Non-Intrusive Activity Detection and Prediction in Smart Residential Spaces

Aniruddha Patel
Department of Electrical and Computer Engineering
San Diego State University
San Diego, CA 92182
Email: anpatel@sdsu.edu

Chinmay Prabhudesai
Department of Electrical and Computer Engineering
San Diego State University
San Diego, CA 92182
Email: cprabhudesai@sdsu.edu

Baris Aksanli
Department of Electrical and Computer Engineering
San Diego State University
San Diego, CA 92182
Email: baksanli@sdsu.edu

***Abstract*—Non-intrusive human activity detection and prediction is an important and challenging problem in smart and pervasive spaces. The advantages of such a design are the reduced dependency on the users and fewer security/privacy concerns. However, these also make it difficult to effectively and accurately understand the activities in real-time. In residential spaces, this can be even a bigger challenge due to nonuniform space boundaries and multiple people sharing the space. In this paper, we present a system, that consists of both hardware and software components, capable of detecting and predicting human activities in a smart residential environment. Our system deploys a finite state machine-based activity detection with 96% accuracy in real-time. Afterwards, we use several machine learning methods to create an effective activity prediction framework. We demonstrate that we can achieve up to 98.5% activity prediction accuracy with 4ms delay, making it a perfect real-time system example. Since our smart and pervasive space implementation does not use any intrusive sensor or data acquisition unit (such as wearables, camera, or audio sources), we reduce the dependency to the user and potential security/privacy issues.**

I. Introduction

Smart and pervasive spaces are getting more widespread in various application domains due to advancements in technology and ease of deployment [1]. These spaces effectively combine wireless sensor networks, embedded systems, and recently big data management to provide crucial solutions to various domains including, wellness [2], health-care [3], education [4], security [5] etc. One crucial property of these smart spaces is that there are more interactions between the users and the hardware devices, deployed as a part of the smart space, compared to the traditional counterparts. These interactions usually occur as a part of the users' daily routines, and with clever data analysis they can be captured digitally. This activity digitization can be very important for some applications, e.g. health-care. Thus, it is important to timely detect and even predict these activities in smart space settings.

A common way to achieve activity detection in smart spaces is to use audio/video sources, e.g. [6], [7]. Although these methods provide good results, they lead to security and privacy concerns for the users. Also, audio/video analysis requires significant computation power, which makes it difficult to use in resource-limited, real-time smart spaces. Another method is to use wearable devices, such as small watches or bracelets that include accelerometers [8], [9]. The problem with this solutions is that it reduces the applicability of the smart space to only those that own wearable devices, and to times when the users actually wear those devices. Also, in residential smart spaces, wearable devices can cause discomfort and also they are usually not seen as a part of the regular residential life.

A big issue about with smart residential spaces is that these spaces are usually shared by multiple people [10]. As a result, it becomes very challenging to identify single person activities in these areas. This is also tied to the fact that the hardware devices in these spaces generally have very low computational power (e.g. sensors, small microcontrollers, etc.), thus they might not be reliable to collect continuous data and analyze them. The system designers should carefully choose the correct hardwareto preserve high accuracy with real-time properties. For example, high delays can be an issue with time-critical smart space applications, such as elderly health-care.

In this paper, we create a smart and pervasive residential space, where we can detect and predict the activities of the users in real-time without compromising their security and privacy. We achieve the latter property by deploying only ambient sensors (such as ultrasonic, infrared sensors, etc.). This way, our system also does not lead to user discomfort. Our system framework consists of these ambient sensors, and several small microcontroller units to collect and analyze the data. We perform careful analysis on sensor placements to reduce the non-uniformity of the residential space. Then, we create a finite state machine-based activity detection algorithm, which is capable of detecting common activities in our smart space (which is deployed in a kitchen) with up to 96% accuracy in real-time. We then, take our intelligent algorithm one step further and implement activity prediction. We leverage multiple machine learning algorithms, including logistic regression, support vector machines, and neural networks, and apply those methods on our collected data with around 20000 data points collected over three months. We achieve up to 98.5% prediction accuracy with 4ms delay. As a result, our non-intrusive smart space implementation can achieve highly accurate activity detection and prediction with low computation overhead, making it a perfect example of a real-time system and suitable for time-critical applications.

978-1-5386-7693-6/18/$31.00 ©2018 IEEE

II. Related Work

There are several previous studies that focus on detecting human activities in a given space or in an outside area. One subset of this work is indoor human localization. Zappi et al. [11] devise methods to track motion direction and distance between users and obstacles by analyzing analog signals from passive infrared (PIR) sensors. They use the analog signals from PIR sensors to detect the position of a person, by applying machine learning algorithms on the collected data. Zhang et al. [12] propose a detection algorithm which uses adaptive threshold with constant false alarm rate calculation and calculates the location of a person in an area equipped with PIR sensors. This algorithm works well for both indoor and outdoor environment. It considers the noise received by the PIR sensor and removes the noise with the help of filter. However, the work does not do activity detection. Narayana et al. [13] provide an in-depth documentation of PIR sensors. They demonstrate that PIR sensors can be used for tracking when used as an analog sensor. Song et al. [14] describe region based tracking algorithms for PIR sensors to detect the path of a person. Other tracking based approaches such as range-based and range-free are discussed in detail by Xiao et al. [15]. Teixeira et al. conduct a survey of the various types of sensors. They discuss the active/passive nature of sensor, how sensors measure presence, count, location, identity, and tracking. They also state the network densities for each sensor type [16].

Other studies demonstrate more in-depth activity analysis. Ghosh *et al.* [17] build a grid of ultrasonic HC-SR04 sensors. They apply machine learning on the collected data to detect activities. But this activity detection is not applicable to nonuniform spaces (such as most residential spaces). Wearable devices also play an important part in activity detection. Mannini *et al.* [8] use a single accelerometer sensor, Chawla *et al.* [18] use a single wrist-mounted module, and Casale *et al.* [19] use a single chest-mounted accelerometer sensor to detect activities. The disadvantage of wearable devices is that they might provide discomfort to the users and prevent their natural movements. Additional studies leverage audio/video sources to track the behavior of humans [20], [21], [22]. However, this creates security and privacy concerns among the users.

The machine learning methods used in these smart spaces also vary significantly. Osiegbu et al. use support vector machines for their autonomous smart home application [23]. Chua et al. consider supervised and unsupervised labeling for Markov Models using compression and text analysis [24] for human behavior recognition. Dong et al. use hidden Markov models [25] for their intelligent conference room. However, previous studies do not analyze the overhead of having these machine learning methods running on small embedded computers, making them suitable with edge computing [26].

III. System Components

Our system components consist of various off-the-shelf sensors (Breakbeam, PIR, Ultrasonic) and development boards viz. Raspberry Pi and Arduino. Our goal is to create a light-weight system that does not have a lot of hardware units and does not require computation-heavy operations. To achieve this, we create a hierarchical hardware deployment as shown in Figure 1. We use the Raspberry Pi board to be the brain of our system, which enables us to access the system remotely over the Wi-Fi network. It also acts as the bridge between the software components and the lower-level hardware components. The rest of this section discusses the hardware components, and software tools used to manage data.

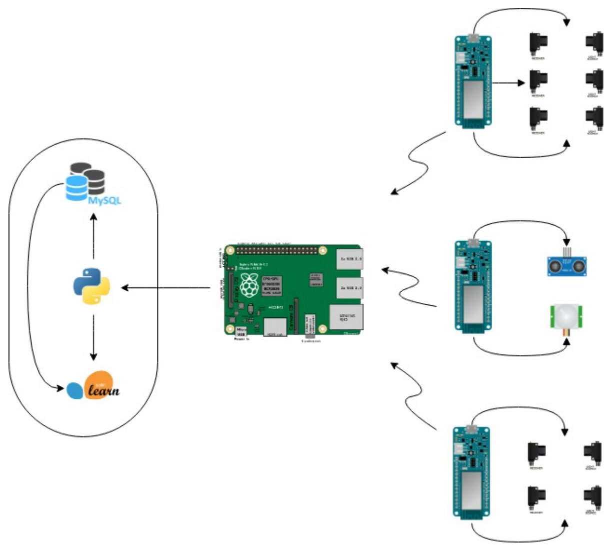

Fig. 1. Software/hardware block diagram

A. Hardware

*1) **Raspberry Pi 3**:* Raspberry Pi, as mentioned in previous context, allows us to have enough functioning power (of a small computer) but not really consumes or requires the same amount of resources as a full sized server or a general purpose computer. We use this board as the main computational unit in our system, that is connected to the other microcontrollers, as shown in Figure 1. Raspberry Pi is the highest level hardware unit in our hierarchical hardware deployment. We discuss the software components we leverage in the Raspberry Pi in detail further in sub-section III-B.

*2) **Arduino MKR1000**:* We use this microcontroller board to interface our sensors. Thus, we have to physically place these closer to the sensors. This board has a built-in Wi-Fi module which made it easier to connect to the network and transfer data over HTTP to the main Raspberry Pi node. It also supports running scripts over-the-air, which was very useful as it facilitates remote modification to the system even if the microcontroller itself was inaccessible directly.

*3) **Infrared (IR) Breakbeam Sensor**:* A Breakbeam sensor completes circuit when a certain amount of light is detected by the receiver. The IR gives a low signal when an object obstructs the path in between the transmitter and receiver and blocks the IR light transmitted by the transmitter from reaching the receiver. We use a pair of such sensors to determine the direction in which a person was moving into or to outside from our smart space. Another use in our deployment is to log refrigerator activity.

*4) **Ultrasonic Sensor (HC-SR04):*** These sensors are analog sensors based on Doppler-shift effect. An ultrasonic sensor sends a small pulse of signal and receives the pulse back after reflecting from a surface within its field of view. We use the time difference between sending a pulse and receiving one back to calculate the distance between the sensor and a person.

*5) **Passive Infrared Sensor (HC-SR501):*** These sensors might be used to detect extrinsic properties of people in a smart space, such as a thermal signature. They detect change in the heat signature around the sensor within a few meters range. If a person walks in, it gives a high signal denoting that there was a change in heat. If a person stays still after entering, the sensor adjusts and re-calibrates to that heat signature. It triggers again, once a person starts moving.

B. Software

This section focuses mainly on the different parts of software we use for the Raspberry Pi configuration and device setup. We use the default Raspbian operating system that is specifically designed for Raspberry Pi. Since the Raspberry Pi in our system acts as an access point, we set up an Apache web server, that we use concurrently with Flask using the WSGI (Web Server Gateway Interface) specification for convenient interface between the web-server and our Flask-based web-application. To store the sensor readings, we create a database with MySQL and we implement several server-side, Python scripts to receive data from sensors as well as use machine learning modules to perform data analysis. We use multiple software modules to implement the software side of our smart space deployment: 1) Flask: Web framework used to create web-pages for sending/receiving HTTP requests (MQTT: An alternative to HTTP was also tested), 2) MySQL connector: Module for Python, required interface between our application and the database to send and receive data, 3) JSON: JSON was used as the file format to transfer data objects between the numerous Arduino units and RPi, 4) Scikit-Learn (sklearn): A Python library used for machine learning algorithms.

IV. Activity Detection and Prediction Framework

In this section, we demonstrate the core methods we use to detect activities in our smart space and then how to predict the next activity in an efficient and accurate way.

A. Activity Detection

Fig. 2 shows the method in which we have interpreted our data to create a system on which we used various machine learning algorithms to predict the next action.

We model activities and activity transitions using a Finite State Machine (FSM) representation. Our FSM = $\{S_n, T_m\}$ has states (S) and transitions (T), where states represent distinct activities and transitions show the changes between activities. In our model, we have the following states (S): *Nothing, Entered, Using Refrigerator, Used Refrigerator, Near Burner, and Using Burner*; abbreviated as *N, E, UFr, Fr, ABr, Br* respectively. We developed an algorithm to determine the state transitions. States change from one to another, the moment a person enters the kitchen until the person leaves. These transitions (Table I) are governed by triggered sensors.

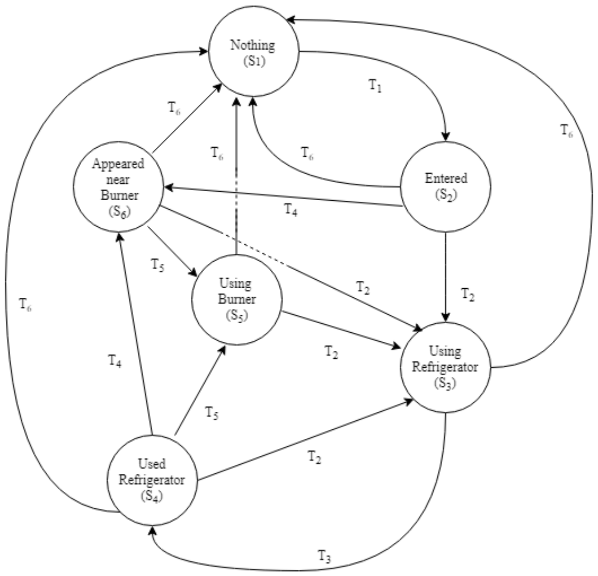

Fig. 2. State diagram showing activities and transitions
Note: Each state also has a transition to itself (Rule 8)

The state transitions occur based on the following rules:

- **Rule 1:** If nobody is in kitchen state is Nothing (N),

$$T_0 : N \rightarrow N$$

- **Rule 2:** When someone enters the kitchen, state changes from Nothing to Entered (E),

$$T_1 : N \rightarrow E$$

- **Rule 3:** If someone opens the refrigerator, state changes to using refrigerator (UFr),

$$T_2 : any\, state \rightarrow UFr$$

  Condition: Opening Refrigerator Door
- **Rule 4:** If someone closes the refrigerator, state changes to used refrigerator (Fr),

$$T_3 : UFr \rightarrow Fr$$

  Condition: Using Refrigerator (UFr)
- **Rule 5:** If someone comes near burner, state changes to near burner,

$$T_4 : any\, state \rightarrow ABr$$

- **Rule 6:** If someone stays near burner for more than x amount of time state changes to using burner,

$$T_5 : any\, state \rightarrow Br$$

  Condition: State changed to ABr recently and frequently

- **Rule 7:** If someone leaves the kitchen, state changes to Nothing,

  $$T_6 : any\,state \rightarrow N$$

  Condition: $p_{count} = 0$
- **Rule 8:** If someone enters with $p_{count} > 1$ in kitchen or leaves kitchen with $P_{count} < 1$ then state transition doesn't take place, also, for no transition, the FSM maintains current state,

  $$T_7 : any\,state \rightarrow self$$

These state transitions are recorded for each event spanning from state *Entered* (when person enters the kitchen) to *Nothing* (when person leaves the kitchen) by running our algorithm. Each of these states is then analyzed to determine the bigger activities, like cooking, eating, walking through the kitchen etc. Our FSM-based model can be easily scaled up with more activities, by defining the state and relevant transitions. As a future work, we are planning to detect new activities and transitions in the system automatically by analyzing the new, real-time data.

B. Activity Prediction

This section describes our activity prediction algorithm and how we use it to train multiple machine learning models to predict the next activity. Detecting an activity is different from predicting an activity. We use the rules defined in the previous subsection to infer/determine an activity from given set of sensor data points, while predicting an activity determines the most probable next activity following the current activity from the past activity patterns. This scheme can be used especially for time-critical applications, such as determining the dangerous situations for elderly-care, child-care, etc.

Once we detect the current activity (refer section IV-A), we predict the next most probable activity based on the current activity and activities occurred at the same time (the same minute and same hour but for date past the current date) from past data. For example, if person has just entered the kitchen, the prediction algorithm will predict what the person might do most likely (whether (s)he will open refrigerator, or (s)he will use burner) based on the past data.

We use information obtained from the sensors and the timestamps of data points to feed into the prediction framework. We select these features as they emphasize the past activity (past activity is a good indicator of the next activity) and the time that activity occurred. We combine this information with the previously collected data (stored in the sensor database) and cross-reference them. The prediction framework then uses a combination of this data to predict the next activity.

The following scenario describes our implementation. In this scenario, all the items have a *Before Transition* and *After Transition* state. The *Before Transition* state along with it's corresponding *Timestamp (t)* were used as our features and stored in a vector (V_1) . The second vector (V_2) contained the next immediate state following the state from the first vector. A list of vectors was created for each activity. This list or an array was then used to train a machine learning model.

Activity Scenario: Consider a person *"A"* cooking in kitchen. We describe the states of the kitchen for this person. All parameters from Table II are initially zero.

- Initially, the kitchen is empty **(Rule 8)**,
  Before Transition: *Nothing (N)*
  After Transition: *Nothing (N)*
  Parameter(s): $p_{count} = 0$ (refer Table II)
  Prediction Parameter: V_1 = [t, *N*]; V_2 = [*N*]
- At some time t_1, assume that *"A"* enters kitchen which triggers an event 'someone entering kitchen' **(Rule 2)**,
  Before Transition: *Nothing (N)*
  After Transition: *Entered (E)*
  Parameter(s): p_{count} += 1 (refer Table II)
  Prediction Parameter: V_1 = [t, *N*]; V_2 = [*E*]
- Now at time t_2, *"A"* goes near burner for cooking which triggers an event 'someone near burner' **(Rule 5)**,
  Before Transition: *Entered (E)*
  After Transition: *Appeared Near Burner (ABr)*
  Parameter(s): b_{count} += 1 (refer Table II)
  Prediction Parameter: V_1 = [t, *E*]; V_2 = [*ABr*]
- As *"A"* is cooking, (s)he moves around the kitchen to gather ingredients, generating a series of repeated events 'someone near burner', causing variable b_{count} to increase by 1 each time. If these repeated events occur for more than two minutes from time t_2, then, the system confirms that *"A"* is indeed cooking **(Rule 6)**,
  Before Transition: *Appeared Near Burner (ABr)*
  After Transition: *Using Burner (Br)*
  Prediction Parameter: V_1 = [t, *ABr*]; V_2 = [*Br*]
- Now, assume that *"A"* opens the refrigerator **(Rule 3)**,
  Before Transition: *Using Burner (Br)*
  After Transition: *Using Refrigerator (UFr)*
  Prediction Parameter: V_1 = [t, *Br*]; V_2 = [*UFr*]
- Next, *"A"* shuts the fridge door **(Rule 4, 6)**,
  Before Transition: *Using Refrigerator (UFr)*
  After Transition: *Used Refrigerator (Fr)* $\rightarrow$ *Using Burner (Br)*
  Note: FSM returns to *Br* as it is a super-set of state Fr — i.e. *UFr* and *Fr* states fall under an activity of cooking which is represented by state *Br*
- After cooking, at time t_3, *"A"* leaves the kitchen **(Rule 7)**,
  Before Transition: *Using Burner (Br)*
  After Transition: *Nothing (N)*

A Finite State Machine helps the system to determine the current activity of a person in the kitchen. Moreover, it backtracks to the previous state by immediately returning to it.

V. EXPERIMENTAL SETUP

In this section, we demonstrate our experimental setup and the system parameters used in our experiments. To demonstrate the effectiveness of our activity detection/prediction framework, we build a smart space in a subset of a residential space, i.e. a kitchen. With this choice, our smart space deployment is also addressing the non-uniform space boundaries (i.e. we cannot apply grid-based solutions). Fig. 3

TABLE I
TRANSITIONS AND THEIR RESPECTIVE CONDITIONS

Transition	Condition before sensor trigger	Condition after sensor trigger
T1	$p_{count} == 0$	d_1 or $d_2 == 1$ and $p_{count} > 0$
T2	$r == 1$ and $p_{count} >= 1$	$r == 0$ and $p_{count} >= 1$
T3	$r == 0$ and $p_{count} >= 1$	$r == 1$ and $p_{count} >= 1$
T4	$u > 80$ and $b_{count} == 0$ and $p_{count} >= 1$	$u < 80$ and $b_{count} == 1$ and $p_{count} > 1$
T5	$u < 80$ or $u > 80$ and $b_{count} <= 2$ and $2min > time > 0min$ and $p_{count} >= 1$	$u < 80$ and $b_{count} >= 3$ and $time > 2min$ and $p_{count} >= 1$
T6	$p_{count} > 0$	$p_{count} == 0$
T7	$p_{count} >= 1$	$p_{count} >= 1$

TABLE II
TRANSITIONS AND THEIR RESPECTIVE CONDITIONS

Parameter	Definition
p_{count}	Number of persons in kitchen
b_{count}	Number of time person comes near burner
$time$	Time interval over which parameter value is seen
$u < number$ or $u > number$	Distance of a person from burner greater or lesser than some distance number
$r == 0$	Refrigerator door is open
$r == 1$	Refrigerator door is close
$d_1 == 1$	Someone enters in kitchen from door 1
$d_1 == -1$	Someone leaves kitchen from door 1
$d_2 == 1$	Someone enters in kitchen from door 2
$d_2 == -1$	Someone leaves kitchen from door 2

shows how we deployed various sensors and microcontrollers around the smart kitchen space. Note that we use the same figures to represent sensors and microcontrollers (Arduinos and Raspberry Pis) as in Figure 1. We deploy three Arduino boards and interface sensors with them based on the data required from that location to create a smart environment.

The sensors are placed in activity-dense areas, in other words, we choose the location of the sensors based on high probabilities of observing activities. For example, there are sensors i) in the entrances of the kitchen to keep count of number of people in the area and ii) close to the kitchen appliances (such as refrigerator, oven, microwave, etc.). This way, we are maximizing the probability of capturing an activity with the smallest number of sensors deployed.

In terms of hardware, we use the Raspberry Pi 3 device as a sink node, running a Linux-based operating system. It is responsible for fetching the data from other microcontrollers, process and store the data in a local database. We interface the sensors with three Arduino MKR1000 boards, that transmit data using HTTP and JSON data formats over the Wi-Fi

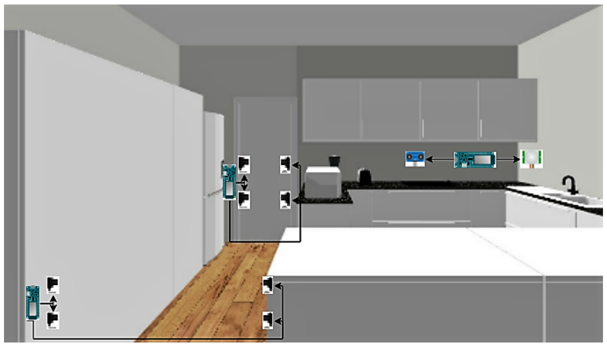
Fig. 3. Experimental Hardware Setup

TABLE III
ACTIVITY DISTRIBUTION IN GROUND TRUTH DATA POINTS

Activity	**Percentage**
Nothing	30%
Entered	20%
Using Refrigerator	12.5%
Used Refrigerator	12.5%
Appeared near Burner	15%
Using Burner	10%

network. In the entrances to the kitchen space, we placed two pairs of breakbeam sensors, to keep count of the number of people in the area. There is another breakbeam sensor by the refrigerator that keeps track of the status of the refrigerator (i.e. open vs. closed). Over the counter, we have an ultrasonic sensor placed near the burner oven to measure the activity related to it and a PIR sensor calibrated to measure any changes in heat or person movements in the area. Our deployment does not include a lot of hardware units and exhibit a hierarchical deployment schema, as shown in Figure 1.

To store the data for processing, we create a database, where sensor readings are stored with three fields: sensor ID, sensor value, and timestamp. The sensor ID field makes each sensor unique. We record activities in this smart space over three months, leading to a total of 19,890 samples. The distribution of data points with respect to different activities are shown in Table III. Each line in this table demonstrates the percentages of activities observed in ground truth data of 19,890 samples. Initially, the sensors have generated data points continuously, however, we realize that not all of these points are necessary. We apply data filtering at Arduino level, which later reduces the overhead Raspberry Pi would have. We also observe some noise in the sensor readings, and apply some data pre-processing methods to overcome this issue. This noise is mostly introduced because of the PIR sensor. We solve this issue by verifying the $p_{count} > 0$ condition before storing the PIR sensor values. If the sensors give any reading when the p_{count} is less than one, all the sensor readings are discarded.

And lastly, the python script running on the Raspberry Pi uses data from the local database and runs our FSM-based activity detection algorithm in real time and also applies machine learning techniques to predict the next most likely activity a

TABLE IV
ACTIVITY DETECTION PERFORMANCE

Activity	Ground Truth Samples	Correct Detection	Detection Accuracy
Nothing	100	96	0.96
Entered	100	96	0.96
Using Refrigerator	100	100	1.0
Used Refrigerator	100	100	1.0
Appeared near Burner	100	95	0.95
Using Burner	100	93	0.93
Total			0.96

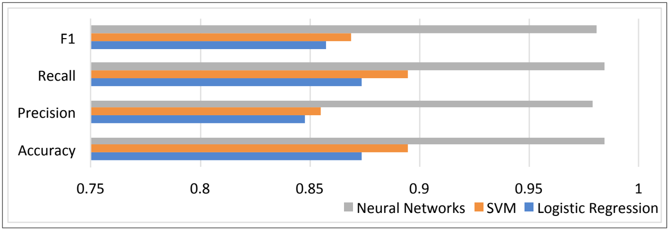

Fig. 4. Activity prediction results: comparison of different methods

person might do. We use three algorithms for this: i) logistic regression with parameters penalty=12, tolerance=0.01, inverse regularization strength=0.5, and class weight as auto, ii) support vector machines (SVM) with automatic class weight, and iii) artificial neural networks with default parameters in the *sklearn* module. Although most common implementations of SVM and logistic regression are binary, they both have multi-class classification models which leverage the One-vs-Rest implementation. Since our problem involves more than two classes, we use this multi-class classification.

VI. RESULTS

In this section, we present the detailed analysis and the results of our residential activity detection and prediction framework. For the activity detection part, we use our FSM-based algorithm. To run the experiments, we group the data points at a scale of per-minute interval. Our activity detection algorithm, that takes the grouped data points as input, generates a list of every transition change as output. We then calculate the activity detection accuracy in terms of how many times our algorithm can detect a specific activity. To measure activity detection accuracy, we collect ground truth data for every activity performed by the users for the first couple of weeks and compare the results with the output of our algorithm. To verify the accuracy of the detection algorithm on random days at random time, we collect ground truth activity data and compare it against activity given by the algorithm at that time. We collected around 600 ground truth samples. Table IV shows the analysis results, where each row corresponds to an activity. We see that the activity detection accuracy ranges from 95% to 100% with an average of 96.66%.

TABLE V
ACTIVITY PREDICTION PERFORMANCE COMPARISON

	Logistic Regression	SVM	Neural Networks
Accuracy	0.8738	0.8950	0.9847
Precision	0.8478	0.8551	0.9788
Recall	0.8738	0.8950	0.9847
F1 score	0.8573	0.8690	0.9810

For the prediction part, we use multi-class classification models on the data set we have. The models we experiment with include logistic regression, support vector machines, and neural networks. Each model takes a subset of our data set as the training set, and predicts the next activity on the test set. Finally, we compare the efficiency of these algorithms using four metrics, namely accuracy, precision, recall, and F1 score. These individual values demonstrate the classification of prediction results in terms of being correct vs. not. Furthermore, to ensure robustness in our experiments, we use a cross-validation method, where we perform up to 100 iterations of splitting the available data set into different training and test sample sets. We split the data in the standard 70:30 format, where 70% of the entire data set was used to train the model, and the remaining 30% was used to test the model. We calculate the above four metrics for each data split, and then compute the average of all 100 experiments. Figure 4 (Table V has the numerical values) outlines the results and compares the three methods we use to predict activities.

We observe that the there is a clear gap between different methods in terms of prediction performance, where logistic regression provides the worst and neural networks has the best results. This is consistent across all four performance metrics. Furthermore, we can see that the prediction accuracy can get as high as 98.5%, showing that the framework we have created can result in highly accurate prediction of human activity.

The next analysis we have is the computation overhead of these prediction methods. This is imperative to provide the prediction result to the user, or to another application that uses these results in a timely manner. This computation delay can be very crucial for time-critical applications, e.g. health-care. Another important factor regarding the computation overhead is that the smart and pervasive spaces use low-power and resource-limited embedded computers, such as the Raspberry Pi in our framework. To evaluate the real-time property of these prediction methods, we calculate the training and test time overhead on Raspberry Pi. Table VI shows the summary of this experiment. We see that logistic regression and support vector machines have very similar training and test times, whereas the neural network model is up to 10x and 60x slower than the other two in terms of training and test times, respectively. Depending on how time-critical applications are, the neural network model might have a serious disadvantage. At this point, support vector machines provide a good trade-off between accuracy and computation overhead. If the application

TABLE VI
TRAIN AND TEST TIME FOR THE ACTIVITY PREDICTION METHODS

	Logistic Regression	SVM	Neural Networks
Training Time (sec)	2.897	1.915	19.256
Testing Time (msec)	4.369	4.379	258.3

is not highly time-critical, neural networks is the best option.

Although it is difficult to compare our method directly to previous methods since the experimental setup along with the observed set of activities are not the same, we can still compare the accuracy of our work with a similar study [24], that does activity detection. We observe that in that work, the best accuracy is observed at 90%, where we can achieve up to 98.5%. Furthermore, the previous study does not analyze the computational overhead, thus we cannot comment on the real-time applicability of that work.

VII. Conclusion

Smart and pervasive spaces are gaining serious attention due to advances in technology and ease of access to cheap, commodity hardware. These spaces can provide useful applications in several domains including time-critical services, such as health-care, etc. One important challenge in these smart spaces is human activity detection in a real-time fashion. Traditionally, this activity detection was handled using multimedia data sources (such as audio and video). And recently, wearable devices also provide additional insight in activity detection. But rising security and privacy concerns prohibit audio/video devices. Furthermore, wearable devices result in too much dependency to the users, and may lead to an uncontrolled smart space. In this paper, we build a completely ambient smart residential space that is capable of detecting human activities in real-time. We devise an activity detection and prediction framework that can detect and classify activities up to 96% accuracy and predict the next activity with up to 98.5% accuracy. Our framework is also extremely lightweight, with as low as 4ms computation overhead while detecting/predicting the activities. Thus, our framework can be implemented in an edge-computing system, reducing the dependency to cloud resources, and increasing the overall local system resilience.

References

[1] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, “The gator tech smart house: A programmable pervasive space,” *Computer*, vol. 38, no. 3, pp. 50–60, 2005.

[2] H. Ghayvat, J. Liu, S. C. Mukhopadhyay, and X. Gui, “Wellness sensor networks: A proposal and implementation for smart home for assisted living,” *IEEE Sensors Journal*, vol. 15, no. 12, pp. 7341–7348, 2015.

[3] S. Amendola, R. Lodato, S. Manzari, C. Occhiuzzi, and G. Marrocco, “Rfid technology for iot-based personal healthcare in smart spaces,” *IEEE Internet of things journal*, vol. 1, no. 2, pp. 144–152, 2014.

[4] J. Aguilar, P. Valdiviezo, J. Cordero, and M. Sánchez, “Conceptual design of a smart classroom based on multiagent systems,” in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015, p. 471.

[5] J. Zhang, Y. Shan, and K. Huang, “Isee smart home (ish): Smart video analysis for home security,” *Neurocomputing*, vol. 149, pp. 752–766, 2015.

[6] J. Sung, C. Ponce, B. Selman, and A. Saxena, “Unstructured human activity detection from rgbd images,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 842–849.

[7] J. Ramırez, J. C. Segura, C. Benıtez, A. De La Torre, and A. Rubio, “Efficient voice activity detection algorithms using long-term speech information,” *Speech communication*, vol. 42, no. 3-4, pp. 271–287, 2004.

[8] A. Mannini, S. S. Intille, M. Rosenberger, A. M. Sabatini, and W. Haskell, “Activity recognition using a single accelerometer placed at the wrist or ankle,” *Medicine and science in sports and exercise*, vol. 45, no. 11, p. 2193, 2013.

[9] Z. Chen, Y. Chen, L. Hu, S. Wang, and X. Jiang, “Leveraging two-stage weighted elm for multimodal wearables based fall detection,” in *Proceedings of ELM-2014 Volume 2*. Springer, 2015, pp. 161–168.

[10] A. Benmansour, A. Bouchachia, and M. Feham, “Multioccupant activity recognition in pervasive smart home environments,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, p. 34, 2016.

[11] P. Zappi, E. Farella, and L. Benini, “Tracking Motion Direction and Distance With Pyroelectric IR Sensors,” *IEEE Sensors Journal*, vol. 10, pp. 1486–1494, September 2010.

[12] J. B. Zhiqiang Zhang, Xuebin Gao and J. K. Wu, “Moving Targets Detection and Localization in Passive Infrared Sensor Networks,” in *International Conference on Information Fusion*, no. 10. Quebec, Que., Canada: IEEE, December 2007, pp. 2788–2789.

[13] S. Narayana *et al.*, “PIR sensors: characterization and novel localization technique,” in *IPSN*, 2015.

[14] B. Song, H. Choi, and H. Lee, “Surveillance Tracking System Using Passive Infrared Motion Sensors in Wireless Sensor Network,” in *Information Networking, ICOIN 2008*, February 2008, pp. 1–5.

[15] Q. Xiao, “Range-free and range-based localization of wireless sensor networks,” Ph.D. dissertation, The Hong Kong Polytechnic University, Hong Kong, May 2011. [Online]. Available: http://ira.lib.polyu.edu.hk/handle/10397/4932

[16] A. S. Thiago Teixeira, Gershon Dublon, “A Survey of Human-Sensing: Methods for Detecting Presence, Count, Location, Track, and Identity,” Yale University, Tech. Rep., September 2010.

[17] A. Ghosh, A. Sanyal, A. Chakraborty, P. K. Sharma, M. Saha, S. Nandi, and S. Saha, “On automatizing recognition of multiple human activities using ultrasonic sensor grid,” in *Communication Systems and Networks (COMSNETS), 2017 9th International Conference on*. IEEE, 2017, pp. 488–491.

[18] J. Chawla and M. Wagner, “Using machine learning techniques for user specific activity recognition.” in *INC*, 2016, pp. 25–29.

[19] P. Casale, O. Pujol, and P. Radeva, “Human activity recognition from accelerometer data using a wearable device,” *Pattern Recognition and Image Analysis*, pp. 289–296, 2011.

[20] M. Cristani, R. Raghavendra, A. Del Bue, and V. Murino, “Human behavior analysis in video surveillance: A social signal processing perspective,” *Neurocomputing*, vol. 100, pp. 86–97, 2013.

[21] J. Sung, C. Ponce, B. Selman, and A. Saxena, “Human activity detection from rgbd images.” *plan, activity, and intent recognition*, vol. 64, 2011.

[22] S. S. Intille, K. Larson, E. M. Tapia, J. S. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson, “Using a live-in laboratory for ubiquitous computing research,” in *International Conference on Pervasive Computing*. Springer, 2006, pp. 349–365.

[23] C. Osiegbu, S. B. Amsalu, F. Afghah, D. Limbrick, and A. Homaifar, “Design and implementation of an autonomous wireless sensor-based smart home,” in *Computer Communication and Networks (ICCCN), 2015 24th International Conference on*. IEEE, 2015, pp. 1–7.

[24] S.-L. Chua, S. Marsland, and H. W. Guesgen, “Unsupervised learning of human behaviours.” in *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2011, pp. 319–324.

[25] B. Dong and B. Andrews, “Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings,” in *Proceedings of Eleventh International IBPSA Conference*, 2009, pp. 1444–1451.

[26] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.