Activity Recognition Using a Hierarchical Model

Caglar Tirkaz
Computer Science & Engineering
Sabanci University, Istanbul, Turkey
Email: caglart@sabanciuniv.edu

Dietmar Bruckner, GuoQing Yin, Jan Haase
Institute of Computer Technology
Vienna University of Technology, Austria
Email: (bruckner|yin|haase)@ict.tuwien.ac.at

***Abstract*—In this paper, we propose a human daily activity recognition method that is used for Ambient Assisted Living. The proposed system is able to learn a user's activities using the data from motion and door sensors. We extract low level features from the sensor data and feed the features to a model that combines support vector machines (SVMs) and conditional random fields (CRFs) to give accurate recognition results. We propose to combine SVM and CRF classifiers in a hierarchical model which results in better accuracies and can also make use of high level features. We conducted experiments and presented the effectiveness and accuracies of the proposed method.**

I. Introduction

Human activity recognition using sensor data has been a key problem for designing smart environments that are able to provide health monitoring and assistance to elderly people. The development of such technologies is becoming more important as the portion of elderly people gets bigger in society especially for developed countries such as in Europe [1].

Smart environments that are capable of activity recognition [2]–[6] and assistive technologies for the disabled [7], [8] has been designed. In the scope of ATTEND (AdapTive scenario recogniTion for Emergency and Need Detection) project, we are working on a system that is capable of activity recognition using a network of sensors installed in the living environment. The project also aims to detect unusual activities in the environment such as the falling of a person or a person not waking up in the morning as usual. In summary ATTEND has four main goals:

- User friendly: The system should be easily deployed and able to adapt to the environment automatically with a simple interface.
- Comfortable: The system should be as non-intrusive as possible without limiting user activities in any way.
- Accurate: The system should give high recognition results about user activities and give an alert in case something unusual occurs.
- General: The system should assume no knowledge about neither the living environment nor the location of the sensors.

In order to achieve these goals, we created a human activity recognition model that only uses information from door and movement sensors. Thus, our model does not rely on wearable sensors in decision making. In addition to that, the deployment of the system only requires placement of movement and door sensors to the living area. In order to achieve accurate results, we created an algorithm that hierarchically builds and combines conditional random fields (CRFs) and support vector machines (SVMs). The designed system learns the topology of the living environment and relates actions with places gradually with time. So, the designed system adapts itself according to the the user's behavior and the topology of the living environment. All in all, the output of ATTEND will be a product that can easily be bought and deployed with little expert knowledge. In Sec. II we talk about SVMs and CRFs and explain why they are good candidates for classifier combination; in Sec. III we describe the features we use for decision making; we explain the algorithm to create a hierarchical model in detail in Sec. IV; and finally in Sec. V we present the experiment results and conclude the paper with Sec. VI.

II. Background

SVM and CRF classifiers have been shown to be good candidates for classifier combination [9]–[11]. Essentially, we first use SVMs to learn to predict the labels of individual input sequence data items. Then, we use a CRF to predict the sequence of all output labels, where the input to the CRF is the outputs of the SVMs applied to the inputs along with the inputs to the SVM (The details are explained in Sec.IV). This two-stage method gains high accuracy from two complementary strengths: margin-maximization approaches can be more accurate than likelihood-maximization approaches as discriminative classifiers, and learning correlations between neighboring output labels helps resolve ambiguities.

Conditional random fields (CRFs) [12] have been initially proposed for natural language processing but since then they have been used to solve many other research questions including human activity recognition [13]–[16]. CRFs are undirected graphical models that are capable of labeling sequential data. Given a sequence of labels, Y, and an observation sequence, X, CRFs represent $P(Y|X)$. Unlike HMMs, which model the joint probability $P(Y, X)$ and which are generative models, CRFs are discriminative models. This feature of CRFs allow for incorporating complex features without violating independence assumptions. In our work we used the implementation of CRFs from [17].

Support vector machines are max-margin classifiers and they have been used for a wide area of classification tasks [18]. We used the SVM package LIBSVM for the implementation of SVMs [19]. Using LIBSVM one can produce probability

978-1-4673-2420-5/12/$31.00 ©2012 IEEE

estimates for given input. We use these probability estimates as an input to CRF during training and testing.

III. Feature extraction

The features that we use to classify human activities can be grouped into two categories. The first set of features can be generated directly using sensor data. Those are low-level features that represent the amount of movement, a door open/close activity etc. The second set of features are high level features that require both knowledge of the environment and the activities that are performed. That is to say, the features in the second set depend both on the topology of the house and the resident's daily routine. Those features cannot be extracted initially since there is no initial knowledge about the environment. We discuss these features in more detail in the following sections.

A. Low-level Features

We extract four types of features in this category:

- Topology features
- The time of the day
- The information from movement sensors
- The information from door and movement sensors

In order to extract topology features we discover the relationship between movement sensors. Movement sensors send a '1' if there is a movement and send nothing otherwise. The minimum sending time between two consecutive '1' is 3 seconds. In order to find the topology of the house, firstly we discretize the time of day into 3 second intervals. So, for instance, a minute is represented by a vector of length 20 for a movement sensor. Secondly, for each time interval and for each movement sensor we set a 1 if there is a message from the sensor or set it to zero otherwise. Essentially, in this scheme, each movement sensor is represented by a vector, v, which has a length proportional to the considered time interval. Then, we compute the distances between these created vectors. In order to compute the distance between two vectors, (v_1, v_2), while also supplying a window length, w, we use Alg 1. Basically, the algorithm finds the minimum distance between two vectors allowing a vector to be shifted by a window length while considering only the 1's. In our experiments we set $w = 3$, allowing nine seconds intervals for two 1's to be considered similar.

After computing the distances between all the movement sensors, we use Alg. 2 to compute the final topology of the house. As a result of this algorithm we create a connected graph to represent the house. That is, we create a graph in which there is a path between any two vertices.

We extract a single feature from the created topology which is either a 1 or 0 for each 3 seconds. This feature is 1 if the found location of the person changes from the previous time interval and 0 otherwise. This feature helps in deciding if the person is moving in the house. Doing so much work to extract only a single feature might look like to much work to do however, the created topology is later utilized when extracting high level features. Two sample topologies found using the described algorithm is illustrated in Fig. 1

Algorithm 1 Compute distance between movement sensors.

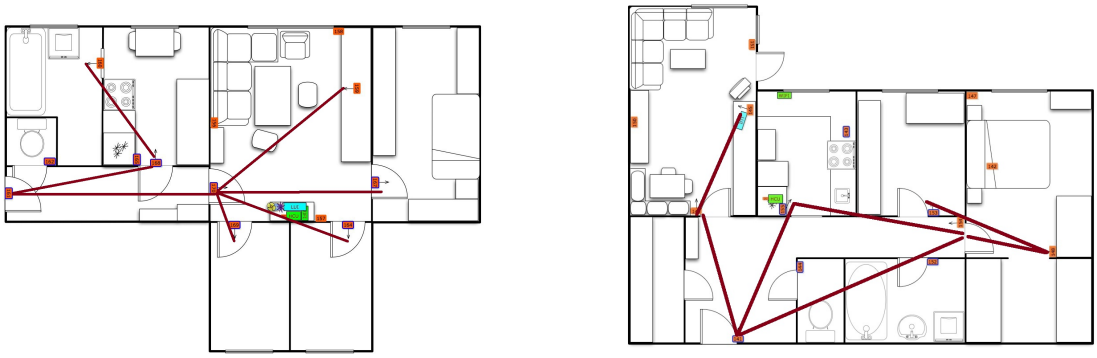
$minDist \leftarrow Inf$
$len \leftarrow length(v_1)$
for $i = -w$ **to** w **do**
  $v_3 \leftarrow shift(v_1, i)$
  $similarity \leftarrow 0$
  for $j = 1$ **to** len **do**
    if $v_3(j) == v_2(j)$ **and** $v_3(j) == 1$ **then**
      $similarity \leftarrow similarity + 1$
    end if
  end for
  $dist \leftarrow 1/similarity$
  if $dist < minDist$ **then**
    $minDist \leftarrow dist$
  end if
end for
return $minDist$

Algorithm 2 Find topology.

$V \leftarrow$ count of movement sensors
$E \leftarrow$ empty list
$G \leftarrow$ Graph(V, E)
while G is not connected **do**
  $(V_1, V_2) \leftarrow$ nearest vertices that are not already connected
  $E \leftarrow E + (V_1, V_2)$
  $G \leftarrow$ Graph(V, E)
end while

Using the time of day, we create 24 features. In order to produce these features, for a specific time we compute the distance of the time to all hours of a day. For instance, at 1am the extracted features take values $(1, 0, 1, \ldots, 11, 12, 11, 10, \ldots, 3, 2)$. So, the time of day features take on values in range [0-12] and are of length 24.

The features from motion sensors are extracted using simple filters. They represent total movement and the total change in movement for specific time intervals. As described earlier, we create a vector for each movement sensor to represent the sensor readings. In order to compute total movement for a specific time interval, we count the total number of sensor responses and we do this for varying window lengths. These features help the model identify the level of movement at a specific time interval. In order to compute the total change in movement, we count the total number of sensor responses before and after the specified time interval and take a difference. We then sum the absolute values of the differences to compute the total change in movement. These features help the model identify whether or not a person is moving around the house. We use varying window lengths in order to compute these features: $(1, 3, 5, \ldots, 11)$. Increasing window length acts as a smoothing factor while computing the features.



(a) Flat I (b) Flat II

Fig. 1. Topologies discovered using the described topology finding algorithm for two different flats.

The feature that we compute from the door and movement sensors together is aimed to find when the person goes out. This feature is turned on if there is change in one of the door sensors, and there is less than a pre-specified number, d, of movement sensor messages after that event. So, for example, if a door is closed at time t, and there is m movement sensor messages after that event in a certain time interval, we use Eq. 1 to compute this feature. Note that, if a door is closed and this feature has a positive value, the value of this feature is continuously updated for following time intervals. If the feature is 0 then the following time intervals are also set as zero.

$$f(x) = \begin{cases} 1 - (m/d) & \text{if } m \leq d \text{ is even} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Currently, the system uses only the door sensor at the outside door. Notice that the system is not given which sensor is placed at the outside door and the model has to discover it. So, in fact, the door sensors other than the outside door sensor are making the problem more complicated but these are still kept in case more features are extracted using door sensors.

B. High level features

The high level features require knowledge about the topology and the activities performed. These features aim at relating the locations with specific activities. So, for instance, the corridor of the house should be related with moving most of the time, or similarly the living room should be related with activities. Although we can learn the topology of the house using movement sensors, the action labels cannot be computed without having an initial model.

Moreover, the type of activity might change dependent on the time of the day. For instance, the bedroom is related to sleeping in the night whereas in the morning after the person wakes up it might be related to another type of activity. In order to take this into account we divide each day into 4 intervals each having 6 hours: *Morning* (6:00-12:00), *Afternoon* (12:00-18:00), *Evening* (18:00-24:00) and *Night* (24:00-6:00).

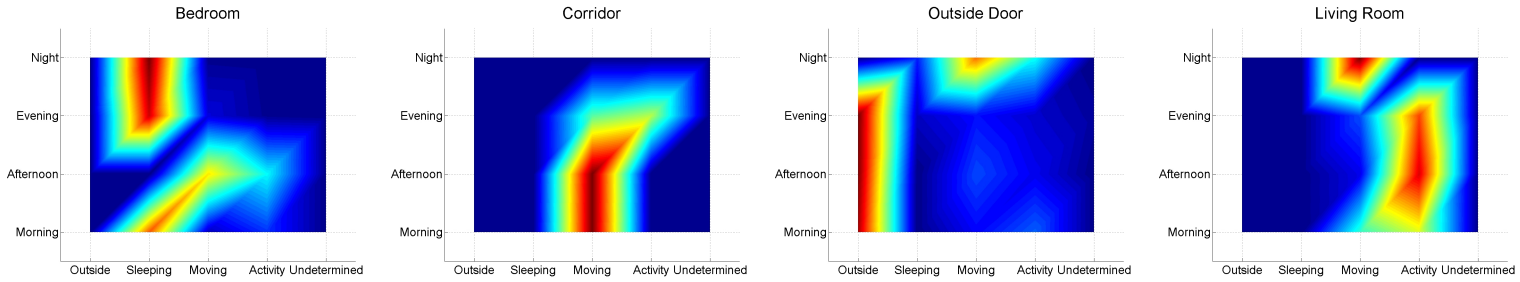
In order to extract these features, we adopted the approach presented in [13]. In the paper, Liao et al. build a hierarchical CRF to find important places and activities of a person using GPS data. We explain how we construct a hierarchical CRF for activity recognition in Sec. IV. After creating a hierarchical model, the extracted high level features are the same length as the count of pre-determined activities. In our model we used 5 labels for activities, namely: *Outside*, *Sleeping*, *Moving*, *Activity* and *Undetermined*. These labels are described in Sec. V-B. After creating an initial model, we use the discovered topology along with the model to create an activity distribution for each sensor and for each four of the time intervals of the day.

In Fig. 2, the discovered place-action distributions are illustrated. As displayed in Fig. 2a, the model successfully discovers that bedroom is related with sleeping in the night and in the morning whereas the place is related with other actions in the afternoon and in the evening. Similarly, as illustrated in Fig. 2b, the corridor is related with moving most of the time. Moreover, the model successfully relates the outside door with outside activity as can be seen in Fig. 2c. It is important for the model to find out when the person goes out. If the model cannot predict that the user is outside, and since there is no sensor activity when the user goes out, the system might raise a false alarm which is inconvenient for the user. Lastly, the living room is displayed in Fig. 2d and this place is related with activity most of the time as it is the case.

Depending on the time of the day and the location of the person, the corresponding activity distribution is extracted as high level features. For instance, if it is night time and the person is in the bedroom, we use the first row of Fig. 2a as high level feature which indicates that the person is probably sleeping.

IV. Hierarchical CRF model

We have no previous knowledge of the location that the system is going to be deployed and more importantly we do not have any labeled actions so we cannot extract the high level features in the initial phase of the deployment. In order to use the high level features, we create a hierarchical model in which an initial model is used as an input.



(a) Bedroom (b) Corridor (c) Outside door (d) Living room

Fig. 2. The extracted high level features. As displayed in Fig. 2a, the bedroom is related with *Sleeping* in the night and in the morning whereas it is related with *Moving* and *Activity* in the afternoon. The corridor is displayed in Fig. 2b and it is related with *Moving* most of the time. It is observed that the model could also locate the outside door as illustrated in Fig. 2c. Finally, the living room, Fig. 2d, is almost always related with *Activity* whereas it is related to *Moving* in the night which makes sense because the person passes through the living room when going to toilet.

In order to create a hierarchical model, we first use the training set and extract the sensor features, F_{sen}. The features extracted represent every 3 second interval. Then, we train an SVM classifier using the labeled training data. We employed a radial basis kernel and the parameters of the SVM are learned through 5-fold cross validation on the training data. These parameters are fixed and kept constant throughout the experiments. Then, we test SVM on the training data to get a probability, F_{svm}, for each training instance. Next, F_{svm} and F_{sen} are concatenated to create the updated set of features, F, and these features are used as input to CRF. That is, each SVM output for a 3 second interval is used just like any other feature and all features are used to train the CRF classifier. The CRF model has a simple chain structure where each node has an edge with its previous and next neighbor. Each node is used to make a decision for a 3 second interval and a chain contains 1200 nodes. So, essentially a chain instance is used to make predictions for one hour.

Given the initial SVM and CRF, the algorithm we use to create a hierarchical model on test data is displayed in Alg. 3. As a first step, F_{sen} is extracted and the SVM is used to make predictions for each test instance to create F_{svm}. Then, the test data are labeled, L_{update}, using the CRF and the concatenated set of features, $F = [F_{sen}, F_{svm}]$.

The core of our algorithm starts at line 9 of the Alg. 3 where in each iteration we create a more refined model. We start the iteration by saving the previously found labels. After that step, the high level features, F_{high}, is created using the topology and the found labels. The sensor features are concatenated with high level features to create hierarchical features, F_{hier}. At line 13, an updated SVM is created using F_{hier} and L_{update}. So essentially in the algorithm not only the CRF but also the SVM changes with each iteration. The algorithm continues by creating F_{svm} using the updated SVM on the test data. All the extracted features are then collected, $F = [F_{hier}, F_{svm}]$ and the updated CRF is built using F and L_{update}. We use the updated CRF model again on the test data and update the labels with the new results. This iteration continues until either the maximum number of iterations is reached or the predicted labels do not change.

Algorithm 3 Build hierarchical model.

1: $SVM \leftarrow$ The initial SVM
2: $CRF \leftarrow$ The initial CRF
3: $F_{sen} \leftarrow$ Sensor features
4: $F_{svm} \leftarrow$ Test SVM on test data
5: $F \leftarrow [F_{sen}, F_{svm}]$
6: $L_{update} \leftarrow$ Test CRF with F
7: Init $maxIter$
8: $iter \leftarrow 0$
9: **repeat**
10: $L_{init} \leftarrow L_{update}$
11: $F_{high} \leftarrow$ High level features with topology and L_{update}
12: $F_{hier} \leftarrow [F_{sen}, F_{hier}]$
13: $SVM \leftarrow$ Train SVM with F_{hier} and L_{update}
14: $F_{svm} \leftarrow$ Test SVM on test data
15: $F \leftarrow [F_{hier}, F_{svm}]$
16: $CRF \leftarrow$ Train Hierarchical CRF
17: $L_{update} \leftarrow$ Test CRF with F
18: $iter \leftarrow iter + 1$
19: **until** $iter < maxIter$**and** $L_{init} \neq L_{update}$

V. Experiments

A. Experimental setup

The flats we used in the experiments are displayed in Fig. 1. The first flat contains 8 motion and 8 door sensors whereas the second flat contains 7 motion and 6 door sensors. For each flat we recorded and labeled 4 days of activities. The collected data is discretized by 3 second intervals. Thus, for each flat there are 115200 $(4 * 24 * 60 * 60/3)$ instances.

B. Data annotation

In order to annotate the data we created an annotation tool where the user could select one of the 5 annotations available in the interface for each time interval. An illustration of the interface is given in Fig. 3. The available labels are:

1) *Outside*
2) *Sleeping*
3) *Moving*

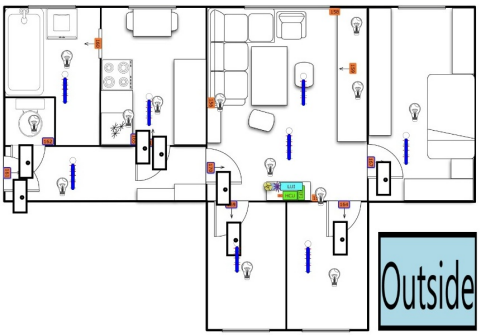


Fig. 3. The interface that is used for data annotation. The user can select one of the available labels for each time interval.

TABLE I
TWO-FOLD CROSS VALIDATION RESULTS ON THE TRAINING DATA. THE ROWS REPRESENT THE PERFORMANCE OF THE SYSTEM ON THE FOLDS.

SVM Result(%)	CRF Result(%)	Hierarchical Model Result(%)
74.40	85.96	86.56
83.97	80.72	81.08

4) *Activity*
5) *Undetermined*

Most of the labels are self explanatory other than the last label, *Undetermined*. Although there is no unusual event in the data, the user can select this label if there is no movement for some time and the person is not outside. We also want to detect these cases in case an emergency occurs.

C. Experiment Results

In order to test the hierarchical model, we first experimented with how the model performs for the same flat where the performance is defined as the proportion of the activities found by the model that are in par with the labeled data. We did a two-fold cross validation on the training data of the first flat for that purpose. That is, we split the labeled data for the first flat into 2 halves and firstly used the first half for training and the second for testing, secondly we used the second half for training and the first half for testing. The results of this experiment is given in Tab. I. The first line of the table shows the result of this experiment. In the first test, the SVM, CRF and hierarchical model achieve an accuracy of 74.40%, 85.96% and 86.56% respectively. As can be seen classifier combination and creating a hierarchical model achieves higher results for the first test. The second cross validation result is a bit different: The SVM classification gives the highest result in this test. However, on the average the hierarchical model gives the best accuracy and building a hierarchical model always improves the CRF accuracy.

In our second test, we trained and tested our algorithm using different flats. In order to do that, we used the first flat for training and the second flat for testing. We again split the training data into two and trained two models and tested them both on the second flat data. The test results are displayed in Tab. II. As can be seen, in both tests the hierarchical model gives the best accuracies on the test data.

TABLE II
ACCURACIES USING THE FIRST FLAT FOR TRAINING AND THE SECOND FLAT FOR TESTING. ROWS REPRESENT THE ACCURACIES USING THE FIRST AND THE SECOND HALF OF THE TRAINING DATA.

SVM Result(%)	CRF Result(%)	Hierarchical Model Result(%)
83.56	86.84	87.70
81.68	82.52	83.78

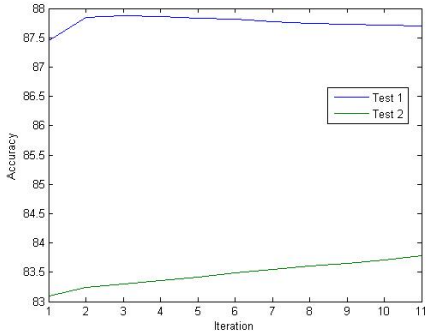
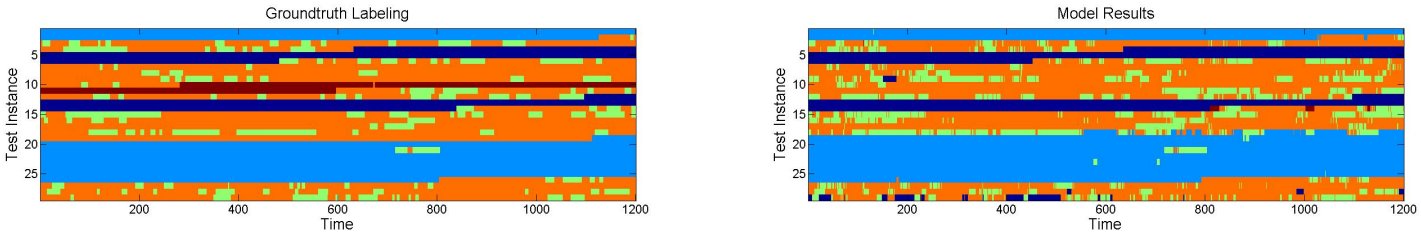


Fig. 4. The test accuracies plotted against iterations when building the hierarchical model.

We also plotted how the accuracy of the hierarchical model changes with each iteration of the algorithm. The accuracy/iteration plots are displayed in Fig. 4. As described in Sec. IV, while building the hierarchical model, the algorithm makes iterations and refine the model with each iteration. We set the maximum number of iteration to 10 during experiments. As can be seen in the figure, in the first test, the accuracy of the hierarchical model increases with the first iteration while the accuracy drops a bit with more iterations. On the contrary, in the second test, the accuracy of the hierarchical model increases constantly with increasing number of iterations.

We also visualized the decisions made by the model for the experiment given in Tab. I. The visualization is displayed in Fig. 5 where each color represents a different label. The respective colors for *Outside*, *Sleeping*, *Moving*, *Activity* and *Undetermined* are dark blue, light blue, light green, orange and dark red. In these figures the x axis represents time and time is discretized in 1 hour intervals where the model makes a decision for every 3 seconds. Thus, the maximum x value in the figures is 1200. The y axis represents consecutive hours and 30 hours of decisions are displayed in the figures. Notice that, although there is no unusual activity in the experiments, in the groundtruth labeling there is a time interval that is labeled as *Undetermined* (dark red) by the user. The reason for that is lack of activity in movement sensors for a long period of time. The same time interval is labeled as *Activity* by our model. That makes sense because there is a little time interval in the groundtruth data that is labeled as *Undetermined* and the model might ignore this label during testing. In order to better deal with situations like that synthetic alarm situations can be created.



(a) Groundtruth (b) Model Results

Fig. 5. The visualization of test results. Each color in the figure represents a different label. The respective colors for *Outside*, *Sleeping*, *Moving*, *Activity* and *Undetermined* are dark blue, light blue, light green, orange and dark red. In Fig. 5a the groundtruth labeling done by the user is displayed whereas in Fig.5b the decisions made by the model are displayed for the same time interval.

VI. Conclusion

In this paper, we described a human activity recognition system. There are few remarks that can be made about the system. Firstly, although the system is not given any information about the environment the model successfully recovers the topology of a new flat and relates the locations with specific activities. This is achieved using a topology finding algorithm along with the creation of a hierarchical model. The model achieves accurate results through hierarchically building SVM and CRF classifiers in decision making. As discussed earlier, the combination of SVM and CRF classifiers yield better results since they provide orthogonal information.

Secondly, the designed system relies only on information from door and movement sensors and not wearable sensors. This design is user friendly and comfortable since once the sensors are placed in the living environment the user of the system does not need to worry about forgetting to wear a sensor anymore. The system learns the living environment and user's habits automatically in time. Moreover, the deployment of the system requires little effort and expert knowledge.

Thirdly, as we demonstrated in the experiments section, the model generalizes well to a flat that is completely different than the flat used for training. That is made possible through the use of low level features that generalize well to different flats and a successful topology finding algorithm. The high level features are only extracted during building of the hierarchical model and they provide more specific information about locations. These features allow the model to relate places with activities which also add to the performance of the model.

References

[1] R. Muenz, "Aging and demographic change in european societies : main trends and alternative policy options," The World Bank, Social Protection Discussion Papers 39174, Mar. 2007. [Online]. Available: http://ideas.repec.org/p/wbk/hdnspu/39174.html

[2] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. A. Shafer, "Easyliving: Technologies for intelligent environments," in *Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, ser. HUC '00. London, UK: Springer-Verlag, 2000, pp. 12–29.

[3] D. Surie, T. Pederson, and L.-E. Janlert, "The easy ADL home: A physical-virtual approach to domestic living," *J. Ambient Intell. Smart Environ.*, vol. 2, pp. 287–310, August 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1834668.1834674

[4] C. Zhu and W. Sheng, "Human daily activity recognition in robot-assisted living using multi-sensor fusion," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ser. ICRA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3644–3649. [Online]. Available: http://dl.acm.org/citation.cfm?id=1703775.1704035

[5] J. Ye and S. Dobson, "Exploring semantics in activity recognition using context lattices," *J. Ambient Intell. Smart Environ.*, vol. 2, pp. 389–407, Dec. 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1877756

[6] G. Yin and D. Bruckner, "Build user daily activity model and model structure changing," in *Proceedings of the 10th IEEE AFRICON*, 2011.

[7] D. J. Patterson, O. Etzioni, D. Fox, and H. Kautz, "Intelligent ubiquitous computing to support alzheimer's patients: Enabling the cognitively disabled," in *In UbiCog 02: First International Workshop on Ubiquitous Computing for Cognitive Aids*, 2002.

[8] P. C. Roy, S. Giroux, B. Bouchard, A. Bouzouane, C. Phua, A. Tolstikov, and J. Biswas, "A possibilistic approach for activity recognition in smart homes for cognitive assistance to alzheimers patients," in *Activity Recognition in Pervasive Intelligent Environments*, ser. Atlantis Ambient and Pervasive Intelligence, L. Chen, C. D. Nugent, J. Biswas, J. Hoey, and I. Khalil, Eds. Atlantis Press, 2011, vol. 4, pp. 33–58.

[9] G. Hoefel and C. Elkan, "Learning a two-stage svm/crf sequence classifier," in *Proceedings of the 17th ACM conference on Information and knowledge management*, ser. CIKM '08. New York, NY, USA: ACM, 2008, pp. 271–278.

[10] P. Cai, H. Luo, and A. Zhou, "Semantic entity detection by integrating crf and svm," in *Web-Age Information Management*, ser. Lecture Notes in Computer Science, L. Chen, C. Tang, J. Yang, and Y. Gao, Eds. Springer Berlin / Heidelberg, 2010, vol. 6184, pp. 483–494.

[11] A. Ekbal and S. Bandyopadhyay, "Improving the performance of a ner system by post-processing and voting," in *Structural, Syntactic, and Statistical Pattern Recognition*, ser. Lecture Notes in Computer Science, N. da Vitoria Lobo, T. Kasparis, F. Roli, J. Kwok, M. Georgiopoulos, G. Anagnostopoulos, and M. Loog, Eds. Springer Berlin / Heidelberg, 2008, vol. 5342, pp. 831–841.

[12] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. [Online]. Available: http://dl.acm.org/citation.cfm?id=645530.655813

[13] L. Liao, D. Fox, and H. Kautz, "Hierarchical conditional random fields for gps-based activity recognition," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, S. Thrun, R. Brooks, and H. Durrant-Whyte, Eds. Springer Berlin / Heidelberg, 2007, vol. 28, pp. 487–506.

[14] D. L. Vail, M. M. Veloso, and J. D. Lafferty, "Conditional random fields for activity recognition," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS '07. New York, NY, USA: ACM, 2007, pp. 235:1–235:8.

[15] K.-C. Hsu, Y.-T. Chiang, G.-Y. Lin, C.-H. Lu, J. Y.-J. Hsu, and L.-C. Fu, "Strategies for inference mechanism of conditional random fields for multiple-resident activity recognition in a smart home," in *Proceedings of the 23rd international conference on Industrial engineering and other applications of applied intelligent systems - Volume Part I*, ser. IEA/AIE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 417–426.

[16] E. Nazerfard, B. Das, L. B. Holder, and D. J. Cook, "Conditional random

fields for activity recognition in smart environments," in *Proceedings of the 1st ACM International Health Informatics Symposium*, ser. IHI '10. New York, NY, USA: ACM, 2010, pp. 282–286.

[17] M. Schmidt, *UGM: Matlab code for undirected graphical models*, 2012. [Online]. Available: http://www.di.ens.fr/~mschmidt/Software/UGM.html

[18] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273–297.

[19] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.