Hierarchical Activity Recognition Using Automatically Clustered Actions

Tim L.M. van Kasteren[1], Gwenn Englebienne[2], and Ben J.A. Kröse[2]

[1] Department of Computer Engineering
Boğaziçi University, Istanbul, Turkey
[2] Intelligent Systems Lab Amsterdam
University of Amsterdam, The Netherlands
tim0306@gmail.com

Abstract. The automatic recognition of human activities such as cooking, showering and sleeping allows many potential applications in the area of ambient intelligence. In this paper we show that using a hierarchical structure to model the activities from sensor data can be very beneficial for the recognition performance of the model. We present a two-layer hierarchical model in which activities consist of a sequence of actions. During training, sensor data is automatically clustered into clusters of actions that best fit to the data, so that sensor data only has to be labeled with activities, not actions. Our proposed model is evaluated on three real world datasets and compared to two non-hierarchical temporal probabilistic models. The hierarchical model outperforms the non-hierarchical models in all datasets and does so significantly in two of the three datasets.

Keywords: Hierarchical Models, Activity Recognition, Sensor Networks.

1 Introduction

The automatic recognition of human activities such as cooking, showering and sleeping allows many potential applications in the area of ambient intelligence [10]. In recent years, temporal probabilistic models have been shown to give a good performance in recognizing activities from sensor data [3,9]. However, many of these models assume a direct correlation between activities and the sensor data. Because activities contain a rich hierarchical structure, modeling this hierarchy explicitly might be beneficial for the recognition performance of the model.

In this paper, we use a hierarchy in which we assume that each activity consists of a number of actions. For example, the activity cooking might consist of an action 'cutting vegetables and meats' and an action 'frying them in a pan'. We present a two-layer hierarchical model for activity recognition in which the top layer of the model corresponds to activities and the bottom layer corresponds to actions. Although it is possible to train such a model using data which is annotated with labels of both activities and actions, in this paper, we train

D. Keyson et al. (Eds.): AmI 2011, LNCS 7040, pp. 82–91, 2011.

the model using only labels for the activities. There are two advantages to this approach: 1) Annotating the data becomes significantly less involved when only the activities have to be annotated, 2) We do not force any structure upon the model with respect to the actions, but rather let the model find this structure in the data automatically. The automatic allocation of structure can be considered as a clustering task. The clusters found in the data do not necessarily have to be meaningful clusters that correspond to actual actions that are intuitive to humans. We therefore distinguish between the term 'action clusters' to refer to the actions found through clustering and 'actions' to refer to the actions intuitive to humans. We evaluate our approach by comparing the recognition performance of our hierarchical model to the recognition performance of two non-hierarchical temporal probabilistic models and we do so using three real world datasets.

The remainder of this paper is organized as follows. In Section 2, we compare our approach to related work. Section 3 provides the details of our hierarchical model and its learning and inference algorithms. Section 4 presents the experiments and results and in Section 5 we discuss these results. Finally, in Section 6, we sum up the conclusions.

2 Related Work

In previous work hierarchical models have mainly been applied to video data to recognize activities such as entering and leaving a store [6]. Nguyen et al. compare the performance of a learned hierarchical hidden Markov model (HHMM), a hand-coded HHMM and a conventional hidden Markov model (HMM), the learned HHMM gives the best performance [8]. Duong et al. compare the performance of the HHMM and the hidden semi-Markov model (HSMM). In their work, the HHMM gives very poor performance in the recognition task which, according to the authors, is caused by a poorly estimated transition matrix. They do not explain why the hierarchical model is unable to learn the transition matrix, while the semi-Markov model is able to learn this matrix accurately [2]. In work by Luhr et al., hierarchical models consisting of several layers are hand crafted by closely inspecting the sequence of actions performed by the subject. Their results show that these models perform well in recognizing several cooking related activities [5].

Overall these works confirm the potential of using hierarchical models for activity recognition, however, none of these works involve the recognition of activities from a wireless sensor network data. Our paper contributes by providing experimental results on several real world datasets, consisting of several weeks of data and involving a large number of activities. We provide a systematic comparison between our HHMM and the HMM and HSMM, showing that the use of a hierarchy results in an increase in performance. Furthermore, we compare two ways of modeling observations in a hierarchical model and show which approach is most effective in modeling activities. Finally, our work demonstrates that the automatic clustering of actions leads to accurate activity recognition with a limited need for annotation.

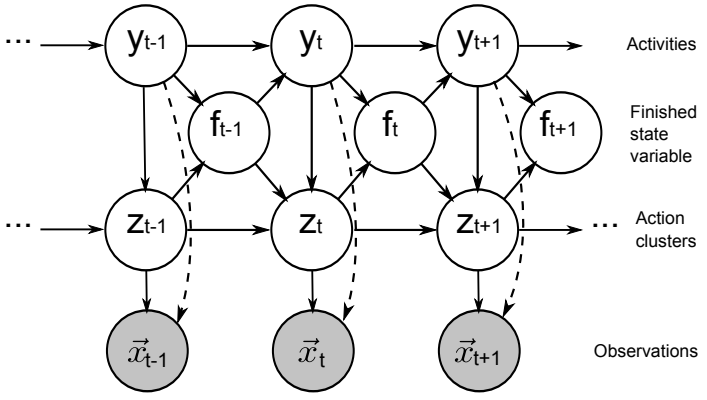


Fig. 1. The graphical representation of a two-layer HHMM. Shaded nodes represent observable variables, the white nodes represent hidden states. The dashed line is an optional dependency relation; we can choose to model the observation probability as $p(\boldsymbol{x}_t \mid y_t, z_t)$ or as $p(\boldsymbol{x}_t \mid z_t)$.

3 Hierarchical Hidden Markov Model

We assume a house in which we perform activity recognition is equipped with a sensor network of binary sensors. The data obtained from the sensors is discretized into T timeslices of length Δt. A single feature value is denoted as x_t^i, indicating the value of feature i at timeslice t, with $x_t^i \in \{0, 1\}$. In a house with N installed sensors, we define a binary observation vector $\boldsymbol{x}_t = (x_t^1, x_t^2, \ldots, x_t^N)^T$. The activity at timeslice t, is denoted with $y_t \in \{1, \ldots, Q\}$ for Q possible states. We use an HHMM to form a mapping between a sequence of observations $\mathbf{x}_{1:T} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$ and a sequence of activities $\mathbf{y}_{1:T} = \{y_1, y_2, \ldots, y_T\}$ for a total of T timeslices.

In this section, we discuss the details of a two-layer hierarchical model for activity recognition and explain the inference and learning algorithms.

3.1 Model Definition

We consider a two-layer hierarchical model for activity recognition. The top layer state variables y_t represent the activities and the bottom layer variables z_t represent the action clusters (Fig. 1). Each activity consists of a sequence of action clusters and the temporal ordering of the action clusters in a sequence can vary between different executions of an activity. Of particular interest to us is the last action cluster that is performed at the end of an activity, because this action cluster signifies the end of a sequence and announces the start of a new sequence of action clusters. We therefore introduce a third variable, the finished state variable f_t, which is used as a binary indicator to indicate the bottom layer has finished its sequence.

We further explain the details of this model by going over all the factors of the joint probability distribution of hidden states and observations given by:

$$p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{f}_{1:T}, \mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{x}_t \mid y_t, z_t) p(y_t \mid y_{t-1}, f_{t-1}) \\ p(z_t \mid z_{t-1}, y_t, f_{t-1}) p(f_t \mid z_t, y_t)$$

where we have defined $p(y_1 \mid y_0, f_0) = p(y_1)$ and $p(z_1 \mid z_0, y_1, f_0) = p(z_1 \mid y_1)$ for the sake of notational simplicity. The entire model consists of a set of parameters $\theta = \{\pi_0, \pi_{1:Q}, A_0, A_{1:Q}, B, \phi\}$. The initial state parameters π and transition parameters A exist for both the top layer and bottom layer states. To distinguish between these two types of parameters, we include a 0 in the subscript to indicate that a parameter is of the top layer and an index of 1 to Q for each of the bottom layer parameters. The distributions of the bottom layer states depend on which top layer state the model is in and so there is a separate set of bottom layer state parameters for each possible top layer state, with Q being the number of top layer states. For example, if the model at one point is in the top state $y_t = k$, then the transition parameter A_k is used for the bottom layer state transitions. We now provide a detailed explanation of each of the factors that make up the joint probability and how they are parameterized.

At the first timeslice, the initial state distribution of the top layer states is represented by a multinomial distribution which is parameterized as $p(y_1 = j) = \pi_0(j)$. This top layer state generates a bottom layer state, also represented by a multinomial distribution and parameterized as $p(z_1 = j \mid y_1 = k) = \pi_k(j)$.

The factor $p(z_t = j \mid z_{t-1} = i, y_t = k, f_{t-1} = f)$ represents the transition probabilities of the bottom layer state variable. These transitions allow us to incorporate the probability of a particular temporal order of action clusters with respect to a given activity. A transition into a new state z_t, depends on the previous bottom layer variable z_{t-1}, the current top layer state variable y_t and the finished state variable f_{t-1}. Two distributions make up this factor, depending on the value of the finished state variable f_{t-1}. Either a new sequence of bottom layer states starts ($f_{t-1} = 1$), or a transition within an existing sequence takes place ($f_{t-1} = 0$).

These two cases can be compactly formulated as:

$$p(z_t = j \mid z_{t-1} = i, y_t = k, f_{t-1} = f) = \begin{cases} A_k(i,j) & \text{if } f = 0 \\ \pi_k(j) & \text{if } f = 1 \end{cases} \tag{1}$$

Transitions of the top layer state variables are represented by the factor $p(y_t = j \mid y_{t-1} = i, f_{t-1} = f)$. Depending on the finished state variable f_{t-1}, the model either transitions into a new state ($f_{t-1} = 1$) or remains in the same state ($f_{t-1} = 0$). These two cases can be compactly formulated as:

$$p(y_t = j \mid y_{t-1} = i, f_{t-1} = f) = \begin{cases} \delta(i,j) & \text{if } f = 0 \\ A_0(i,j) & \text{if } f = 1 \end{cases} \tag{2}$$

where $\delta(i,j)$ is the Kronecker delta function, giving 1 if $i=j$ and 0 otherwise.

The probability of a bottom layer state sequence finishing is represented by the factor $p(f_t = f \mid y_t = j, z_t = l)$. This factor depends on both the bottom layer state z_t and the top layer state y_t. Even though the variable f_t indicates whether z_t is a finishing state, it is important that the distribution is also conditioned on the top layer state y_t. This is because the probability of a particular action cluster being the last action cluster for that activity can differ among activities. The factor is represented using a binomial distribution, parameterized as $p(f_t = f \mid y_t = j, z_t = l) = \phi_f(j,l)$.

Two possible observation models. In the graphical representation of our hierarchical model, shown in Figure 1, there is a dashed line between the top layer state variables y_t and the observation variables $\boldsymbol{x}_t$. This line represents an optional dependency relationship, because we wish to experiment with two types of observation models. If we do take the dependence relation into account, our observation model is represented by the factor $p(\boldsymbol{x}_t \mid y_t, z_t)$. In this model, each combination of top and bottom state values gets its own set of parameters. Alternatively, if we do not include the dependence relation, our observation model is represented by the factor $p(\boldsymbol{x}_t \mid z_t)$. In this case, the observation model is independent of the top layer state variable. Note that in the transition probabilities of the bottom layer state variable described above, there still exists a dependency on the top layer state, regardless of which observation model is used. The same holds for the finished state probability distribution.

Observations are modeled as independent Bernoulli distributions, with each sensor corresponding to one Bernoulli distributions. In case of model 1 the observation probability factorizes as $p(\boldsymbol{x}_t \mid y_t, z_t) = \prod_{n=1}^{N} p(x_n \mid y_t, z_t)$, with $p(x_n \mid y_t = j, z_t = k) = \mu_{jkn}^{x_n}(1-\mu_{jkn})^{(1-x_n)}$. For model 2 we get $p(\boldsymbol{x}_t \mid z_t) = \prod_{n=1}^{N} p(x_n \mid z_t)$ with $p(x_n \mid z_t = k) = \mu_{kn}^{x_n}(1-\mu_{kn})^{(1-x_n)}$, where N is the number of sensors used. Model 1 requires Q times more parameters than Model 2, because of the additional dependency on the top layer states, with Q being the number of top layer state values. The observation parameters are collectively represented by a variable $B = \{\mu_{jkn}\}$ for Model 1 and $B = \{\mu_{kn}\}$ for Model 2.

3.2 Inference and Learning Using a Flattened Implementation

Inference in our proposed HHMM can be done by using the Viterbi algorithm for HMMs [7]. We can flatten our HHMM to a HMM by creating a HMM state for every possible combination of states in the HHMM. Because our model structure is not fully connected some parameters will be shared between states, this means the same set of parameters is used for different states. The use of a flattened implementation allows us to perform inference in linear time.

Parameters are learned iteratively using the Expectation Maximization (EM) algorithm [1]. The E-step consists of using the forward-backward algorithm to calculate the probability distribution $p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{f}_{1:T} \mid \mathbf{x}_{1:T}, \theta)$ given a set of parameters θ. From this distribution, we can calculate the expectation and reestimate the parameters in the M-step. Because no labels are available for the

action clusters, we start with a set of randomly initialized parameters. The procedure of calculating the forward-backward probabilities and reestimating the parameters is repeated until the parameter values converge, indicating a local maximum has been reached.

4 Experiments

Our experiments are aimed at answering three questions: 1) What number of action clusters is needed for modeling activities? 2) Which observation model gives the best performance? 3) How does the performance of our hierarchical model compare to the performance of the HMM and the HSMM, two commonly used models for activity recognition? Our first experiment compares the performance of the hierarchical model using Observation Model 1 to the performance of the HMM and the HSMM. The second experiment makes the same comparison, but uses Observation Model 2. In both experiments, results are given for various number of action clusters. In the remainder of this section we present the details of our experimental setup, we describe the experiments and their results and finally discuss the outcomes.

4.1 Experimental Setup

We used three publicly available datasets and Matlab code used in previous work for the HMM and HSMM [3]. A summary of the relevant details for each dataset can be found in Table 1. The datasets were recorded using wireless sensor networks consisting of simple binary sensors such as reed switches to measure whether doors and cupboards are open or closed; pressure mats to measure sitting on a couch or lying in bed; mercury contacts to detect the movement of objects (e.g. drawers); passive infrared (PIR) to detect motion in a specific area; float sensors to measure the toilet being flushed. The observed sensor data is ambiguous with respect to which activity is taking place. For example, the sensors can observe that the refrigerator is opened, but cannot observed which item is taken from the refrigerator. This makes the recognition task especially challenging.

The datasets include annotation of activities, but do not include annotation of actions. Since we do not have any ground truth for the actions and because we are only interested in using action clusters for modeling purposes, our evaluation is based solely on the inferred activities. We used the F-measure metric for evaluation, which is a combination of the average precision and recall per activity. This metric considers the recognition of each activity as equally important and provides a reliable way for evaluating activity recognition methods [3].

Data obtained from the sensors is discretized in timeslices of length $\Delta t = 60$ seconds and transformed to the changepoint representation, which has been shown to consistently gives a good performance in activity recognition [3]. We split our data into a test and training set using a 'leave one day out' cross validation. In this approach, one full day of sensor readings is used for testing and the remaining days are used for training, we cycle over all the days in the

Table 1. Information about the datasets used in the experiments

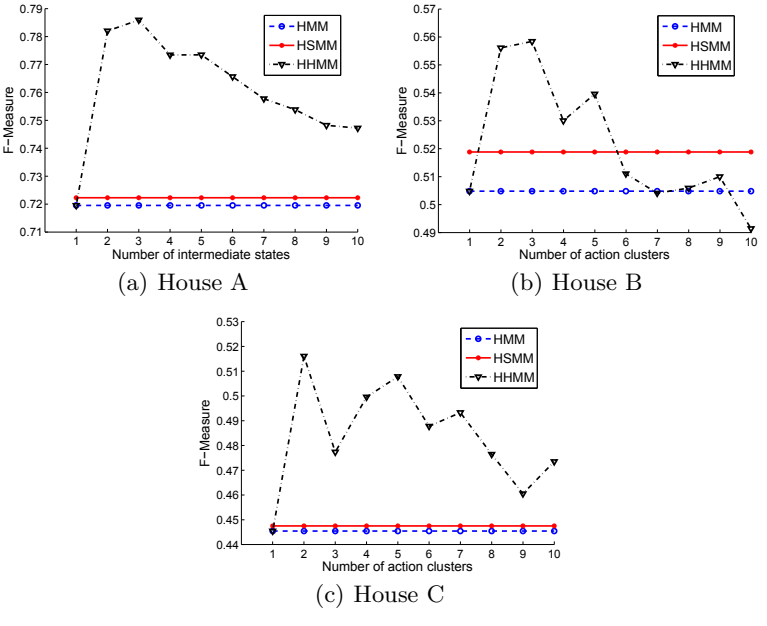
	House A	House B	House C
Activities	10	14	16
Sensors	14	23	21
Days of data	25 days	13 days	18 days

dataset and present the average performance over all test days. In the case of the HHMM, we repeat the experiment five times and present the average over those five runs. This is done because the EM algorithm requires a random initialization of the parameters.

4.2 Experiment 1: Observation Model 1

In this experiment, we use Observation Model 1 ($p(\boldsymbol{x}_t \mid y_t, z_t)$). We compare the performance of our HHMM to the performance of the HMM and HSMM. Furthermore, we experiment with various number of action clusters. The average F-measure performance over five runs for various number of action clusters is given in Figure 2 for all three houses.

We see that the performance of the HHMM is equal to the HMM when a single action cluster per activity is used. Using a single action cluster for each activity is equivalent to using an HMM and therefore results in the same performance.



(a) House A (b) House B

(c) House C

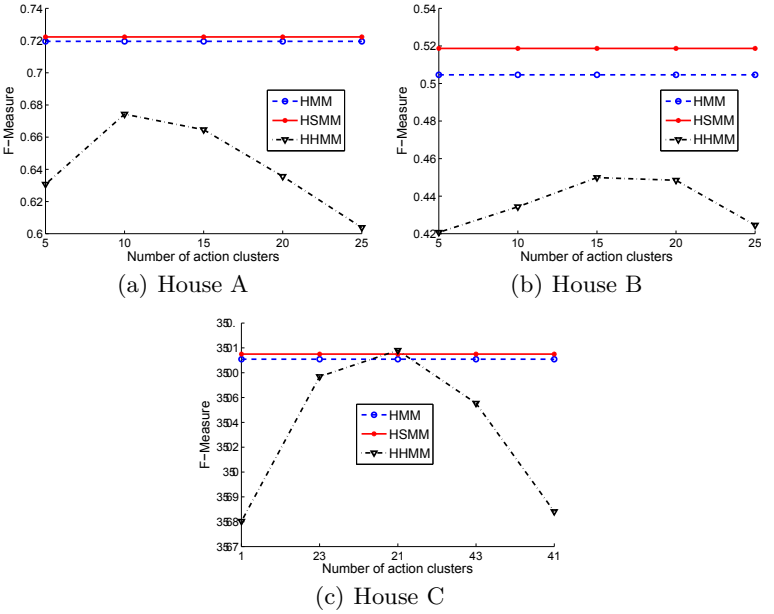
Fig. 2. Experiment 1: Plot of the F-measure performance of the HMM, HSMM and HHMM using Observation Model 1. The number of action clusters signifies the number of state values that are used for the bottom layer state variable of the HHMM.

Generally the best performance for the HHMM is obtained when using two or three action clusters, the performance decreases as more action clusters are considered. To determine the significance of our results we used a one-tail student t-test with matching paired days. The increase in F-measure performance of the HHMM, taken over an average of five runs, compared to the F-measure performance of the HMM and the HSMM is significant for houses A and C, at a confidence interval of 95%.

4.3 Experiment 2: Observation Model 2

In experiment 2 Observation Model 2 ($p(\boldsymbol{x}_t \mid z_t)$) is used. The experimental setup is similar to experiment 1 and the average F-measure performance over five runs for various number of action clusters is given in Figure 3 for all three houses.

We see that in Houses A and B, the HHMM does not manage to perform better than the HMM or the HSMM. In House C, we see a slight improvement in performance over the HMM and the HSMM, when 15 action clusters are used, but this increase in not significant. Overall, the best performance is obtained when using 10 or 15 action clusters. Using more or less action clusters than that quickly results in a significant decrease in performance.



(a) House A (b) House B

(c) House C

Fig. 3. Experiment 2: Plot of the F-measure performance of the HMM, HSMM and HHMM using Observation Model 2. The number of action clusters signifies the number of state values that are used for the bottom layer state variable of the HHMM.

5 Discussion

The results from our experiments show that observation model 1 (separate set of actions) gives significantly better performance than Observation Model 2 (shared set of actions). Observation model 1 includes a dependency on the activity, which means action clusters are allocated separately for each activity. In observation model 2 action clusters are found independent of the activities, as a result the clusters found might not be ideal for distinguishing activities, which results in a lower performance.

The use of two or three action clusters gives the best performance, when using a separate set of action clusters for each activity. Too few action clusters does not provide the model with enough expressive power, while too many action clusters results in too many parameters for which there is too little data to estimate them accurately.

Our proposed HHMM significantly outperformed the HMM and the HSMM in two of the three houses. This increase in performance is mainly due to differences in the observation model and the modeling of transition probabilities of the bottom layer state variable. The inclusion of action clusters allows the model to divide activities into separate stages, based on the actions that are performed. By modeling the transition probabilities between consecutive action clusters, we are able to calculate the probability of a particular temporal ordering of action clusters within an activity.

The presented results assume recognition is performed in an offline fashion on a daily basis. Such an approach is useful in applications such as long term health monitoring in which the activity behavior of a patient is studied over long periods of time (several months). Recognition can also be done in an online fashion, which means the recognized activity can be made available in realtime. Using online recognition will result in a decrease in performance, since only the sensor data up to the point of recognition can be used during inference. A comparison of online and offline recognition in the case of the HMM can be found in one of our previous works [4].

6 Conclusion

We presented a two layer hierarchical hidden Markov model for activity recognition in which one of the layers corresponds to action clusters and one layer corresponds to the activities. Two observation models were proposed and experiments on three real world datasets revealed that using a separate set of action clusters for each activity works best. Using a shared set of action clusters does not necessarily result in meaningful clusters and therefore using that observation model gives a significantly lower performance.

Our proposed hierarchical model outperforms the HMM and the HSMM in all datasets and does so significantly in two of the three datasets, with an increase of 7 percentage points in F-measure performance for both datasets. The gain in performance shows that our proposed hierarchy allows a more accurate modeling

of the activities recorded in the datasets. This is primarily caused by the ability of the model to take into account the temporal order of both activities and action clusters.

References

1. Bilmes, J.: A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Computer Science Institute (1997)
2. Duong, T.V., Bui, H.H., Phung, D.Q., Venkatesh, S.: Activity recognition and abnormality detection with the switching hidden semi-markov model. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, CVPR, pp. 838–845. IEEE Computer Society, Washington, DC, USA (2005)
3. van Kasteren, T.L.M., Englebienne, G., Kröse, B.J.A.: Activity Recognition in Pervasive Intelligent Environments. In: Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software, ch. 8, pp. 165–186. Atlantis Press (2011)
4. van Kasteren, T.L.M., Noulas, A., Englebienne, G., Kröse, B.J.A.: Accurate activity recognition in a home setting. In: Proceedings of the 10th International Conference on Ubiquitous Computing, Ubicomp, pp. 1–9. ACM, New York (2008)
5. Lühr, S., Bui, H.H., Venkatesh, S., West, G.A.W.: Recognition of human activity through hierarchical stochastic learning. In: Proceedings of the 1st International Conference on Pervasive Computing and Communications, PerCom, pp. 416–422. IEEE Computer Society, Washington, DC, USA (2003)
6. Muncaster, J., Ma, Y.: Hierarchical model-based activity recognition with automatic low-level state discovery. Journal of Multimedia 2(5), 66 (2007)
7. Murphy, K., Paskin, M.: Linear time inference in hierarchical hmms. In: Advances in Neural Information Processing Systems 14, NIPS (2001)
8. Nguyen, N.T., Phung, D.Q., Venkatesh, S., Bui, H.: Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, CVPR, pp. 955–960. IEEE Computer Society, Washington, DC, USA (2005)
9. Patterson, D.J., Fox, D., Kautz, H.A., Philipose, M.: Fine-grained activity recognition by aggregating abstract object usage. In: Proceedings of the 9th International Symposium on Wearable Computers, ISWC, pp. 44–51. IEEE Computer Society (2005)
10. Turaga, P., Chellappa, R., Subrahmanian, V., Udrea, O.: Machine recognition of human activities: A survey. IEEE Trans. on Circuits and Systems for Video Technology 18(11), 1473–1488 (2008)