AALO: Activity recognition in smart homes using Active Learning in the presence of Overlapped activities

Enamul Hoque & John Stankovic
Department of Computer Science
UVA Center for Wireless Health, University of Virginia
Charlottesville, VA, USA
Email: {enamulhoque1, stankovic}@cs.virginia.edu

***Abstract*—We present AALO: a novel Activity recognition system for single person smart homes using Active Learning in the presence of Overlapped activities. AALO applies data mining techniques to cluster in-home sensor firings so that each cluster represents instances of the same activity. Users only need to label each cluster as an activity as opposed to labeling all instances of all activities. Once the clusters are associated to their corresponding activities, our system can recognize future activities. To improve the activity recognition accuracy, our system preprocesses raw sensor data by identifying overlapping activities. The evaluation of activity recognition performance on a 26-day dataset shows that compared to Naive Bayesian (NB), Hidden Markov Model (HMM), and Hidden Semi Markov Model (HSMM) based activity recognition systems, our average time slice error (24.15%) is much lower than NB (53.04%), and similar to HMM (29.97%) and HSMM (26.29%). Thus, our active learning based approach performs as good as the state of the art supervised techniques (HMM and HSMM).**

I. Introduction

Due to increasing number of elderly people and single households living alone, wide-spread deployment of sensors has become prominent in home environments for detecting medical emergencies and assessing behavioral changes. In such smart homes, living spaces and objects used for daily activities are instrumented with passive sensors. When a resident moves from one room to another or uses different objects that have attached sensors, a series of sensor firings with corresponding timestamps are generated that allow us to automatically detect which activity the resident is currently performing, its duration and what objects are used for this activity. Accurate detection and summarization of these daily activities are essential for many remote home healthcare applications such as assessing behavioral rhythms ([21], [5]), monitoring cognitive decline ([9]). In this work, our focus is on unobtrusive long-term monitoring of daily activities of single person homes on a daily basis.

However, existing activity recognition algorithms suffer from many practical problems. Many of them ([20], [3], [17], [19], [24]) are based on supervised learning where the training data needs ground truth i.e., accurate labeling of all activities. To ensure high accuracy, the classifiers need to be trained with long traces of data that may range from months to years. However, collecting ground truth for such a long period is difficult. Either the resident has to keep record of all the activities which is not convenient or we need to use cameras and label each activity manually which may not be practical. There are some existing unsupervised activity recognition algorithms that do not need ground truth ([13], [16], [4]). They either require to mine activity models from web definitions or depend on domain knowledge about activities and the environment (e.g., which objects are used during an activity). Such systems may not be generalized to wide variety of home environments and deployments.

In this paper, we approach this problem with the insight that different daily activities are performed in different rooms, each of them triggers a different set of sensors to fire and is often performed during similar time periods of day. Therefore, we divide the sensor firings into room-level occupancy episodes (segmentation), and then for each room, we find the group of sensors that are frequently fired together (mining) in similar times with similar durations (clustering). Our hypothesis is that each such group represents a daily activity and if we can automatically detect these groups from the raw sensor firings, users can just label each group as an activity so that all the instances of this group can be automatically labeled. Our approach is a type of active learning; a learning technique where the system chooses the subset of training data that needs to be labeled by users ([14]). We use unsupervised clustering to find the clusters representing daily activities and users need to label each cluster as one activity.

However, one practical problem that needs to be addressed is overlapping activities. For example, people may leave the kitchen in the middle of cooking to do something else and come back again to finish cooking. Here, one activity spans multiple occupancy episodes. Alternatively, people may cook and have a drink at the same time while in the kitchen. Here, multiple activities occur in the same occupancy episode.

In this paper, we present a novel **A**ctivity recognition system for a single person home using **A**ctive **L**earning considering **O**verlapped activities (AALO). Our main contributions are:

1) A novel framework for training activity recognition systems that includes segmentation, mining and clustering of

low level sensor events. To improve accuracy, our system preprocesses raw sensor data by identifying overlapping activities across multiple occupancy episodes. Users only need to label each cluster as one activity after training which ensures feasibility of long term training. We evaluate this contribution with a real world dataset ([20]) consisting of 26 days of data. After our system generates the clusters, we label each cluster as an activity and compare these labels with the labeled ground truth for each instance of each activity. Using our system, the user only needs to label 19 clusters after training as opposed to labeling 291 activity instances over the 26 days.

2) An activity recognition system based on active learning that automatically recognizes new room-level occupancy episodes as members of one of the clusters (i.e., activities) constructed during training. Comparison of the activity recognition performance of our system with state of the art supervised activity recognition systems including Naive Bayesian (NB), Hidden Markov Model (HMM), and Hidden Semi Markov Model (HSMM) based solutions using the same 26-day dataset shows that the average of time slice error over all activities for our system (24.15%) is much lower than NB (53.04%), and almost similar to HMM (29.97%) and HSMM (26.29%).

II. Related Work

Recognizing daily activities in complex home settings using in-home sensors is a well-researched problem. We consider the existing solutions that use simple sensors that detect movements of the resident from one room to another (i.e., motion sensors in the doorway) or changes in state of objects and devices (i.e., contact sensors). Kasteren et al. use temporal probabilistic models in [20] to recognize activities from sensor readings. In this work, authors divide time series data into time slices of constant-length and label the activity for each slice. They use the sensor firings of each time slice to build probabilistic models (naive Bayes, hidden markov models and conditional random fields). However, defining a constant length time slice for all activities may not be practical. They also do not consider duration of activities.

Later in [19], the same authors use hidden semi markov models that consider duration of an activity to improve accuracy of activity recognition. Zhang et al. [23] also use activity durations for recognizing them. Tapia et al. apply a naive Bayes classifier in [17] for activity recognition. They propose to learn different time slice durations for different activities from the training data and use these durations for building models for different activities. Logan et al. use both naive Bayes and C4.5 decision tree models for activity recognition in [11]. Recent works ([8], [10]) have focused on recognizing concurrent and inter–leaved daily activities.

One common problem associated with all the works discussed so far is that they require accurate labeling of activities (either by the resident or by manual annotation after viewing the data) during training which may be difficult to obtain for a long period. As an alternative, Kasteren et al. present a technique in [18] to use the ground truth collected in one house to train activity recognition systems in other houses. However, details of activities may vary significantly from person to person and from home to home in which case this technique may not perform well.

Barger et al. present an unsupervised technique in [2] that clusters the sensor firings using mixture models. Each cluster has distinct time of occurrence, duration and rate of sensor firings. However, it does not consider the group of sensors being fired for clustering. Zheng et al. [24] also use clustering by a self-adaptive neural network to summarize the timing of sensor firings for each activity. Gu et al. [7] present an unsupervised approach for activity recognition based on object-use fingerprints to recognize daily activities without human labeling. This is done by first mining a set of object terms for each activity class from the web, and then mining contrast patterns among object terms based on emerging patterns. Similarly, Emmanuel et al. [16] extract activity models from text corpora such as the web and uses them to automatically produce labeled segmentations of activity data.

Philipose et al. present another similar approach in [13] to learn activity models from the web. However, the list of objects used for different activities may not be always extracted from web and mapping them to the actual deployed sensors is also complicated. Dimitrov et al. [4] propose another unsupervised activity recognition approach that utilizes background domain knowledge about user activities and environment such as which objects are used for an activity. Unfortunately, such background knowledge may not be available. Also, none of the unsupervised solutions mentioned so far address overlapping activities which may degrade the performance.

As a compromise between the supervised and unsupervised techniques, there have been previous work that require only a subset of training data to be labeled by users so that annotation effort is reduced. Stikic et al. [15] apply multi-instance learning for activity recognition from sparsely labeled data. Users are prompted after a predefined time interval (which is varied from 10 to 180 minutes) for labeling some activities. Therefore, users need to provide feedback multiple times in a day as opposed to our approach of labeling the clusters offline after training. Wu et al. [22] present a semi-automatic lifelog summarization system for elderly care. Similarly, Longstaff et al. [12] use active learning for activity recognition. However, both these systems require the users to always carry cell phones (having embedded sensors e.g., acclerometer, GPS, microphone) which may not be comfortable.

III. Framework For Training

Fig. 1 shows the block diagram of our training framework for the activity recognition system. The input I to the system is a sequence of pairs of the form (s_i, t_i) where $s_i \epsilon S$ (S is the set of all sensors deployed in a home) represents a sensor firing at time t_i. The set of sensors S may include passive infrared (PIR) to detect motion in a specific area, reed switches to measure open-close states of doors and cupboards, pressure mats to measure sitting on a couch or lying in bed, float sensors to measure the toilet being flushed; temperature sensors to measure the use of the stove or shower. We assume

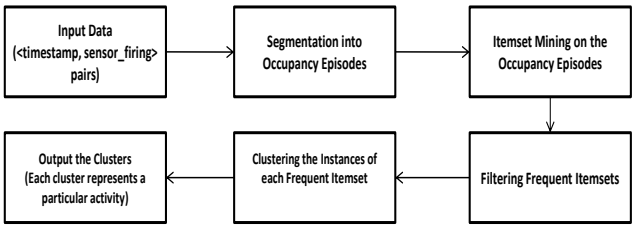

Fig. 1: Block Diagram of the Training Framework

that each sensor is associated with only one room $r \epsilon R$ (R is the set of all rooms in a home) and this information is available to our system. I consists of all the sensor firings and their corresponding timestamps during the entire training period. Now we describe the different steps of our framework.

A. Segmentation into Occupancy Episodes

Most activities of daily living have spatial regularity. For example, we cook in the kitchen and sleep in the bedroom. Therefore, the first step of our training algorithm is to segment consecutive sensor firings based on which room the sensors are in. Algorithm 1 shows the pseudo code of the segmentation algorithm. The output of this algorithm is a set of room occupancy episodes of the form ($roomID$, $entranceTime$, $duration$, $usedSensors$). When calculating the duration of an occupancy episode at line 9 of this algorithm, we consider the time interval between the last and first sensor firings of that room during that episode. Alternatively, we could consider the time interval between the first sensor firing in the next room and the first sensor firing of the previous room as the duration of the last occupancy episode of the previous room. We did not choose this option, because based on the floor plan of different homes and sensor deployment, sometimes it may happen that in between being in these two rooms, the resident was in a room where there are no sensors. Therefore, choosing this option would lead us to infer that the resident was in the last room for the entire period.

Algorithm 1 Segmentation Algorithm

1: {Input: I, a sequence of pairs of the form (s_i, t_i)}
2: {Output: E, a set of room occupancy episodes of the form $(roomID_i, entranceTime_i, duration_i, usedSensors_i)$}
3: $E = []$;
4: $previous\_room = room[s_1]$;
5: **for** each $<s_i, t_i>$ in I **do**
6: **if** $room[s_i] \neq previous\_room$ **then**
7: $new\_segment.room\_ID = previous\_room$;
8: {$new\_segment.entranceTime$ = start time of the last occupancy episode;}
9: {$new\_segment.duration$ = duration of the last occupancy episode;}
10: {$new\_segment.usedSensors$ = sensors fired during the last occupancy episode;}
11: $E.add(new\_segment)$;
12: $previous\_room = room[s_i]$;
13: **else**
14: {keep track of which sensors are used during this occupancy episode}
15: **end if**
16: **end for**

Our hypothesis is that if we find some segments that have similar sensor uses, start times and durations over the course of the entire training period, then such segments represent one daily activity. However, one practical problem against this hypothesis arises from the fact that people may not always start and finish an activity within the same occupancy episode. For example, people may get up in the middle of sleeping, visit the toilet and then come back to sleep again. Similarly, in the middle of cooking, people may leave the kitchen to do something else and come back again to finish cooking. In such cases, one instance of an activity can expand across multiple occupancy episodes. Each of these episodes may not have the start times or durations or set of used sensors that are representative of the actual activity. Accordingly, the corresponding instances of that activity would remain undetected. Here we discuss our solution to solve this problem which is generic and can be applied to any home deployment.

1) Successive Occupancy Episode Merging: To address this practical problem, we construct a new occupancy episode by merging two occupancy episodes in the same room that are separated by less than a *time interval threshold*. For each room, we define a *time interval threshold* as the normal time interval between two successive visits to the same room. To calculate this threshold for each room, we enumerate time intervals between all the successive visits to that room during the entire training period and calculate the first and third quartiles (Q_1 and Q_3, defined as the 25th and 75th percentiles). We set the *time interval threshold* of this room as $(Q_1 - h)$, which represents the lower outer fence of a corresponding box plot ($h = 3 \times (Q_3 - Q_1)$). Any value less than the lower outer fence is an extreme outlier. We use this technique, because during such occurrences of overlapping activities, the time interval between successive room visits are expected to be shorter than usual and also such occurrences do not occur too frequently. However, if such occurrences are frequent i.e., the resident frequently leaves a room while performing a specific activity in that room, in that case the time interval will no longer be less than the calculated time interval threshold. Accordingly, no new occupancy episodes will be considered. In such cases, that particular activity is usually done in two episodes and each episode will be individually clustered.

For two successive occupancy episodes in a room r, ($entranceTime_i$, $duration_i$, $usedSensors_i$) and ($entranceTime_j$, $duration_j$, $usedSensors_j$) ($j > i$), if we find that the time interval between them ($entranceTime_j - entranceTime_i - duration_i$) is less than r's *time interval threshold*, then by merging them we create a new occupancy episode with start time $entranceTime_i$, duration $entranceTime_j + duration_j - entranceTime_i$, and set of used sensors $usedSensors_i \cup usedSensors_j$. We do not delete the two smaller episodes, as each of them may represent one activity by itself. If the merged episode or the smaller episodes do not represent any activity, the clustering step would ignore them.

B. Itemset Mining on Occupancy Episodes

Specific activities of daily living are performed using specific sensors (i.e., objects). For each room r, there are a set of occupancy episodes of the form (r, $entranceTime_i$,

$duration_i$, $usedSensors_i$), where i = 1, 2, ..., *Number of Occupancy Episodes in* r. In this step, from each of these tuples we only use $usedSensors_i$ which is of the form $\{s_{ij}\}$ ($s_{ij} \epsilon S$). The temporal characteristics of these tuples are used in the next step. We apply frequent itemset mining ([1]) on $usedSensors_i$ of all occupancy episodes of room r where each sensor s_{ij} is an item and the group of sensors fired in each occupancy episode $\{s_{ij}\}$ is a transaction. The goal of the itemset mining is to find the groups of sensors (i.e., items) that are frequently fired together (i.e., occur together in transactions); they are called frequent itemsets. Our hypothesis is that each frequent itemset represents an activity.

We use a state of the art itemset mining algorithm Apriori ([1]). For each room r, we run Apriori separately with $\{s_{ij}\}$ as input (i = 1, 2, ..., *Number of Occupancy Episodes in* r). As output, we get the set of frequent itemsets $\{FI_k\}$,(k = 1, 2, ... *Number of Frequent Itemsets for* r), where each FI_k is a set of sensor firings (i.e., items) of the form $\{s_{kl}\}$ (l = 1, 2, ..., *Number of Sensors in Frequent Itemset* FI_k). An itemset is considered to be frequent if the number of different occupancy episodes (i.e., transactions) in which the itemset occurs is more than a threshold number of days. In the Apriori algorithm, this threshold is called the *support threshold* which we specify before running the algorithm.

After getting the output, for each frequent itemset FI_k, we find the occupancy episodes in room r where FI_k occurs and we construct the set of tuples $\{(startTime_{km}, duration_{km})\}$, m = 1, 2, ..., *Number of Occupancy Episodes where* FI_k *occurs*. Here, $startTime_{km}$ is the earliest timestamp when any sensor $s_{kl} \epsilon FI_k$ fires during occupancy episode m. Suppose, $endTime_{km}$ is the latest timestamp when any sensor $s_{kl} \epsilon FI_k$ fires during occupancy episode m. $duration_{km}$ is defined as the difference between $endTime_{km}$ and $startTime_{km}$. Each frequent itemset along with thus constructed set of tuples are used in the next step for filtering significant frequent itemsets.

C. Filtering Frequent Itemsets

One disadvantage of Apriori is that it produces redundant itemsets. To remove redundant itemsets, we could have used an itemset mining algorithm that produces a set of maximal frequent itemsets. A set of maximal frequent itemsets FI has the property that it does not have any pair of itemsets $FI_1, FI_2 \epsilon FI$ ($FI_1 \neq FI_2$) such that $FI_1 \subset FI_2$ or vice versa. However, in daily activities, it may be normal that FI_1 is a subset of FI_2, but FI_1 occurs by itself (when FI_2 does not occur) during significant number of occupancy episodes. Suppose, for a room ('kitchen'), two frequent itemsets are $\{s_1, s_2, s_3, s_4\}$ (represents the activity 'prepare breakfast') and $\{s_1, s_3\}$ (represents the activity 'prepare coffee'). Here, $\{s_1, s_3\}$ is a subset of $\{s_1, s_2, s_3, s_4\}$, therefore each occupancy episode where s_1, s_2, s_3, and s_4 fire , s_1 and s_3 also fire i.e, whenever the user prepares breakfast, he also prepares coffee. However, in this room (i.e., 'kitchen'), there may be such occupancy episodes where only s_1 and s_3 fire i.e., the user may also prepare coffee at other times of the day. If such instances are more than the *support threshold*, then both $\{s_1, s_2, s_3, s_4\}$ and $\{s_1, s_3\}$ should be included in the set of frequent itemsets so that we can identify both these activities separately. We do not use an itemset mining algorithm that produces a set of maximal frequent itemsets, because it would have deleted $\{s_1, s_3\}$. Alternatively, we define and reduce redundant itemsets in the following way.

As stated earlier, we represent an instance ins_k of a frequent itemset FI_i by the tuple ($startTime_{ik}$, $duration_{ik}$) (k = 1, 2, ..., *Number of Occupancy Episodes where* FI_i *occurs*). An instance ins_p = ($startTime_{ip}$, $duration_{ip}$) *temporally covers* another instance ins_q = ($startTime_{iq}$, $duration_{iq}$) if $startTime_{ip} \leq startTime_{iq}$ *and* ($startTime_{ip}$ + $duration_{ip}$) $\geq$ ($startTime_{iq}$ + $duration_{iq}$). For each instance ins_k of a frequent itemset FI_i, we consider the instance ins_k as *covered* by another frequent itemset FI_j, if $FI_i \subset FI_j$ and FI_j has an instance that *temporally covers* ins_k. For each frequent itemset FI_i of a room, we remove all its instances that are covered by instances of other frequent itemsets of the same room. After removing covered instances, if total number of remaining instances of FI_i is less than the *support threshold*, we define FI_i as a redundant frequent itemset and delete it.

Note that, in the set of frequent itemsets we have constructed thus far, for a room, there can be two instances of two different frequent itemsets that may belong to the same occupancy episode. For example, it may be the case that on some days, the user is cooking dinner and at the same time doing laundry in the washing machine in the same occupancy episode. If there are many other days, when the user does not do these two activities during the same occupancy episode, then we will have two different frequent itemsets, one consisting of sensors related to cooking, and the other related to sensors related to washing machine. Some instances of these two will belong to same occupancy episodes. In this way, we can differentiate and identify overlapping activities in the same room. However, if the user always does two activities in the same room, then our system will not be able to differentiate between two such activities and will generate one frequent itemset.

D. Clustering Instances of Each Frequent Itemset

As we hypothesized earlier, each frequent itemset represents one activity. However, there may be multiple activities that use the same set of sensors (e.g., prepare breakfast and prepare dinner). To differentiate among such activities, we need to consider the times of day when the group of sensors are used and also the durations. Therefore, in this step, we cluster the instances of each frequent itemset based on their temporal characteristics i.e., start times and durations.

We use the DBSCAN clustering algorithm ([6]) which is a density based clustering algorithm. For each frequent itemset FI_i of a room, we run DBSCAN separately on the set of tuples $\{(startTime_{ik}, duration_{ik})\}$ (k = 1, 2, ..., *Number of not Covered Instances* FI_i *has*). We normalize each attribute of each tuple before clustering. The major advantage of DBSCAN is that we do not need to specify how many clusters there are. This is important, because we do not know

how many different activities a resident performs in different rooms. DBSCAN automatically calculates the neighborhood radius of each cluster.

Note that, there may be some frequent itemsets for which some instances are not part of any clusters. During training, we do not consider such instances as part of any activity and ignore them as outliers. After training, when our system is used for activity recognition, we report such instances (that are not within the neighborhood radius of any cluster; details in Section IV) as irregular / anomalous behavior. Also, there may be some frequent itemsets for which DBSCAN does not produce any significant cluster. For such frequent itemsets, we try to cluster them again based on durations of each instance only. This is because, there are activities that do not have any regularity about which time of day they occur. However, they may have similar durations (e.g., the user can take a drink at any time of day; the duration of this activity should be similar). Finally, if there is any such frequent itemset for which even clustering based on durations does not produce any significant cluster, we remove that frequent itemset.

After the clustering, for each room, we get one or more clusters each of which represents a particular event that happens in that room, uses same set of objects, may or may not start in similar times and last for similar durations. We consider each of these events (represented by each cluster) as an activity.

E. Output

The output is the set of all clusters constructed by applying DBSCAN on all frequent itemsets of all rooms separately. Each cluster C_i is represented by the tuple ($Used\_Sensors_i$, $Mean\_Start\_Time_i$, $Mean\_Duration_i$, $Neighborhood\_Radius_i$, $Probable\_Label_i$). $Mean\_Start\_Time$ may be null for some clusters. The labeling is done by the resident of the home after looking at the other attributes of the cluster. Once a cluster is labeled by the user, all the instances of that cluster are automatically labeled as that activity. In this way, users need to label only a small number of clusters instead of labeling all the instances of all the activities.

IV. Activity Recognition

Once the clusters are labeled as specific activities by users, our system can recognize future occupancy episodes as activities. During activity recognition, we construct room occupancy episodes from raw sensor firings as described in the previous section. As each room occupancy episode e is created, we take it's set of used sensors u_e and find the set of clusters $\{C_j\}$ such that $Used\_Sensors_j \subset u_e$ ($j \in \{$ 1, 2, 3, ..., *Number of Clusters* $\}$). There can be multiple such clusters. Because, the user may do multiple activities in the same occupancy episode or, even with the same set of used sensors, there may be multiple activities based on the temporal characteristics.

For each $\{C_k\} \subset \{C_j\}$ (j, $k \in \{$ 1, 2, 3, ..., *Number of Clusters* $\}$) such that all the clusters in $\{C_k\}$ have the same $Used\_Sensors$ = u_l ($u_l \subset u_e$), we construct the tuple ($startTime_{el}$, $duration_{el}$). Here, $startTime_{el}$ is the earliest timestamp when any sensor $s_e \in u_l$ fires during occupancy episode e. Suppose, $endTime_{el}$ is the latest timestamp when any sensor $s_e \in u_l$ fires during occupancy episode e. $duration_el$ is defined as the difference between $endTime_{el}$ and $startTime_{el}$. From the set of clusters $\{C_k\}$, we find the cluster that is temporally nearest to the tuple ($startTime_{el}$, $duration_{el}$) and also has the tuple within its neighborhood radius. We assign the episode ($startTime_{el}$, $duration_{el}$) the activity label of that cluster. If we do not find any such cluster, then we report this episode as an irregular one.

The cluster $C_i \in \{C_k\}$ is temporally nearest to the tuple ($startTime_{el}$, $duration_{el}$) if ($startTime_i$, $duration_i$) is the nearest point from ($startTime_{el}$, $duration_{el}$). This helps in differentiating among activities that trigger the same set of sensor firings, but are done in different times of day or last for different durations. Note that, multiple activities can take place in the occupancy episode in a room, and we accommodate such scenarios by assigning activity labels to each episode of $\{(startTime_{el}, duration_{el})\}$.

V. Evaluation

A. Dataset

Sensor Location	Symbol	Sensor Location	Symbol
Microwave	a	*Dishwasher*	h
Bathroom Door	b	*Toilet Flush*	i
Toilet Door	c	*Freezer*	j
Cups Cupboard	d	*Pans Cupboard*	k
Fridge	e	*Washing machine*	l
Plates Cupboard	f	*Groceries Cupboard*	m
Frontdoor	g	*Bedroom Door*	n

TABLE I: List of Sensors in the Dataset

We use a publicly available dataset ([20]), in which a wireless sensor network was used in a single–resident home to observe the resident's behavior and annotation was done by the resident using a bluetooth headset. We use this dataset, because it has 26 days of data which no other publicly available dataset has and also because it has a variety of sensors in different rooms. The list of sensors and their corresponding symbols are shown in Table I. The dataset contains a total of 16 activities. They are: 'Sleep', 'Eating', 'Prepare Breakfast', 'Prepare Dinner', 'Get Drink', 'Get Snack', 'Load Dishwasher', 'Unload Dishwasher', 'Load Washing machine', 'Unload Washing machine', 'Store Groceries', 'Use Toilet', 'Take Shower', 'Brush Teeth', 'Leave House', and 'Receive Guest'. Two of these activities ('Eating' and 'Store Groceries') have only one occurrence in the 26 days, so we do not consider these two activities. There are four rooms: *Bedroom*, *Kitchen*, *Toilet*, and *Shower*. We consider out of home as another room named *Outside* where the only sensor is the 'Frontdoor' sensor. All the sensors are binary sensors that include reed switches to measure open-close states of doors and cupboards, and float sensors to measure the toilet being flushed.

B. Clustering Accuracy

To show the accuracy of clustering, we train our system with 26 days of data. Accordingly, for each room, we get a number

Room	Num. of Clus-ters
Bedroom	1
Toilet	4
Shower	1
Kitchen	12
Outside	1

TABLE II: Number of Clusters for Each Room

Activity	Num. of Instan-ces
Sleep	24
Prepare Breakfast	20
Prepare Dinner	9
Get Drink	20
Get Snack	12
Load / Unload Dishwasher	9
Load / Unload Washing Machine	7
Use Toilet	114
Take Shower	23
Brush Teeth	16
Leave House	34
Receive Guest	3

TABLE III: Instances per Activity

of clusters (shown in Table II) each of which corresponds to an activity that is performed in the room. From this table we see that there are a total of 19 clusters. Users only need to label these 19 clusters as activities after we generate the clusters. If we use any supervised learning system, then user has to manually label all instances of all activities (291 in total as shown in Table III). In this way, our system ensures feasibility of long term training. Note that, each cluster contains a number of instances of the activity it represents; as a threshold in DBSCAN, we set the minimum number of instances in each cluster to 15% of the total number of days in the dataset (i.e., four days). Also, in the itemset mining algorithm we set the *support threshold* as 15% of the total number of days. These thresholds ensure that each candidate cluster has to have activity instances in at least four different days.

Table IV shows the set of clusters generated for *Kitchen*. We label each cluster by ourselves, based on the sensors, time of day and duration of each cluster. From this table, we see that some activities have multiple clusters. Cluster 2 and 3 have the same sets of sensors, but they are differentiated based on their temporal characteristics. Similar is the case with Cluster 5 and 8. Some of the clusters (7, 10, 11 and 12) do not have any temporal regularity in start times; however, the durations of their instances are similar. Note that, some of the activities are overlapped within the same occupancy episode (e.g., 'load / unload dishwasher' is sometimes overlapped with either 'prepare breakfast' or 'prepare dinner'). Our system can still identify 'load / unload dishwasher' as a separate activity.

Now, we evaluate the improvement in training due to our techniques of merging successive occupancy episodes that belong to the same activity. Firstly, we train on the raw data; we refer to it as *Normal Clustering*. Secondly, in the raw data, we merge successive occupancy episodes that belong to the same activity, and then perform mining and clustering on the changed data; we refer to it as *Clustering after Successive Occupancy Episodes Merging (SOEM)*. Fig. 2 shows the performance of training under these two configurations for the activity 'Sleep'. The x-axis shows day of month and the y-axis shows the durations (in hours) of the sleeping episodes as detected by the clustering methods and also as recorded by the resident in the ground truth.

From Fig. 2, we see that for *Normal Clustering*, many instances of the 'Sleep' activity are not included in the 'Sleep'

Cl. ID	Used Sensors	Mean Start Time (HH:MM)	Std. Start Time (Min.)	Mean Duration (Min.)	Std. Dur. (Min.)	Probable Label
1	{e, f, m}	9:25	70	1.76	1.65	prep. breakfast
2	{a, e, j}	9:12	65	6.59	3.78	prep. breakfast
3	{a, e, j}	18:50	45	30.24	9.25	prep. dinner
4	{d, f, m}	19:20	60	40.38	17.84	prep. dinner
5	{j, m}	19:30	40	20.65	5.62	prep. dinner
6	{f, k}	19:25	45	10.54	3.29	prep. dinner
7	{d, e}			3.98	2.57	get drink
8	{j, m}	21:54	90	1.71	1.28	get snack
9	{a, f}	20:12	150	2.31	0.75	get snack
10	{l}			4.23	2.39	use wash. mach.
11	{f, h}			4.25	1.5	use dishwasher
12	{h, k}			4.76	2.45	use dishwasher

TABLE IV: Set of Clusters for Kitchen

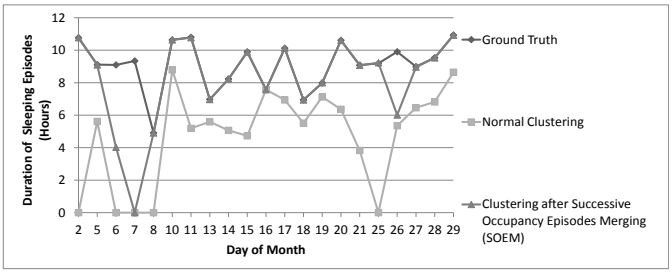

Fig. 2: Comparison of different instances of the cluster corresponding to 'Sleep' with the actual instances of 'Sleep' as recorded in the ground truth.

cluster ($2nd$, $6th$, $7th$, $8th$, and $25th$) as indicated by durations of zero for the corresponding nights. However, *Clustering after Successive Occupancy Episodes Merging (SOEM)* shows that merging successive occupancy episodes in the *Bedroom* improves the performance; four of the five missing instances of 'Sleep' are included in the 'Sleep' cluster. This is because during some nights, the resident goes to the toilet in the middle of sleep and comes back and sleeps again. If we do not combine the occupancy episodes before and after this toilet visit, then the individual episodes do not have the start time or duration characteristics of 'Sleep'.

$$TSE_a = \frac{\text{Total durations of } \mathbf{a} \text{ that remain undetected}}{D_a} \quad (1)$$

Due to lack of space, we do not discuss each activity in detail. For each activity $a \in A$ (A is the set of all activities), time slice error TSE_a is defined in Equation 1. Here, D_a represents sum of durations of all the instances of activity a in the dataset. Fig. 3 shows the TSE_a for each activity a. We do not show the training error for the activity 'Brush Teeth', because there were no sensors to recognize this activity. From this figure, we see that due to merging successive occupancy episodes, time slice errors improve only for the activities 'Sleep' and 'Prepare Dinner'. The reason is that in our dataset, only these two activities have large durations and other activities are overlapped only with them.

The results presented in this subsection show that AALO

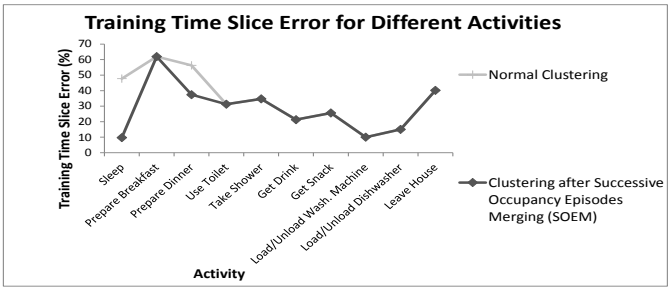

Fig. 3: Training time slice error for each activity.

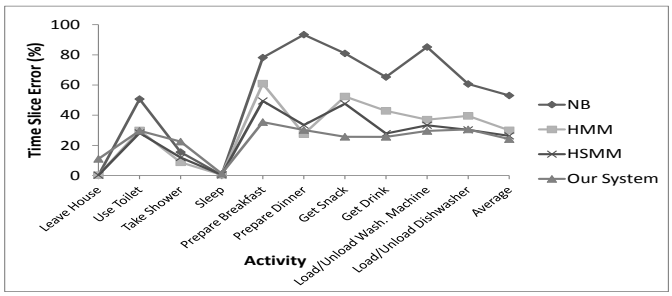

Fig. 4: Comparison of Time Slice Error for Activity Recognition among different classifiers.

groups different activities of daily living that have spatial and temporal regularity, and trigger similar group of sensors to fire under different clusters. Moreover, by considering the practical problem of temporally overlapped activities in different rooms, AALO improves the performance of clustering. Also, by including set of sensors used as a feature for clustering, AALO can differentiate among activities that have similar temporal and spatial regularity, but use different objects.

C. Evaluation of Activity Recognition Accuracy

To evaluate the activity recognition performance of AALO, we perform leave one day out cross-validation on the 26 days of data in the dataset. The metric for comparison is time slice error TSE_a for each activity as defined in Equation 1. We take the average of the time slice errors from cross-validation. During each step of cross-validation, we train our system with 25 days of data which generates a set of clusters; then we use the trained system on the remaining day to label its room occupancy episodes as instances of the clusters as described in Section IV. Thus we get the activity labels and we compare them with the ground truth to calculate the time slice error.

Table V shows the confusion matrix for all activities (percentage values). The values represent average over all runs in cross-validation. There are two extra columns in the matrix: 'Idle' and 'Irregular'. 'Irregular' corresponds to the cases when a frequent itemset is not within the neighborhood radius of any cluster. This may happen when an activity happens in an unusual time or for an unusual duration (compared to the corresponding training set). 'Idle' corresponds to the cases when our system infers that no activity is going on.

As we can see from Table V, many time slices of some activities are mistakenly predicted as 'Idle'. One reason for this is that we classify sub-periods of occupancy episodes (the duration for which a subset of sensors fire) as activities as opposed to classifying the whole occupancy episode as one activity. Also, in some time slices, according to the ground truth user starts an activity. However, corresponding sensors start to fire after some time slices.

We compare the performance of AALO with Naive Bayesian (NB), Hidden Markov Model (HMM) and Hidden Semi Markov Model (HSMM) classifiers presented in [20], [19] where the authors divide raw sensor data into fixed length time slots and classify each slot based on the sensors fired within that slot. We set the time slot length of their system to 60 seconds. They convert the raw sensor data to different formats; we compare with the 'last' format where once a sensor fires, it is considered to be firing until a different sensor fires. We choose this format, because it gives the highest accuracy for their system. There is another format named 'changepoint + last' which gives higher accuracy, but that code is not publicly available. Fig. 4 shows comparison of the average time slice error of our system with them.

For some activities ('Leave House', 'Take Shower', 'Sleep'), some instances take place in unusual times (e.g., the resident leaves house in midnight, or takes a shower in the evening, or goes to sleep at 3 AM); such instances occur rarely in the entire dataset. Therefore, our system considers these instances as outliers which makes the time slice error higher for these activities. NB, HMM and HSMM perform well for these activities, because they only depend on a single sensor and these algorithms take decisions based on those particular sensor firings ignoring temporal characteristics. NB performs much worse than others, because it classifies each time slot independently without considering the previous activity or durations of the current / previous activity.

Overall, Fig. 4 shows that the average of time slice errors over all activities for our system (24.15%) is much lower than NB (53.04%), and almost similar to HMM (29.97%) and HSMM (26.29%). In spite of being an active learning based approach, our system performs as good as the state of the art supervised activity recognition systems. None of these supervised techniques take time of day into account during activity recognition. If they are implemented in a way that time of day is also considered, then their performance should further improve. For example, there would be fewer errors due to confusion between 'prepare breakfast' and 'prepare dinner'. In that case AALO may perform slightly worse. However, AALO has the benefit of using significantly fewer number of user-labeled activity instances in the training set.

VI. Discussion and Limitations

We have evaluated our system only on one dataset. However, our techniques can be generalized to be applied to any dataset from a single person multiple room home. The occupancy threshold value used in the preprocessing step is automatically calculated from the training data for each dataset. The *support threshold* used in the itemset mining algorithm and the threshold for DBSCAN algorithm are set to be 15% for this

	Leave House	Use Toilet	Take Shower	Sleep	Prepare Breakfast	Prepare Dinner	Get Snack	Get Drink	Washing Machine	Dish washer	Idle	Irregular
Leave House	89.9	0	0	0	0	0	0	0	0	0	0	10.1
Use Toilet	0	70.3	0	0	0	0	0	0	0	0	14.2	15.5
Take Shower	0	0	77.5	0	0	0	0	0	0	0	4.2	18.3
Sleep	0	0	0	98.7	0	0	0	0	0	0	0	1.3
Prepare Breakfast	0	0	0	0	64.5	0	4.2	6.3	0	5.8	10.4	8.8
Prepare Dinner	0	0	0	0	0	69.8	2.4	2.9	0.5	1.3	13.5	9.6
Get Snack	0	0	0	0	0	10	74.3	4.6	0	0	0	11.1
Get Drink	0	0	0	0	11.1	9.4	5.2	74.3	0	0	0	0
Washing Machine	0	0	0	0	0	10	0	0	70.4	0	0	19.6
Dishwasher	0	0	0	0	4.6	20.5	0	5.5	0	69.4	0	0

TABLE V: Confusion Matrix for all activities (percentage values). The rows represent actual activities and columns represent the predicted activities. An entry in row *x* and column *y* represents percentage of time activity *x* was recognized as activity *y*.

dataset which ensures that out of the 26 days in the dataset, a cluster has to have instances in at least 4 days for it to be considered as regular behavior. These thresholds should be set according to the size of the dataset; if we have only 7 days of data then threshold set to 15% would mean that a cluster can even contain an activity with only one occurrence which is not ideal. Because our system utilizes the regularity in human behavior, it is more suitable for large datasets.

Our system works for single person homes. We plan to extend our system for multi person homes in the future. It will be straightforward if each sensor firing is associated with the corresponding user who triggers it. If such data association is not possible, we can use the variation in temporal characteristics of how different users perform an activity. Another limitation is that the segmentation step assumes that there are multiple rooms in the home and will not work if there is only one room. In such cases we need to design new techniques for segmentation. Also, as results show many activity instances are considered as irregular. However, the user may start to do an activity in a different way after the training. Therefore, we will develop ways to incorporate new trends in behavior by periodically re-training the system.

VII. Conclusion

AALO ensures ease of use in practical deployments, because it does not need ground truth for all instances of all activities during training. We address overlapping activities in two ways: 1) in the preprocessing step, we identify if an activity consists of multiple occupancy episodes with another activity overlapped in between them; and 2) in the itemset mining step, we identify multiple activities temporally overlapped in the same occupancy episode. The performance of our system is as good as the state of the art supervised activity recognition systems (HMM, HSMM).

VIII. Acknowledgements

This research was funded, in part, by NSF grants IIS–0931972 and EECS–1035303.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *VLDB*, 1994.

[2] T. S. Barger, D. E. Brown, and M. Alwan, "Health-status monitoring through analysis of behavioral patterns," *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, vol. 35, no. 1, 2005.

[3] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall, "Recognizing daily activities with rfid-based sensors," in *UbiComp*, 2009.

[4] T. Dimitrov, J. Pauli, and E. Naroska, "Unsupervised recognition of adls," in *SETN*, 2010.

[5] D. Elbert, H. Storf, M. Eisenbarth, O. Ünalan, and M. Schmitt, "An approach for detecting deviations in daily routine for long-term behavior analysis," in *PervasiveHealth*, 2011.

[6] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996.

[7] T. Gu, S. Chen, X. Tao, and J. Lu, "An unsupervised approach to activity recognition and segmentation based on object-use fingerprints," *Data Knowl. Eng.*, vol. 69, no. 6, 2010.

[8] T. Gu, Z. Wu, X. Tao, H. K. Pung, and J. Lu, "epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition," in *PerCom*, 2009.

[9] M. R. Hodges, N. L. Kirsch, M. W. Newman, and M. E. Pollack, "Automatic assessment of cognitive impairment through electronic observation of object usage," in *Pervasive*, 2010.

[10] D. H. Hu and Q. Yang, "Cigar: concurrent and interleaving goal and activity recognition," in *AAAI*, 2008.

[11] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. Intille, "A long-term evaluation of sensing modalities for activity recognition," in *UbiComp*, 2007.

[12] B. Longstaff, S. Reddy, and D. Estrin, "Improving activity classification for health applications on mobile devices using active and semi-supervised learning," in *PervasiveHealth*, 2010.

[13] M. Philipose, K. P. Fishkin, D. Fox, H. Kautz, D. Patterson, and M. Perkowitz, "Guide: Towards understanding daily life via auto-identification and statistical analysis," in *Ubihealth*, 2003.

[14] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report, 2009.

[15] M. Stikic and B. Schiele, "Activity recognition from sparsely labeled data using multi-instance learning," in *LoCA*, 2009.

[16] E. M. Tapia, T. Choudhury, and M. Philipose, "Building reliable activity models using hierarchical shrinkage and mined ontology," in *Pervasive*, 2006.

[17] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *Pervasive*, 2004.

[18] T. van Kasteren, G. Englebienne, and B. Kröse, "Transferring knowledge of activity recognition across sensor networks," in *Pervasive*, 2010.

[19] T. van Kasteren, G. Englebienne, and B. Krose, "Activity recognition using semi-markov models on real world smart home datasets," *J. Ambient Intell. Smart Environ.*, vol. 2, no. 3, 2010.

[20] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *UbiComp*, 2008.

[21] G. Virone, "Assessing everyday life behavioral rhythms for the older generation," *Pervasive and Mobile Computing*, vol. 5, no. 1, 2009.

[22] P. Wu, H. Peng, J. Zhu, and Y. Zhang, "Senscare: Semi-automatic activity summarization system for elderly care," in *MobiCASE*, 2011.

[23] S. Zhang, S. McClean, B. Scotney, P. Chaurasia, and C. Nugent, "Using duration to learn activities of daily living in a smart home environment," in *PervasiveHealth*, 2010.

[24] H. Zheng, H. Wang, and N. Black, "Human activity detection in smart home environment with self- adaptive neural networks," in *ICNSC*, 2008.