain different ways of performing activities, while activity patterns capture typical activities performed by a user everyday. Synthetic dataset generator uses probabilistic mechanisms to simulate sensor activations for a given number of days. Generated datasets are already labelled to work as ground-truth.

4.2 Experimental Set-up and Results

Four experimental set-ups were prepared to evaluate the SA^3 algorithm:

Ideal scenario There is no noise in this set-up. Seven activities are defined, with two variations for each activity. Three typical days are modelled through activity patterns. Generated datasets contain around 900 sensor activations. Five tests that simulate 30 days each, were run using five datasets generated from the same script. Overall results for the seven activities are depicted in Table 1. Success rate for all activities is 100%. No false positives or negatives were detected.

Table 1. Results for *Ideal scenario*

Activity	Instances	Correctly annotated	Total annotated
MakeChocolate	150	150	150
WatchTelevision	133	133	133
BrushTeeth	411	411	411
WashHands	150	150	150
MakePasta	178	178	178
ReadBook	130	130	130
MakeCoffee	67	67	67

Missing sensor noise scenario The same seven activities are used in this set-up. Three typical days are modelled and 60 days are simulated. A typical

dataset contains around 1900 sensor activations. Two activities, *MakeChocolate* and *BrushTeeth*, have a probability of being performed with a missing sensor activation which is used in their respective minimal activity models. Five experiments are run, where failing probabilities for both sensors are set to 0.01, 0.02, 0.05, 0.07 and 0.1, to see the behaviour of SA^3 in presence of increasing noise. Results show that the performance for the rest of activities is the same as the previous experiment. However, as noise increases, correctly annotated activities for *MakeChocolate* and *BrushTeeth* decrease, as it can be seen in Table 2.

Table 2. Results for *Missing sensor noise scenario*

Activity	**Missing sensor probability**				
	0.01	0.02	0.05	0.07	0.1
MakeChocolate	59/60	58/60	58/60	54/60	53/60
BrushTeeth	162/163	158/161	146/159	150/161	145/157

Positive sensor noise scenario The same set-up as previous scenario, but containing positive sensor noise and no missing activations. Some sensor activations that are linked to minimal activity models are assigned a probability to generate faulty activations in any time of the day. Some other faulty sensors that are not linked to activity models are also used. To sum up, 7 out of 9 noisy actions are used in minimal activity models. Once again, five experiments are run, where activation probability is increased. Results are shown in Table 3, where for each activity and noise probability, three parameters are given: instances of that activity in the dataset, correctly annotated activities and total annotated activities.

Table 3. Results for *Positive sensor noise scenario*; I: instances; C: correctly annotated; T: total annotated

Activity	**Missing sensor probability**														
	0.05			0.1			0.15			0.2			0.25		
	I	C	T	I	C	T	I	C	T	I	C	T	I	C	T
MakeChocolate	60	60	60	60	58	59	60	59	59	60	59	60	60	59	59
WatchTelevision	55	55	55	54	53	54	54	54	54	47	47	47	60	60	60
BrushTeeth	162	162	162	162	161	161	162	162	162	161	160	160	168	166	166
WashHands	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
MakePasta	66	66	66	65	65	65	64	63	64	63	63	63	77	76	77
ReadBook	55	55	55	49	49	50	50	50	51	52	52	53	50	50	50
MakeCoffee	24	24	24	23	23	24	22	22	26	22	22	32	29	29	51

Challenging activity models scenario In this scenario, seven activities are used again, but three of them contain the same actions to be described. Concretely:

$$MakeCoffee = \{hasContainer, hasCoffee, hasFlavour\}$$
$$MakeTiramisu = \{hasContainer, hasCoffee, hasCream\}$$
$$MakeWhippedCream = \{hasFlavour, hasCream, hasContainer\}$$

The other four activities remain the same as in previous examples. The aim of the current set-up is to test the validity of the algorithm with activities that contain very similar action sequences. Five experiments were run in ideal conditions, where each experiment simulates 30 days. Combined results are depicted in Table 4. For all activities, 100% success rate has been achieved, with 0 false positives and false negatives.

Table 4. Results for *Challenging activity models scenario*

Activity	Instances	Correctly annotated	Total annotated
MakeCoffee	222	222	222
MakeTiramisu	120	120	120
MakeWhippedCream	72	72	72
WashHands	150	150	150
BrushTeeth	420	420	420
ReadBook	127	127	127
WatchTelevision	151	151	151

5 Discussion

Results shown in Section 4 show that SA^3 can handle varying order of actions in activity performance without any problem. All four set-ups contain different activation patterns, where actions are executed in different orders. For example, for the activity *MakeChocolate*, two activation patterns are considered: (i) {*hasContainer*, *hasMilk*, *useCookingAppliance*, *hasChocolate*} and (ii) {*useCookingAppliance*, *useCookingUtensil*, *hasMilk*, *hasChocolate*, *hasContainer*}. To define the minimal activity model, only two actions are used: *hasContainer* and *hasChocolate*. As the noiseless scenario shows, the success rate of the algorithm for that activity is 100%. Notice that activation patterns used for each activity contain more actions that are not used in minimal activity models. Hence, SA^3 can also handle non-considered actions.

As far as noisy scenarios regard, results suggest that missing sensor activations cannot be properly handled by the algorithm, when those activations are linked to actions that are used in minimal activity models. The non-captured activities are proportional to the missing activation probability of those actions.

That is due to the completeness criterion used in the second step of SA^3, as explained in Section 3. We believe this error can be mitigated, using completion metrics, but as minimal activity models contain very few actions (2-3), those metrics do not seem very useful. Moreover, user-object interaction recognition information gathered by Chen et al. in [2], show that contact sensors, for instance, have a missing probability of around 2%. Hence, due to the nature of minimal activity models and small missing errors, developing special mechanisms that deal with missing sensor activations is not crucial.

For positive noise, the scenario is different. SA^3 has shown to perform reliably even with high levels of noise. For example, even having certain sensors with faulty activation probabilities of 0.25 per hour, average success rate was 99%. Results show an interesting case: *MakeCoffee* activity shows an increasing rate of false positives, scaling up to 76% (look at Table 3). However, none other activities show this behaviour. That is due to the model used for *MakeCoffee*, which only contains *hasContainer* and *hasCoffee* actions. Both actions are involved in the noise generation, hence the probability of getting false activation patterns of those two actions is very high. Overall, results obtained in noisy scenarios can be considered as very good.

Finally, a test involving challenging activity models showed that as far as activity models are unique, SA^3 does not have any problem finding them. A success rate of 100% was achieved under those conditions, without any noise.

In conclusion, considering usual noise levels described in [2], where missing sensor activation rates are around 6% (including poorly behaving pressure sensors), SA^3 showed to be able to annotate very accurately the activities in a time-stamped sensor dataset.

6 Conclusions and Future Work

A novel algorithm for activity dataset annotation was presented in this paper. SA^3 is a knowledge-driven algorithm, which uses sensor-action transformations and minimal activity models as prior expert domain knowledge, to annotate time-stamped sensor datasets.

The results of the experiments described in Section 4 and discussed in Section 5, show that SA^3 can handle varying order of sensors, incomplete activity models and several forms of noise. The main weakness of the algorithm appears when missing sensor activations occur for actions that are used to describe an activity. However, due to the low missing activation rates reported by previous research, the problem does not compromise the performance of the algorithm.

We believe that SA^3 can be a very useful tool for those working on activity recognition. Activity annotation is accurate, as far as sensor-action transformations and minimal activity models can be defined. SA^3 can be a good complement to a human annotator, who only would have to review the results obtained by the automatic tool. In the experiments we carried out, it was easy to identify some activities that were not captured by SA^3. It was also easy to discard false

positive activities emerging from sensor noise. From our experience, SA^3 makes activity annotation much easier and faster.

As future work, it would be very useful to test SA^3 with publicly available real-world activity datasets. Additionally, further research can be done to take advantage of frequent behaviours of users to mitigate more the sensor noise, specially for missing activations. The idea would be to add another step in the algorithm in order to find frequent activity sequences. Afterwards, that information could be used to analyse actions around activities that are in frequent activity patterns. That way, even having noisy scenarios, better results could be obtained for activity annotation.

References

1. L Chen, J Hoey, CD Nugent, DJ Cook, and Z Yu. Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics-Part C*, 42(6):790–808, 2012.
2. L Chen, CD Nugent, and H Wang. A knowledge-driven approach to activity recognition in smart homes. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):961–974, 2012.
3. DJ Cook and M Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 48(5):480, 2009.
4. T Gu, Z Wu, X Tao, HK Pung, and J Lu. epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In IEEE, editor, *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9, 2009.
5. T Huynh, M Fritz, and B Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th International conference on Ubiquitous computing*, pages 10–19. ACM, 2008.
6. T Van Kasteren and A Noulas. Accurate activity recognition in a home setting. In *Proceedings of the 10th International conference on Ubiquitous computing*, pages 1–9, 2008.
7. L Liao, D Fox, and H Kautz. Location-based activity recognition. *Advances in Neural Information Processing Systems*, 18:787, 2006.
8. U Maurer and A Smailagic. Activity recognition and monitoring using multiple sensors on different body positions. In *In Wearable and Implantable Body Sensor Networks*, pages 4–116, 2006.
9. George Okeyo, Liming Chen, Hui Wang, and Roy Sterritt. Dynamic sensor data segmentation for real-time knowledge-driven activity recognition. *Pervasive and Mobile Computing*, December 2012.
10. M Philipose and KP Fishkin. Inferring activities from interactions with objects. *Pervasive Computing*, 3(4):50–57, 2004.
11. P Rashidi and DJ Cook. Discovering activities to recognize and track in a smart environment. *Knowledge and Data Engineering, IEEE Transactions on*, 23(4):527–539, 2011.
12. EM Tapia, SS Intille, and K Larson. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. *Pervasive computing*, 3001:158–175, 2004.
13. CR Wren and EM Tapia. Toward scalable activity recognition for sensor networks. In *Location-and context-awareness*, pages 168–185. Springer Berlin Heidelberg, 2006.