Behavior Pattern Detection for Data Assimilation in Agent-Based Simulation of Smart Environments

Sanish Rai
Department of Computer Science
Georgia State University
Atlanta, USA
srail2@student.gsu.edu

Xiaolin Hu
Department of Computer Science
Georgia State University
Atlanta, USA
xhu@cs.gsu.edu

Abstract— Agent-based simulation is useful for studying human activity and their interactions in smart environments. Existing agent-based simulations are mostly offline tools that do not utilize real time information of smart environments. In previous work we developed a particle filter-based data assimilation method to assimilate real time sensor data from the environment into an agent-based simulation model. This paper extends the previous work by presenting a framework of behavior pattern informed data assimilation. We describe the structure of this framework and focus on the task of behavior pattern detection using Hidden Markov Model. We apply behavior pattern detection to a smart office case study example and discuss how the detected behavior pattern can inform the data assimilation in agent-based simulation of smart environments.

Keywords—Data assimilation; Particle Filters; Hidden Markov Model; Agent-based Simulation; Smart Environment component

I. INTRODUCTION

A smart environment is a physical area monitored by sensors with the goal of acquiring knowledge about the environment for improving the experience of its inhabitants. Increasing number of research works are being done in using smart environment to assist patients in hospitals, help old people in elderly home, optimize resources in buildings, track occupants for emergency evacuations or to automate tasks in the environment. In case of a smart office, the use of sensors to locate and track the movements of the inhabitants is important for developing emergency evacuation plans, scheduling energy resources like lights, computers, screens, networks, air conditioners and automatic doors. Agent-based simulation can be used to study the various components of a smart environment and the behavior of its inhabitants. Its bottom up approach to model the behavior of each entity and their interactions with their environment makes it suitable as a tool to simulate and design smart environment systems.

In the literature, different agent-based simulation models have been developed to model inhabitants’ behaviors such as walking, working in labs, eating, attending meetings in smart environments. Simulations based on these models are useful for understanding the activities of the inhabitants and their relation with the environment. Nevertheless, they are mostly used as offline tools and are not dynamically data driven in the sense that they do not utilize any real time data of the environment. A smart environment is integrated with various sensors that can provide real time information about the inhabitants present in the environment. This makes it possible for the simulation model to dynamically incorporate real time sensor data and serve as an online tool to support real time decision making in situations such as emergency evacuation. The online tool utilizes both sensor data information and the simulation model to “predict” the dynamics of the system in real time, and thus has a different purpose from traditional offline simulation-based studies (e.g., using agent-based simulation to carry out what-if analysis).

To achieve dynamic data driven simulation, data assimilation is needed. Generally, the real system’s states, which change over time, cannot be directly observed and is unknown to the simulation model. This makes the simulation start from a state different from the state of the real system, leading to inaccurate simulation results. Thus there is a need to dynamically estimate the “current” state of the real system and then feed the estimated states to the simulation model. This is achieved through data assimilation that utilizes real time sensor data for inference of the “current” system state. In previous work [1] we presented the use of data assimilation in a smart office environment to inference peoples’ occupancy information from sensor data. We used Particle Filters (PF) as the data assimilation algorithm to assimilate real time sensor data into a simulation model and achieved improved results for simulating movements of single and multiple agents in a smart office.

Data assimilation combines the observations (i.e., sensor data) of the current state of a system with the results from a prediction model (i.e., the simulation model) to produce an analysis. The results of data assimilation thus depend not only on the observation data, but also on the simulation model that “predicts” the evolution of system state. The simulation model runs in parallel with the real system by assimilating the sensor data, hence it is also important for the simulation model to be engaged in the same behavior pattern as that of the real system so it will simulate the evolution of system state with high accuracy. It is observed that complex spatial-temporal systems often exhibit distinct behavior patterns in both space and time. Considering a traffic system as an example, “traffic congestion” may refer to a behavior pattern where vehicles move in a stop-and-go fashion; while
“smooth traffic” refers to the behavior pattern where vehicles move freely and in fast speeds. Similarly, in an office environment, “having a meeting” refers to the behavior pattern where people stay in the conference room for the meeting; while “meeting break” refers to the behavior pattern where people move out of the conference room and take a break. Note that each of these behavior patterns characterizes a certain way of how the system behaves. Identifying the behavior pattern of a system from sensor data in real time can greatly inform the simulation model for more accurate simulation of the system under study.

Motivated by the above observation, in this paper we propose a new framework of behavior pattern informed data assimilation. We add a new component of behavior pattern detection from sensor data in real time and use the recognized behavior pattern to inform the PF-based data assimilation. The goal of the behavior pattern detection is to correctly estimate the behavior of a real system and engage the simulation with the same behavior as that of the real system. We use Hidden Markov Models (HMM) to carry out this work, where different behavior patterns are defined as a high level system state that needs to be inferred from observation data. The HMM is a statistical markov model in which the hidden system states are predicted based on the visible outputs. The advantage of HMM is that it can learn the parameters from the historical data and use it for state estimation in future. The recognized behavior patterns by HMM will influence the state evolution model (the simulation model) and thus have an impact on the PF-based data assimilation.

The remainder of the paper is organized as follows. Section II presents the related work for smart environment simulation, agent-based models and data assimilation. Section III presents an overview of the framework with behavioral pattern detection for data assimilation. In Section IV we present the experiment and results for a smart office case study. Section V concludes the paper and presents the future research direction for the proposed framework.

II. RELATED WORKS

Smart environment research focuses on utilizing the knowledge of environment’s context to accomplish tasks like occupancy estimation, navigation, user activities tracking, energy resources scheduling. Most of these researches use a physical environment with sensors to get the context data. The authors in [2] use ambient sensing data like lighting, acoustics, CO2, temperature and relative humidity to create a semi-Markov model which predicts the user behavior pattern and utilize in conserving energy resource of the building. Recognizing behaviors like cooking [3] and detecting crowd patterns using mobile sensing [4] could assist users for their activities in normal as well as emergency environment. Other researches include the problem of estimating the position or distribution of occupants [5, 6]. These researches require physical hardware and environment which are quite expensive. In [7] a software tool eHomeSimulator is presented which can be used to simulate smart environments and focuses on software engineering aspect of service development, configuration and deployment. In our research we use an agent-based simulation where we model the physical environment, binary sensors and occupant’s movements.

Agent-based models are well suited for smart environment simulations as agents can model the human and their characteristics naturally. Agents have been used in [25] to dynamically simulate user behavior in domestic settings and to identify the context, beliefs and facts impacting energy related behavior. The agent-based framework is used to develop energy efficient strategies implemented through social campaigns, ubiquitous computing or centralized and distributed approaches. In [26] a multi-agent simulation framework has been presented for the study of human behavior during building emergency evacuations. The framework includes a visual sensor which analyzes the environment and helps agents to make decisions. The system models emergent human behavior through simulating the behavior of human agents at microscopic level.

The method of data assimilation for incorporating observations finds its use in various fields of geosciences, weather forecasting, hydrology, and other environmental systems. The analysis techniques used for data assimilation include methods like three-dimensional variational analysis (3D-VAR), four-dimensional variational assimilation (4D-VAR), Particle Filters, Kalman Filters and others. A data assimilation system using 3D-VAR to improve ozone simulations in Mexico City basin is presented in [8] which generate the optimal estimate of the true atmospheric state during the analysis time. In another work, 4D-VAR is used to a regional ocean modeling system to produce an optimal estimation of the real ocean state using satellite remotely sensed observations [9]. Kalman filter [10] can be used to estimate the state of a dynamic system with observations represented by a linear state space model. In [11] three extensions of the Kalman filter; extended Kalman filter, limited extended Kalman filter and unscented Kalman filter has been presented to find the solutions to nonlinear discrete-time state-space.

The data assimilation algorithm used in this research is Particle Filters (PF). Particle Filters can be applied to dynamic systems with non-linear behaviors, by approximating the state of dynamic systems using particles and associated weights. A framework of dynamic data driven simulation based on PF is presented in [13] for the forest fire spread simulation. It presents a data assimilation framework based on PF to improve wildfire spread simulations using DEVS-FIRE, in which real-time data are fed into the DEVS-FIRE simulation to improve the accuracy of wildfire simulation. In another research [14], PF are used to develop a framework and algorithms to solve the problems of positioning, navigation, and tracking. PF find use in other fields like biology and chemistry as well. In [15] PF are used to create populations of compact long chain polymers and the relationships between packing density and chain length was studied. PF are used in [16] to set up a probabilistic framework providing a basic for process fault diagnosis for the dynamic data rectification.

In this paper we use Hidden Markov Model to dynamically detect the behavior pattern from the smart
environment. HMM has been used in [17] for real-time and scalable tracking of multiple users in a crowded environment using binary sensors. In [18] it has been used to identify activities from a series of sensor activations and the sensor activation sequences using sensors in household equipment. In another work, HMM is applied in a dense sensing approach that uses RFID sensor network technology to recognize human activities [19]. HMM is successfully applied to real-time detection as well, such as in [20] for audio-based chord recognition system in real-time using chroma features and in [21] for real time multiple target tracking.

III. A FRAMEWORK OF BEHAVIOR PATTERN INFORMED DATA ASSIMILATION

In this paper we present a new layer of behavior pattern detection on top of data assimilation and propose to use the detected behavior to inform the data assimilation. Fig. 1 shows the framework with the behavior pattern detection component. The major components of the framework include a real system, simulation model, data assimilation component and the behavior pattern detection component. The data assimilation component uses real time sensor data and the simulation model to infer the state of the system dynamically. On top of data assimilation is the behavior pattern detection layer which detects the behavior pattern of the system in real time. The detected behavior pattern is used to inform the simulation model so it can engage in the behavior pattern of the real system for improving data assimilation results. As can be seen, the framework separates the two layers that have different concerns: the simulation model captures the low-level dynamics of the system behavior; the behavior pattern detection component recognizes the high level behavior pattern to inform the simulation model. Below we describe the main components of the framework in detail.

![Figure 1. The Framework of Behavior Pattern Informed Data Assimilation](image)

A. The agent-based simulation model

We use an agent-based model to simulate people’s movements in the smart environment, where each agent represents an individual. An agent’s position at time $k$ is updated based on its position and velocity at time $k-1$. Given a smart environment, a waypoint graph is generated to model the navigating behavior of each agent. The intersections of corridors and rooms of the building are represented using the vertices of the graph. Using the waypoint graph a position in the floor can be connected to at least one of the vertices without engaging any obstacle.

The state space for the model is represented as:

$$\{l^i_k, v^i_k, d_k\}$$

where $l, v, d$ represent the location, velocity and destination of agent $i$ at time $k$ respectively.

![Figure 2. Floor structure with waypoint graph](image)

Fig. 2 shows an example of a waypoint graph associates to a certain floor structure. According to the waypoint graph, an agent will generate its route to a specific destination from its current position and add current position to the waypoint graph by connecting it to the nearest vertex such that the edge does not cross any obstacles (walls). Then the agent will find a shortest path from the starting position to the destination over the modified way point graph. The vertices along the generated route are stored as the intermediate route points to the final destination. By maintaining a list of intermediate route point, the speed vector of the agent is updated so that the moving direction of the agent always navigates to the final destination. The collision between agents is modeled by defining a comfortable distance for each agent. Agent will try to keep away from other individuals and reschedule a temporary route randomly to avoid potential collisions ahead.

B. Data assimilation component

Data assimilation is a technique to assimilate observation data to improve state estimation of the system under study [22]. We use PF for data assimilation which are a set of sample-based methods that recursively estimate the state of dynamic system from observation data using Bayesian inference and stochastic sampling techniques [12]. A key advantage of particle filters is their ability to represent arbitrary probability densities and to have little or no assumption about the structure of the system model. This makes it an effective method for supporting dynamic simulation with sophisticated simulation models. Meanwhile, PF are recursive methods that are able to recursively adjust their estimations of system states when new observation data becomes available. This feature is suited where new sensor data arrives sequentially and the simulation system needs to be continuously updated.

To carry out the data assimilation based on PF, we need to formulate the problem using a non-linear state-space model as shown by the system transition function in (2).
\[ S_{t+1} = SM(S_t, t) + v_t, \]
\[ Z_t = OM(S_t, t) + w_t. \]  

(2)

In the equation, \( S_t \) and \( S_{t+1} \) are the system state variables at time step \( t \) and \( t+1 \) respectively. \( Z_t \) is the observation variable representing the observations or measurements (sensor data). \( SM \) is the system transition model and defines the evolution of the system state. In our work, this system transition model is the simulation model. \( OM \) is the observation model and defines the computation of observation variable from the current system state. The \( v_t \) and \( w_t \) are random variables which refers to the noises of the system state and the observation data respectively. Based on the non-linear state-space model of (2), PF can be applied to estimate the new states at each time step by assimilating real time sensor data. In the implementation of PF, the system state is represented by particles, where each particle is a state candidate.

Fig. 3 shows the structure of PF and the procedure for data assimilation. In the figure, at time step \( t \), all particles’ states \( S_{t,i} \) from time step \( t-1 \) are input into the simulation model and evolve to a new set of state \( S_t \). Based on the new states, the observation model generates the new observations \( O_t \). These observation data are compared with the real observation data \( o_t \) and the importance weight of each particle is computed. The importance weights of all the particles are then normalized and the resampling algorithm draws a set of offspring samples \( S_t \) from \( S_t \) which has probability proportional to the importance weights. These set of resampled states will be the input for the next time step \( t+1 \).

If we take at the example of simulating occupants’ movement in a smart environment, the system state variables can be defined as the positions, speeds, and moving destinations for the occupants. The simulation model is the agent-based model defined in previous section. It defines how the system will evolve, i.e., how occupants will move based on their positions, speeds, and destinations. Assuming the sensors in this smart environment are binary motion sensors, then the observation model defines how the sensor data are computed from system state. Through this model, the binary sensor data are calculated based on occupants’ positions. The goal of data assimilation is to estimate the system state, e.g., occupants’ positions, based on binary sensor data. More details of PF-based data assimilation can be found in [1].

C. Behavior pattern detection component

The behavior pattern detection component uses real time sensor data to estimate the behavior pattern of the real system. In our framework, the task of behavior pattern detection is performed using methods of HMM. HMM determines the most likely system state based on the observations from a discrete-time, discrete-space system. The observation is generated by a process in the system whose actual state is hidden from the observer. According to the Markov property, the state at time \( t \), \( S_t \) is independent of all the states prior to the state at time \( t-1 \) i.e., \( S_{t-1} \). An \( N \)-state HMM having \( N \) observations can be defined as \( \lambda = (\pi, A, B) \) where \( \pi \) is the initial state probability matrix, \( A \) is the state transition probability matrix and \( B \) is the emission probability matrix. \( \Sigma = \{S_1, S_2, S_3, S_4, \ldots, S_N\} \) is the set of finite set of states and \( \Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \ldots, \omega_M\} \) is the finite set of observations. Now the HMM parameters can be defined as,

\[ \pi = \{\pi_i\}, \]  
where \( \pi_i = P(q_t=S_i), \quad 1 \leq i \leq N \)

\[ A = \{a_{ij}\}, \]  
where \( a_{ij} = P(q_{t+1}=S_j | q_t=S_i \}, \quad 1 \leq i, j \leq N \)

\[ B = \{b_k\}, \]  
where \( b_k = P(q_t=S_k | q_{t-1}=S_i \}, \quad 1 \leq k \leq M, \quad 1 \leq j \leq N \)

Before using the HMM in real time, it needs to be trained from the historical data of the system. The trained HMM is then used for detecting behavior states from real time sensor data. So the input to the HMM will be the sensor data in real time and output will be the recognized behavior patterns. For a given data set of observations, we use the Baum-Welch learning algorithm [23] to train the parameters of HMM \( \lambda \). After determining the HMM parameters for the system from historical training data sets, we use the maximum likelihood for each state and normalize it as the relative probability for each states in real time application. Specifically, we use the maximum likelihood algorithm described in the Forward algorithm [24] to select the state with maximum likelihood probability. We select the state with the maximum likelihood probability as the behavior of the real system at that time. Besides storing only the state with highest maximum likelihood we also normalize the probability of all the states and create a set containing states of the behavior pattern and their relative probability which is used by the simulation model. Each state calculates the maximum likelihood for each time step using the equations below:

\[ V_j(t)=\pi, b_j(\omega_1) \]  

(6)
where \( 1 \leq t \leq N \), and for \( 2 \leq r \leq T, 1 \leq j \leq N \)

\[
V_t(j) = \max[V_{t-1} a_j] b_j(o_t) \tag{7}
\]

where \( T \) is the total simulation time and \( V_t \) represents the maximum likelihood probability of each state at time \( t \). Since HMM is trained from historical data, there is limitation in obtaining maximum likelihood probability for behavior patterns when the pattern has not been previously observed.

### D. Behavior pattern informed data assimilation

![Behavior pattern informed data assimilation](image)

The basic idea of behavior pattern informed data assimilation is to utilize the information generated from the behavior patterns detection component to improve the PF-based data assimilation. Fig. 4 shows how the behavior pattern layer works together with the data assimilation layer in behavior pattern informed data assimilation. As can be seen, compared to the standard PF-based data assimilation (shown in Fig. 3), the new behavior pattern informed data assimilation goes through multiple iterations, each of which includes the sampling, weight computation, and resampling steps as in the standard PF algorithm. Nevertheless, the sampling step is informed by the behavior patterns detected from HMM. More specifically, in each step of the data assimilation, the HMM is first run to identify the “current” behavior pattern of the system. The output of the HMM is a set of behavior patterns and their relative probabilities. After that during the sampling step of the PF-based data assimilation, a particle first chooses a behavior by sampling from the behavior patterns (computed from HMM) according to their relative probability and then engages the simulation model in the chosen behavior pattern. In other words, the state evolution of the particle is based on a simulation model that models the chosen behavior pattern. Since the behavior patterns are identified in real time by the HMM, this sampling process should give more accurate results compared to when using a general simulation model that has no knowledge of the current behavior pattern of the real system. The other steps (weight computation, weight normalization, and resampling) of the data assimilation are the same as in the standard PF-based data assimilation.

### IV. SMART OFFICE CASE STUDY EXAMPLE

We apply behavior pattern detection to a smart office case study example. While our ultimate goal is to have an integrated system with both the data assimilation layer and the behavior pattern detection layer working together, the case study example in this section focuses only on the behavior pattern detection using HMM. In this case study we create a smart office environment using the agent-based model. With the agent-based simulation model we can easily create a number of occupants and create different scenarios. In particular, we are interested in a scenario of a conference event when agents move inside the conference room to start the conference and then move outside the conference room when the conference ends. The smart office environment is deployed with simple binary sensors that reports 1 as long as an occupant enters its sensing area and reports 0 otherwise. The binary sensor provides anonymous position information and cannot identify multiple occupants in its sensing area. The data collected by the sensor contains errors and is subject to environment clutter.

In this work we focus on correctly identifying the behavior patterns in real time. Sensor data are collected from the binary sensors and HMM is used to identify the behavior pattern states during a conference. The binary motion sensors are placed at the doors to capture the motion and are triggered only when users pass through their range. To use the HMM model, first we need to train the model to recognize the observations. For that we learn the HMM parameters from the historical data. We train the HMM for several scenarios (with data generated from simulations) of a general conference and learn the HMM parameters. We then use the trained HMM to estimate the behavior patterns in new scenarios.

To create training data set we create all the possible scenarios concerning the conference room where the users attend the conference. For a system consisting of an environment like a conference room, the behavior pattern will be behaviors like “entering the conference room”, “leaving the room”, “attending the conference”. The observational data are the real time data from the binary motion sensors. The behavior pattern is the states of the real system which are reflected by the sensor observations. For our experiment we consider six different behavior patterns (each is represented by a state in HMM): outside, inconference, few_entering, high_entering, few_leaving and high_leaving. The state outside represents the behavior pattern when there is no conference so that all the occupants are outside the conference room, the state inconference represents the behavior pattern when the occupants are inside the conference room for attending conference, the state few_entering represents the behavior pattern for a small number of occupants entering the conference room and high_entering represents a large number of occupants entering the conference room. Similarly low_leaving and high_leaving represents the behavior patterns when a small and large numbers of occupants leave the conference room, respectively.
It is a challenging task to extract high level information like behavior patterns directly from binary sensor data. So as a pre-processing we check the triggered rate of the sensors for a fixed sample time period. For the experiment we use a sample of 15 time steps and based on the amount of sensor triggered rate during that time sample, assign sensor data for each sample according to three values: zero sensor count, low sensor count, and high sensor count. Zero sensor count is the result of either when there is no conference so no occupants are entering/leaving the room or when the conference is happening so all users are seated inside the room. Low sensor count is the result of a small number of occupants (about one or two) entering or leaving the room. High sensor count is the result of either at the beginning or ending of the conference as the majority occupants enter or leave the conference room at the same time. A sample for sensor frequency data is shown in Fig. 6(a). At the beginning and end of a conference, majority of the occupants enter and leave the room but during the conference they do not move so no sensor are triggered.

We select some training data from historical sensor data and use the Baum-Welch algorithm to learn the HMM parameters. Then in real time, we used the learned HMM to predict the maximum likelihood of each state based on the real time sensor data. The state with the maximum probability estimates the behavior of the system. We can estimate the probability of each state to represent the possibility of each behavior pattern at the current time based on the observation. We utilize this information to predict a behavior pattern of the real system. Table I shows the initial state probability for each of the states and Table II shows the emission probability used for HMM computation. Fig. 5 shows the learned state graph and shows the transition probability between the states. With the learned HMM, we then use it to predict the behavior pattern of new scenarios. Several scenarios have been used to test the HMM, including one when all occupants attend a conference and leave at the end, and one when a few occupants enter the conference room to check if there is conference and leave the room as the room is empty.

**Table I. Initial Probability for HMM**

<table>
<thead>
<tr>
<th>States</th>
<th>Initial Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>outside</td>
<td>0.99</td>
</tr>
<tr>
<td>few_entering</td>
<td>0.002</td>
</tr>
<tr>
<td>high_entering</td>
<td>0.002</td>
</tr>
<tr>
<td>conference</td>
<td>0.002</td>
</tr>
<tr>
<td>few_leaving</td>
<td>0.002</td>
</tr>
<tr>
<td>high_leaving</td>
<td>0.002</td>
</tr>
</tbody>
</table>

**Table II. Emission Probability for HMM**

<table>
<thead>
<tr>
<th>Behavior States</th>
<th>Emission Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O(Zero)</td>
</tr>
<tr>
<td>outside</td>
<td>0.90</td>
</tr>
<tr>
<td>few_entering</td>
<td>0</td>
</tr>
<tr>
<td>high_entering</td>
<td>0.024</td>
</tr>
<tr>
<td>conference</td>
<td>0.90</td>
</tr>
<tr>
<td>few_leaving</td>
<td>0</td>
</tr>
<tr>
<td>high_leaving</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 5. Behavior Pattern States Transition Probability for HMM**

Using (7) we can compute the maximum likelihood probability for each time step and predict the current behavior pattern as shown in Fig. 6(b). In the figure, the blue line represents the real behavior of the state and the red line represents the estimated behavior of the system using HMM. The estimation is done in the real time when the sensor data is received. We can see that we were able to correctly estimate the behavior of the conference scenario and distinguish the conference state from the outside state, even though both of them give the observation of zero sensor count. Fig. 6(c) shows the normalized relative probability for the different states in real time. As can be seen, the probabilities for the conference state and the outside state dominate during the conference time and the outside time, respectively.

The experiment aims at evaluating the possibility of detecting behavior patterns from the sensor data and the evaluation of the accuracy is calculated as:

\[
\frac{\sum_{k=1}^{T} S_{i}^{k} - S_{i}^{real}}{T}
\]

where \(T\) is the total simulation steps and \(S\) is the behavior pattern state.
Figure 6. (a) Sensor frequency data (b) Comparing the real and predicted behavior (c) Normalized probability for the behavior pattern in real time

Table III shows the average accuracy for recognizing the behaviors from the observed sensor data using HMM. We assume that the behavior states always start from all users outside the conference room. From the results we see that we have a good accuracy for recognizing the behavior patterns from the noisy sensor data in the real time. The accuracy for behavior pattern _few_entering_ is low because the occurrence of that behavior is very low compared to other behavior patterns.

For this example, the behavior pattern when there is conference or not is successfully recognized with high accuracy. The information of relative probability as shown in Fig. 6(c) can be used by the data assimilation component for improving state estimation as discussed in the behavior pattern informed data assimilation framework.

**TABLE III. AVERAGE ACCURACY FOR BEHAVIOR PATTERN DETECTION**

<table>
<thead>
<tr>
<th>Behavior State</th>
<th>Average Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>outside</td>
<td>85.55</td>
</tr>
<tr>
<td>inconference</td>
<td>90.44</td>
</tr>
<tr>
<td>few_entering</td>
<td>18.40</td>
</tr>
<tr>
<td>high_leaving</td>
<td>75.01</td>
</tr>
<tr>
<td>few_leaving</td>
<td>63.35</td>
</tr>
<tr>
<td>high_leaving</td>
<td>71.07</td>
</tr>
</tbody>
</table>

V. CONCLUSION

Recognizing the behavior patterns of a system from sensor data in real time can provide useful information to improve particle filter-based data assimilation. In this paper we propose a framework of behavior pattern informed data assimilation and use a smart office case study example to show how the behavior pattern detection works. Future work includes combining the particle filter-based data assimilation with the behavior pattern detection component to improve dynamic data driven simulation.

REFERENCES


