Data Assimilation in Agent Based Simulation of Smart Environment

Minghao Wang
Department of Computer Science
Georgia State University
Atlanta, USA
csloader@gmail.com

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

ABSTRACT

A gent-based simulation is useful for studying people’s movement in smart environment. Existing agent-based simulations are typically used as offline tools that help system design. They are not dynamically data-driven because they do not utilize any real time sensor data of the environment. In this paper, we present a method that assimilates real time sensor data into an agent-based simulation model. The goal of data assimilation is to provide inference of people’s occupancy information in the smart environment, and thus lead to more accurate simulation results. We use particle filters to carry out the data assimilation and present some experiment results, and discuss how to extend this work for more advanced data assimilation in agent-based simulation of smart environment.

Categories and Subject Descriptors
1.6.0 [Simulation and Modeling]: Simulation types and techniques, Agent / discrete models, Data assimilation

General Terms
Algorithms, Design, Theory,

Keywords
Agent-based Model; Data Assimilation; Particle Filter; Smart Environment

1. INTRODUCTION

Location information of occupants in building structure is valuable for scheduling efficient energy utilization, delivering conditioning air or developing evacuation strategy during emergencies. There is an increasing research interest in designing Intelligent/Smart Building architecture and applications that utilize occupancy information to automate the building’s HVAC system. A gent-based simulation is a useful tool to study people’s movement in building structures (see, e.g., [2][8]). It uses a bottom up approach to model individuals’ behavior and their interactions with the environment. Over the years, different agent-based models with different levels of complexity have been developed, ranging from agents that model the basic movement behaviors (e.g., walking, avoiding obstacles) of pedestrians to agents that have psychological states and/or incorporate schedules (e.g., weekly working schedules) of different persons in an office environment. These simulations provide valuable information about the movement patterns of people in building structures and thus help the design of building structure and the development of egress strategies in emergency situations. Nevertheless, these simulations are typically used as offline tools that help system design. They are not dynamically data driven because they do not utilize any real time sensor data of the environment.

On the other hand, with the advances of sensor and communication technologies, more and more building environments are equipped with sensors that can provide real time occupant location information. In this paper, we refer to a smart environment as an indoor building environment equipped with sensors. In particular, we consider office environment on a building floor where people move between office rooms and corridors. With the real time data provided by sensors, how to assimilate these data into a simulation model becomes an important research topic. In this paper, we present a method that assimilates real time sensor data in agent-based simulation of smart environment. The goal of data assimilation is to provide an accurate inference of people’s occupancy information from sensor data and the simulation model of occupancy dynamics. Knowing this information in real time is useful for energy control. It also supports simulations that can more accurately simulate people’s movement behavior from the current time (because the simulation starts from a more accurate initial state, which is the system’s current state), and thus provide useful information in real time decision making in emergency evacuation.

The quality of data assimilation depends on both the sensor data and the agent-based model. Different types of sensor devices have been used in smart environment. Typically, the sensors are sparsely deployed and provide only limited information about the environment, and thus one cannot extract the system state directly from the sensor observation. Sensing device with high resolution such as a video camera or ultrasonic radar has relatively precise measurement; however information collected by such sensors might be intrusive, and they are more expensive than low resolution sensors. In this paper, we consider binary motion sensors that provide non-intrusive information (see section 2 for more details). The quality of agent-based model will also have an impact on the data assimilation results. In general, a higher quality model captures the dynamics of the system more comprehensively and thus directing the simulation to more accurate results. In this paper, we use a relatively simple agent-based model to simulate people’s movements. It is important to
There are several existing works related to estimating building occupancy dynamics from models. In [6], researchers combine the sensor data, structure of the building and prior knowledge about utilization of the building together to estimate the building occupancy. In [5], Tomastik utilizes a combination of two different types of sensors and applied Extended Kalman Filter to estimate zone level occupancy information. In [3], Liao proposed a novel estimation approach which relies on an agent-based model of occupants. Our work differs from these previous works by developing an integrated approach that combines filtering technology and agent-based model to estimate the building occupancy in smart environment. We also note that while our work shares some similarities with research in target tracking, our ultimate goal is not to identify and to track each individual targets. Instead, we are more interested in estimating the overall occupancy information, i.e., the spatial density distribution of occupants, and then use that information for more accurate agent-based simulation of the smart environment.

The data assimilation algorithm we used in this research is Particle Filter (PF). PF is a set of statistical methods aiming at using observable variable to compute the latent variable linked to a Markov chain in some Bayesian model to estimate the state of a dynamic system [1]. It has the advantage that it does not have the assumption about the Gaussian property of the system and it allows the targeting posterior distribution to be multi-modal. In a standard bootstrap filter discussed in [1], the proposal distribution from which samples are drawn is constructed using the transition prior; in our work, the transition prior is the agent-based model for simulating occupants’ movements.

The rest of this article is organized as follows: The framework is illustrated in section 2. Section 3 presents the experiment result and analysis. Conclusion and future work are described in section 4.

2. Data Assimilation in Agent-based Simulation

2.1 The smart environment and sensor

A smart environment is defined as any design that utilizes knowledge of the environment’s context to aid occupants in accomplishing various tasks such as navigation, scheduling energy consumption, notification of activities and evacuation planning. A typical smart environment is deployed with various sensors. The data obtained by the sensing devices is then fused into a center control system for handling configurations for energy supply, air conditioner or providing building information related to the current context to the occupants.

Existing research for smart environment investigates the problem of estimating the position or distribution of occupants. ([3][4][5]) In this work, we assume that the simulated environment is deployed with sensors of the simplest form, that is, binary sensor that reports 1 as long as an occupant enters its sensing area and report 0 otherwise. It has the lowest resolution on detecting the movement of occupants entering its detection range and has the following features in terms of resolution, accuracy and ambiguity:

1. It provides anonymous position information.
2. Information it gets is ambiguous when multiple occupants are in the sensing area of single sensor.
3. Data it collects contains errors and is subject to environmental clutter.
4. Information volume is highly sensitive to the density and spatial pattern of the deployment of the sensors.

We assume that the observations generated by such sensor follow its expected basic functions described above and the model of observation error is simple.

2.2 The agent-based model

Construction of the agent-based model involves a bottom-up procedure. Specifically, the dynamics of occupants in this work are represented by the aggregation of moving behaviors of each individual occupant; agents’ position at time k updates based on its current velocity and speed. A simple equation describing the physical law of position update may be used as the model of occupants’ moving behavior. However, such an equation-based approach does not specify how the agent’s speed and orientation are determined and fails to describe the effect of collision between agent and other object on the movement behavior.

We notice that people’s movements in commercial building are generally associated with clear purposes. We model this procedure by generating a waypoint graph for the navigating behavior of each agent. The vertices of the graph represent the intersections of corridors and rooms of the building floor. With this graph, an arbitrary position in the floor can be connected to at least one of the vertices in the waypoint graph without engaging any obstacle.

The state space is therefore represented as follows:

\[
\{l_k^i, v_k^i, d_k^i\}
\]

And the function for determine velocity of a single agent is:

\[
v_k^i = g(l_k^i, d_k^i) + Q
\]

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

Fig 1 shows an example of a waypoint graph associates to a certain floor structure.

![Fig 1, Floor structure with waypoint graph](image)

According to the waypoint graph, an agent generates its route to a specific destination from its current position with the following rules:

1. Add current position to the waypoint graph by connect it to the nearest vertex such that the edge does not cross any obstacles (walls in this particular example).
2. Replace agent’s current position with the destination of the agent in step 1 and repeat operations in step 1.
3. Find a shortest path from the starting position to the destination over the modified way point graph.
4. 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.

In addition to navigation, corridors and rooms in a typical building structure normally contain limited free space, which can easily result in congestion when the number of occupants is large. An agent-based model should be able to model the congestion behavior of occupants; the model otherwise fails to capture the important characteristic of people’s flow pattern in such environment. Specifically, in this work, the avoidance 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.

2.3 Data Assimilation Framework
The designed agent-based model can be used as the predictive model in a data assimilation framework. Data assimilation involves incorporating observations from sensing devices into a predictive model to adjust the simulation and get the best estimate of a system’s current state. Particle filter algorithms are suitable to deal with the state estimation problem in such scenario. To carry out data assimilation, we need to formulate the problem using a state-space model that contains a state transition equation (also referred to as the system transition model) and a measurement equation (also referred to as the observation model):

\[ x_k \sim p(x_k | x_{k-1}) \]
\[ y_k \sim p(y_k | x_k) \]

where \( x_k \) represents the posterior distribution at time k of system states evolved from states at previous time step \( x_{k-1} \) with some Markov process; and \( y_k \) represents likelihood function up to time k given posterior distribution at k. In particle filtering, the posterior state distributions are sampled into a number of weighted particles. The weights of these particles represent the likelihood of the particles. An additional resampling step is applied for eliminating particles with low weights and multiplying particles with high weights. Theoretically, by constructing an accurate easy-to-draw proposal distribution and utilizing sufficient number of particles, the estimated probability density function of system states will eventually converge to the posterior of the target system. In this paper, we draw samples only from the transition prior using the agent-based model. This means the transition function is then described as follows according to the model defined in section 2.

\[ S_{k+1} = a(S_k) + r_k \]

, where \( r_k \) is the noise and a is a state transition function described by the agent based simulation model described in section 2.

At each time step, samples are drawn by executing \( \Delta k \) steps of simulation based on the model:

\[ S_{k+\Delta k} = \text{est}(a, S_k, \Delta k) \]

2.3.2 The observation model
The observations in this research are assumed to be generated from a set of binary sensors which will produce 1 if there is at least one occupant entering its sensing area and produce 0 otherwise. The error rate of the sensing device is set to 0.05 so that the probability for a sensor to report a false reading is 5%.

Assuming there are \( m \) deployed sensors; the output of the measurement is a size \( m \) array containing 0s and 1s to indicate whether there are agents in the sensors’ detecting area.

To calculate the importance weight of a particle, the actual measurement vector \( SE_{\text{real}} \) is compared to the measurement vector of the sampled particle \( SE_{\text{particle}} \) in bitwise manner, with two observation array

\[ SE_{\text{real}} = [s_{1\text{real}}, ..., s_{m\text{real}}] \]
\[ SE_{\text{particle}} = [s_{1\text{particle}}, ..., s_{m\text{particle}}] \]

The importance weight (likelihood) \( w^\text{particle}_i \) of the particle is calculated as:

\[ w^\text{particle}_i = \prod_{j=1}^{m} \omega_j \]

, where

\[ \begin{align*}
\omega_1 &= a \quad s_{\text{real}}^j = 1, s_{\text{particle}}^j = 1 \\
\omega_2 &= \sqrt{a} \quad s_{\text{real}}^j = 0, s_{\text{particle}}^j = 0 \\
\omega_3 &= 1/a \quad \text{else}
\end{align*} \]

The importance weight represents the degree of approximation of the state of a given particle to the real state. Correct models for the noises \( r_k \) and \( z_k \) are important in constructing proposal distribution. For observation noise, we assume that the error rate is relatively low for the sensors and follows uniform distribution. The error rate of a sensor defines the probability a sensor reports false readings. For system noises, we add uniform distribution \( R^*u(0,1) \) to the positions of each agent, where \( R \) is a constant; we also add noise to agents’ destinations as described below.

Given the destination of the ith agent, \( d_c^i \), and a modifier \( lr \); the destination after adding noise is calculated as follows:
Initially, agents in N particles are assigned with positions uniformly and randomly over the 2D space. The proposal distribution is constructed by executing the simulation for certain amount of time steps, i.e., using the transition function to generate new samples. After this step, observation data from the binary motion sensor are utilized to calculate the likelihood of each particle. The likelihood is normalized to serve as the probability that a particle will be duplicated in the resampling step. In the selection step, we apply a roulette wheel selection mechanism and select N new particles based on their normalized weights. The algorithm then goes to the importance sampling step and repeats.

3. Simulations and experiments

To verify our methods and to evaluate the effectiveness of the proposed algorithm, we have conducted a series of simulations and experiments with different number of agents. The simulation is based on a simulator which implements the system model and observation model specified in previous sections. We employ the identical twin experiment where a simulation is chosen as the “real” system that generates sensor data to be assimilated. We estimate agents’ locations by assimilating these sensor data into the agent-based model and compare the estimated location information with that in the “real” system. In our experiments, the floor structure and sensor deployment information is shown in Fig 3. Initially, the positions and destinations of the agents are distributed uniformly over 800*300 2D space except the locations occupied by walls. Notice that in this research, the posterior is represented by an aggregation of all particles instead of a single one with the highest weight.

3.1 Experiment with Single Agent

This series of experiments aims at evaluating the robustness of the particle filter algorithm by tracking single agent; the evaluation of the accuracy is calculated as:

\[
\frac{\sum_{k=1}^{800} |l^{k}_t - l^{\text{real}}_t|}{800}
\]

We consider two scenarios: one is referred to as the regular route, where the tracked agent is designed to move forward through a sequence of destinations; the other is referred to as the turning back route, where the tracked agent is designed to move straight forward through a sequence of destinations, then turns back to move to its initial position. In both scenarios, the error rate of the sensor is set to 5%, which means the sensor has 5% probability to report 1 when it is supposed to report 0 and vice versa. Fig 4(a) shows these two scenarios.

Fig 3, the simulation environment with obstacles and sparsely deployed sensors. The yellow circle represents the sensing area of deployed sensors.

<table>
<thead>
<tr>
<th>Table 1. The Bootstrap Filter Tracking Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm 1 Bootstrap Filter Tracking Algorithm</strong></td>
</tr>
<tr>
<td>1. Initialization. t = 0</td>
</tr>
<tr>
<td>- For particle ( i = 1, ..., N ), randomly assigning agents’ positions and destinations in particle ( i ); generate routes</td>
</tr>
<tr>
<td>- Set ( t = 1 )</td>
</tr>
<tr>
<td>2. Importance sampling step</td>
</tr>
<tr>
<td>- Add variance to the position and destination of the agents.</td>
</tr>
<tr>
<td>- For particle ( i = 1, ..., N ) after ( \Delta t ) steps of simulation based on model described by the simulation model.</td>
</tr>
<tr>
<td>- For particle ( i = 1, ..., N ), calculate weights</td>
</tr>
<tr>
<td>- Normalize the weights</td>
</tr>
<tr>
<td>3. Selection step(resampling step)</td>
</tr>
<tr>
<td>- Resample with replacement N particles ((X^{(i)}<em>{0,t}; i = 1, ..., N)) from the set ((\hat{X}^{(i)}</em>{0,t}; i = 1, ..., N)) according to weights.</td>
</tr>
<tr>
<td>- Set ( t \leftarrow t + 1 ) and go to step 2.</td>
</tr>
</tbody>
</table>
Both of these two simulations share the same initial positions and sensor error configuration with 800 time steps of simulation. We run each of the simulations for 10 times and the error rates at each time step are averaged and plotted in Fig 4(b). In Fig 4(b) the vertical axis represents the error while the horizontal axis represents the iterations of the simulation. When $\Delta t$ is set to 4, 800 time steps simulation consumes 200 estimation iterations. The curve presents an oscillated wave form where the error increases when the agent enters the blind area and decreases when the agent enters the sensors’ detection area. The result shows that in the turning back route configuration, the error increases at the time interval where the agent is turning to an opposite direction. However, the tracking system was able to reduce this error soon after new sensor data arrive. This experiment shows that the method is working well for single agent.

3.2 Experiment with Multiple Agents

In this section, we conduct a series of simulations and experiments to compare the estimation effectiveness of the tracking algorithms to track two agents simultaneously using different number of particles. We set the route of the two agents spatially separated so that the two agents will keep a relatively long distance to each other. The initial positions of the two agents are set to be located at corridor area and have the maximum possible distance away from the nearest sensors. We modified the error measuring the accuracy calculation using 800 particles as follows:

$$\sum_{k=1}^{800} \left| \left| l_t(2)_{k} - l_t(2)_{real} \right| + \left| l_t(1)_{k} - l_t(1)_{real} \right| \right|$$

$$800 \times 2$$

The number of particles is changed from 800 to 400, 1200 and 1600 for evaluating the relationship between number of particles and the accuracy; we run each simulation for 350 time steps (95 simulation steps), average the errors of 10 simulations. This experiment is for verifying the effectiveness of the data assimilation method in multiple agent scenarios. We set the number of particles to 400, 800, 1200, 1600 and calculate average error rate from time 0 to time 350 for each of these experiments. The results are plotted in Fig 5 and Fig 6, which shows that as the number of particles increases, the average estimation error decreases.

In Fig 5, the vertical axis represents average estimation error in 10 simulation runs. For each run, the error is calculated by averaging the error of time step 0 to time step 350. The horizontal axis represents the number of particles initialized for estimation. When the number of particle is large, it is more likely that they will have lower estimation error. This is due to the fact that larger area of state space is covered for the initial state of the particles, therefore providing a more accurate starting point for subsequent simulations.

In Fig 6, the estimation errors over simulation time using different number of particles are presented. The error rate at a single simulation step is averaged from the error rates of 10 simulation runs at the same step. For $N=400$ and 800, due to the fact that in most of the simulations, the number of particles is insufficient to cover the region of the state space for providing an initial state that can lead to a global convergence; the algorithm can hardly converge to a density that is close to the posterior distribution. While the number of particle increases, this problem is solved because that larger area of the state space is covered for subsequent simulations.

The above experiment shows that there are certain limitations in the standard bootstrap filtering algorithm for inference of multiple
agents. As the number of agents increases further, we expect the standard bootstrap filtering algorithm will not perform well. This is because we consider the locations of all agents as the system state. Thus a particle needs to have the correct combinations of all agents' locations in order to be selected. However, due to the irregular arrival time of the sensor data as well as the nature of the sequential importance sampling, this is not guaranteed that such a particle exists. This is especially true when the number of particles is small. In order to deal with the increasing dimensions of state space while tracking multiple agents in future work we plan to develop an inter-state crossover resampling strategy. This strategy decouples different dimensions of the state space and regroups them together to form new particles. By doing this, combined states of agents in a multi-agent system are relaxed and the sample space is enlarged. To avoid breaking dependencies among agents in a particle which may result in corruption of the simulation for individual particle, we plan to integrate this cross-over strategy into the standard particle filter algorithm. The new method will maintain the integrity of the system state represented by the particles and at the same time to enlarge the sample space by reconstructing new particles utilizing the crossover strategy.

4. Conclusion and Future work

In this paper, we investigate the problem of data assimilation in agent-based simulation of smart environment with proximity sensors. We develop an agent-based model to represent the basic navigation and motion behavior of the occupants in building floor, and apply it in the particle filter algorithms to estimate the occupancy information of the environment at a given time. Several experiments are carried out in order to evaluate the accuracy and robustness of the algorithm. Future work will extend this work and develop more advanced methods to improve the data assimilation results.

5. Reference


