12-18-2014

Data Assimilation for Agent-Based Simulation of Smart Environment

Minghao Wang
Georiga State University

DATA ASSIMILATION FOR AGENT-BASED SIMULATION OF SMART ENVIRONMENT

by

MINGHAO WANG

Under the Direction of Xiaolin Hu, PhD

ABSTRACT

Agent-based simulation of smart environment finds its application in studying people's movement to help the design of a variety of applications such as energy utilization, HAVC control and egress strategy in emergency situation. Traditionally, agent-based simulation is not dynamic data driven, they run offline and do not assimilate real sensor data about the environment. As more and more buildings are equipped with various sensors, it is possible to utilize real time sensor data to inform the simulation. To incorporate the real sensor data into the simulation, we introduce the method of data assimilation. The goal of data assimilation is to provide inference about system state based on the incomplete, ambiguous and uncertain sensor data using a computer model. A typical data assimilation framework consists of a computer model, a series of sensors and a melding scheme. The purpose of this dissertation is to develop a

data assimilation framework for agent-based simulation of smart environment. With the developed data assimilation framework, we demonstrate an application of building occupancy estimation which focuses on position estimation using the framework. We build an agent based model to simulate the occupants' movement s in the building and use this model in the data assimilation framework. The melding scheme we use to incorporate sensor data into the built model is particle filter algorithm. It is a set of statistical method aiming at compute the posterior distribution of the underlying system using a set of samples. It has the benefit that it does not have any assumption about the target distribution and does not require the target system to be written in analytic form .To overcome the high dimensional state space problem as the number of agents increases, we develop a new resampling method named as the component set resampling and evaluate its effectiveness in data assimilation. We also developed a graph-based model for simulating building occupancy. The developed model will be used for carrying out building occupancy estimation with extremely large number of agents in the future.

INDEX WORDS: Data assimilation, Agent-based simulation, Smart environment, Particle filters, Component set resampling, Occupancy Estimation

DATA ASSIMILATION FOR AGENT-BASED SIMULATION OF SMART ENVIRONMENT

by

MINGHAO WANG

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2014

DATA ASSIMILATION FOR AGENT-BASED SIMULATION OF SMART ENVIRONMENT

by

MINGHAO WANG

Committee Chair: Xiaolin Hu

Committee: Saeid Belkasim

Wenzhan Song

Yichuan Zhao

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

December 2014

DEDICATION

Here I want to thank my family, my father and my mother for their strong and continuing support and understanding as I pursue my degree.

ACKNOWLEDGEMENTS

First, I want to thank my advisor, Dr. Xiaolin Hu for his aspiring guidance and generous advice for my doctoral study in Georgia State University. His broad and profound knowledge and insight guide and help me in this dissertation work. I learned a lot from him about writing and methodologies to carry out research.

Second, I want to thank my committee members Dr. Saeid Belkasim, Dr Wenzhan Song and Dr. Yichuan Zhao for the kindly suggestions and advices to improve my work.

I also would like to thank my group mates in SIMS group, Fan Bai, Haidong Xue, Yuan Long, Sanish Rai, Nicolas Keller and Peisheng Wu for valuable suggestions about research ideas.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Overview

This dissertation work aims at creating a data assimilation framework for agent-based simulation of smart environment. Agent-based simulation is a valuable tool to study the movement of occupants in a building. It uses a bottom up approach to model each individual's behavior and interaction with the environment. It provides valuable information about movement patterns of people in building structures and thus helps the design of building structure and the development of egress strategies in emergency situations. Nevertheless, traditional agent-based simulations of smart environment are typically used as offline tools to help system design. They are not dynamically data driven in the sense that they do not utilize real time data of the environment. On the other hand, with advances of sensor and communication technologies, more and more building environments are equipped with sensors that provide real time occupant location information. In this dissertation, we refer to smart environment as indoor building environment equipped with sensors. With real time data provided by sensors, how to assimilate these data into a simulation model becomes a relevant research topic. This dissertation develops a framework to incorporate real sensor data into agent-based model to estimate the occupant's position. The goal of data assimilation is to provide accurate inferences of system's state, which is people's location information in our work, based on real time sensor data and the simulation model. The data assimilation results can then be used to provide initial conditions for simulations for more accurate simulation results.

The motivation of this dissertation work originates from the importance of occupancy estimation problem in many application of building environment. For example, according to research carried out in [1], occupancy has impact on the building's energy consumption.

Estimation of occupancy can help the center control system of building to schedule energy supply more effectively. For buildings' heat, ventilation and control (HVAC) system, it is easier to manage the delivery of conditional air if demand based on occupancy information is available [2]. Building occupancy estimation can also help the design of applications to discover or monitor the occupants' activities in the building [3]. Because these activities normally involve changes in occupants' locations, they can be discovered and recognized from estimated occupants' location information. Furthermore, in an emergency such as fire or earthquake, the occupancy information of the building may help occupants choose route more efficiently to escape and help rescuing agents to deploy their rescuing resources [4]. To obtain building occupancy information, the most direct way is to utilize a set of sensors to measure the locations of occupants directly. As technology advances, more and more buildings are nowadays equipped with various sensors. In most cases, these sensors can provide measurement on occupancy of the building to some extent. However, method relies just on sensors measurement has its limitation. First, sensors have different resolutions. We refer resolution of occupancy sensor as the minimum change a sensor can detect on the occupancy of the building. In previous examples, the gas sensor is a low resolution sensor since occupancy information is derived from sensor reading indirectly, while the video camera is considered to be a high resolution sensor since it measures occupancy directly. If low resolution sensor is used, it is difficult to obtain occupancy information directly from these sensors. Even in the situation that high resolution such as video camera is utilized, the measurement may still suffer from the noise of the sensor itself or the clutter coming from the environment. Second, the coverage of the sensors' detecting area is usually limited; if the occupants are aggregated in some area that is outside the detection area of the sensor, there is no way for the sensor to measure the position of these occupants no matter

what resolution the sensor has. These are considered as the uncertainties of the sensors. Consequently, approaches based just on measurement from sensors are considered to be inappropriate in this research because of the uncertainty of the sensors.

Incorporating agent-based model in the smart environment application can overcome the deficiencies brought by the uncertainties of sensors. The uncertainty of the observations from sensor can be handled using the agent-based models because the agent-based model can represent the prior knowledge about occupant's movement patterns. With data assimilation approach, one can then meld observation from sensor and the developed agent-based model together using some melding scheme to estimate the occupancy. The benefit of data assimilation is that it uses a computer model to infer the state of the system and uses the sensor data to calibrate the simulation; therefore, it is possible to estimate the system state even when the observation is not perfect as long as the model used captures the dynamics of the target system. In this dissertation work, we developed an agent-based model for simulating the occupants' movements in the building. This model includes some basic behaviors of the occupants while moving such as avoiding each other and navigating. It is easy to be extended to add other behaviors. We then use this model in the data assimilation framework to work with the observation generated by the sensors to estimate the system state which is the occupants' locations. The method we used for data assimilation is particle filter, it is a set of statistical method that aims at using a set of samples, namely particles, to represent and estimate the posterior distribution of the system states. It has the benefit that it does not have any assumption (such as must be a Gaussian process or linear) about the target distribution to be estimated. Therefore, the target distribution can be of any form and can be multi-modal. The procedure of particle filter algorithm involves using system transition function to generate new samples at

each estimation step. It then incorporates the observation data to calculate the likelihood of each particle. In a standard bootstrap filter, the likelihood is considered as the importance weight of the particle at re-sampling step where the particles with lower weight have higher chance to be eliminated and particles with higher weight are more likely to be reproduced. Furthermore, to deal with two existing problem of particle filter algorithm itself, namely particle impoverishment and particle deprivation, a novel re-sampling scheme is proposed. It diversifies the sample space by dissembling particles into sub dimensions and re-assembling them again. By doing this, new combinations of values of different dimensions are generated so that the coverage of the samples on state space is enlarged. This method effectively increases particle filter's ability of using small number of particles to represent larger portion of state space and can improve the estimation performance when the target system has high dimensional state space. Furthermore, this dissertation considers a special case of building occupancy estimation that there are massive number of occupants in the building. The number of occupants in the building to be estimated can reach as high as several thousand. Under the assumption that the number of occupants in the building is large, whether it is possible to track each occupant precisely is a question. To estimate the occupancy under the assumption that massive number of occupants is involved, a reduced order model has been developed; instead of consider each agent's position as the state of the system, estimation using this model first divides the space of the building structure into zones and uses a graph to represent the floor structure of the building; it then uses the number of occupants in each zone as the state of the system. In short, the overall work of this dissertation includes establishment of agent-based model of smart environment, the utilization of particle filter algorithm to assimilate data into the simulation model, the development of new re-sampling

technology to deal with high dimensionality of the system and the development of the graph-based model.

1.2 Background

The concept of smart environment initially occurs during early 1980s. A smart environment is referred as any design that utilizes knowledge of the environment's context to aid accomplishing various tasks such as navigation, scheduling energy consumption, recognition of activities and evacuation planning. Smart environment applications usually involve a series of sensors and a central management system. The data are collected by the sensors and then fused in the central management system to aid the control of building's HVAC system or provide contextual knowledge about occupants. Generally, the development of smart environment applications can be categorized into energy efficient building [5-7], effective HVAC control [8-10], activity recognition [11-16] and emergency evacuation [17]. Applications for energy efficient building aim at utilizing occupancy information such as position of the occupant, presence and absence of occupants and occupant's activity to guide energy supply in the building. Applications for HVAC control system schedule the delivery of conditioning air, lighting or other resources based on occupancy information and try to maximize the comfortableness of occupants. Applications for activity discovery and recognition investigate information collected by various sensors and explore the patterns of occupant's behavior to discover and recognize the ongoing events in the building. In all of these applications, the core concept is occupancy estimation, which involves deriving the occupants' location information from sensors. Occupancy is important because occupants' location information (occupancy information) directly related to many aspects of contextual information of a building such as

energy consumption and occupants' activity. For example, consider a scenario that a building has four rooms A, B, C and D. Assuming there are 9 occupants in room A, 6 occupants in room B, 2 occupants in room C and 0 occupants in room D. Since room A has most occupants, it indicates that the demand for energy consumption in room A will be highest. The central control system of building can then schedule its energy supply more efficiently according to the potential energy utilization of each room based on the number of occupants in the rooms. Meanwhile, for HVAC control, since room D has 0 occupants, the conditioning air can be shut off for that room while room A will need more conditioning air to be delivered. Moreover, with the awareness of the feature of each rooms, for example, A is a conference room and D is the rest room, a building intelligence control system may automatically recognize the activity of occupants in the building. In this particular example, there is obviously a meeting going on in the conference room. This activity discovery procedure can help the intelligence building control system understand what is going on in the building and thus provide help to aid occupants in the building to accomplish various tasks. This example is shown in Figure 1.1.

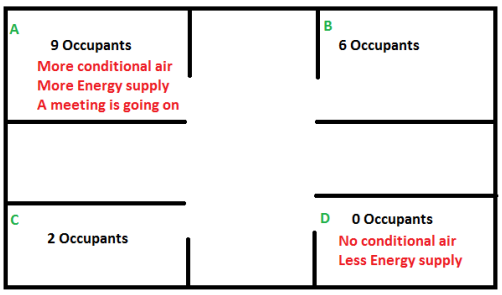


Figure 1.1 An example of how occupancy related to application

In general, occupancy estimation involves utilizing a series of sensors and a computer model to calculate occupants' positions in a building structure. The general framework is shown in figure 1.2.

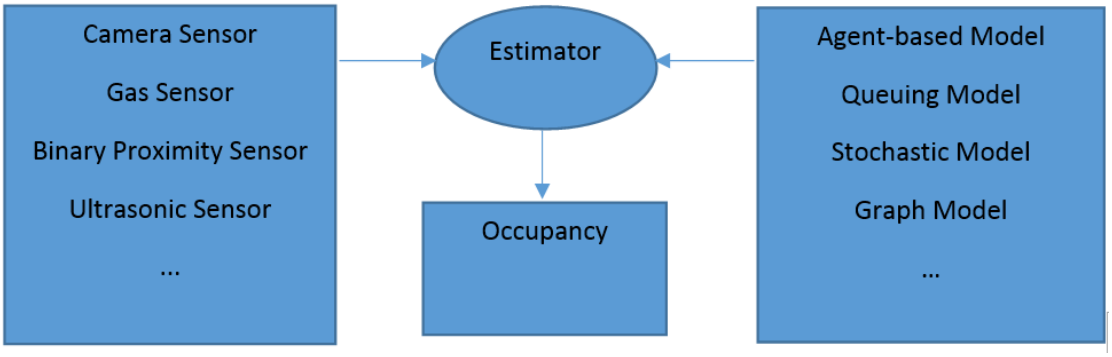


Figure 1.2 General framework of occupancy estimation

In this framework, the data is collected by various sensors with different resolutions. High resolution sensors such as camera sensor provides relatively accurate measurement of occupants' positions and low resolution sensor only provide inaccurate, ambiguous or indirect measurement of the occupants locations. One assumption related to the data collected by sensors is that the data are not perfect. It is assumed that sensor is subject to noise and environment clutter. Also, if sparsely deployed, the ability of sensors to detect occupancy change is limited by the coverage of the sensors. Due to these limitations of sensors, to estimate the occupancy, prior knowledge about occupant movement pattern or behavior is needed to incorporate with the data obtained by sensors. In this research, this prior knowledge is represented by computer models. The benefit of computer model is that it can simulate the occupants behavior in the case the observation data is not reliable. For example, if an occupant entering a zone where no sensor is deployed, its behavior can only be inferred from computer models of occupant's movement. There are various computer models in simulating occupants' motion behaviors in the building. A stochastic model [18] treats the absence and presence of occupants in some specific zone as a stochastic process and use some probability distribution to calculate the probability of occupants' absence and presence at specific time intervals. A queuing model [19] first models the structure of the building as a queuing network where each zone of the building is a service center and then

models the occupants as customers to be served at each of the service center. A graph model [17] models the structure of the building as a graph and then models the occupants' movements as the flows along the edges of the graph. An agent-based model [20] models each individual occupant as an autonomous agent and defines a set of rules for each agent to model their behaviors. Among these models, agent based model is probably the most popular modeling paradigm for simulating occupants' behaviors in building because it is the most direct and nature way to model each occupant's motion , the interactions among occupants and occupants' collective behaviors. In a typical example of agent based model of occupants described in [21], the author developed a model for simulating occupants' behaviors during evacuation in building structures. This model is built based on theories about individuals (following instinct, following experience and bounded rational), theories about interactions among individuals (social identities, personal space and social proof) and theories about the social group (crowd density, environmental constraints and perceived emotion and tension). It is featured with three different social behaviors which are competitive, queuing and herding behaviors. The competitive behavior defines that agents always try to approach the exit with maximized velocity and do not negotiate with others, it has a negative effect on the evacuation performance of agents. The queuing behaviors defines that the agents will try to follow a queue to approach the exit, this behavior yields best evacuation performance. The herding behavior defines that when there are multiple exits in the building, the agents tend to choose the exit that other agents have chosen to approach. This behavior sometimes improve the evacuation performance and sometimes decrease the evacuation performance. This model is a typical example of how agent based model can be utilized to investigate peoples' movement behaviors in buildings. Agent based models such as the evacuation model mentioned above provide relatively accurate description about the system

because they model the system in a bottom-up manner and therefore the abstraction level of the model is considered to be low. On the other hand, models such as a queuing model or graph based model treat the system as a whole and model the system in top-down manner. Consequently, we consider these models as high abstraction level model because they skipped many details about each individual's own behaviors and treat occupants as homogeneous entities. Applying models at different abstraction levels in the framework of occupancy estimation yields different benefits. For example, in the situation that massive number of occupants are involved, using a high abstraction level model is more appropriate because occupants' behavior can be treated as homogenous in this situation and using high abstraction level model improves computational efficiency. On the other hand, if less occupants get involved and we are particularly interested in the positions of each occupant, then agent based model is more appropriate according to the level of details required in the estimation procedure. In this dissertation, we tend to give a discussion on both high level abstraction level (Graph based model) and agent-based model on the performance of occupancy estimation in different situation.

The data collected by the sensors and the computer model are meld together into an estimator to produce the occupancy information (the location information of occupants). In this dissertation, we introduce the concept of data assimilation to meld observation data and computer model. Data assimilation involves incorporating observations into computer model to estimate the system state and use the estimated state as the initial condition of the simulation [22][23]. The details about data assimilation and its application will be described in following chapters.

1.3 Problem definition

As discussed in previous chapters, building occupancy estimation involves utilizing observations from sensors and the agent-based models to estimate the location information of occupants in a building. Formally, assuming there are *m* sensors $SE=<se_1, se_2, se_3, se_4, se_5, \ldots, se_m>$ deployed in the building environment, each sensor produces measurement of building occupancy within a range and is subject to the noise of the sensor itself and environment clutter. Given *n* occupants $O=<o_1, o_2, o_3, o_4, o_5, \ldots, o_n>$ in the building whose positions are updated based on their velocities and their velocities are updated according to the behavior of the occupants. An observation function $y = p * h(SE, O) + q$ defines how observation will be generated according to the positions of the occupants and positions of the sensors. In this equation, *p* represents the coefficient and *q* represents the noise of the sensor. The observation is assumed to be non-perfect, which means occupants' locations are only partially or indirectly measured by the sensors. We assume that prior knowledge of how people move in the building is available in some degree. The problem of building occupancy estimation is then defined as how to utilize the imperfect observation generated from sensors to get best estimate of the system state which is the positions of occupants in the building. The output of the solution to this problem depends on at what granularity level the occupancy is being estimated. If estimated at a low abstraction level, the output of the solution to the problem will be a length *n* vector of $L=<l_1, l_2, l_3, l_4, \ldots, l_n>$ where l_i is the location of occupant *i.* If estimated at a high abstraction level, the output of the solution to the problem will be a length *p* vector $Z=<z_1, z_2, z_3, z_4, \ldots, z_p>$ where z_i is the number of occupants in zone *i* and *p* is the number of zones in the building. As mentioned in previous sections, a general framework of occupancy estimation involves a set of sensors and a set of models. Thus, the problem of occupancy estimation can be further divided into two sub

problems. The first is modeling problem, which investigates how to build a model to utilize prior knowledge and to simulate the occupants' moving behaviors in the building. Modeling is important in a sense that sensors are imperfect and are not reliable. A solid model helps overcome certain challenges brought by the nature of the occupancy estimation problem, such as the uncertainty brought by the sensors. The procedure of building models such as those in [15][18] involves a process of learning from data. Such model is suitable for the specific environment because it utilized data collected from that environment. Other models such as those described in [4][17] are more generalized but less accurate comparing to the models learning from data collected for specific environment. In this dissertation, we intend to build generalized model without learning from data so that the method can be applied to other environment as well. The second sub problem is state estimation problem. Since the information provided by sensors is incomplete, the output of the sensors cannot be directly used as the result for occupancy estimation. The incompleteness of the sensor may come from the noise of the sensor itself, where the sensor reading does not reveal the true position of the occupant, the clutter of the environment, where the sensor reading is generated by other objects instead of occupants, and the lack of coverage of sensor, where the sensors are sparsely deployed so that there are areas considered to be undetectable by sensor and occupants' information becomes latent when they enter such areas. Therefore, to achieve the goal of obtain location information of occupants, a method need to be introduced to derive occupants' locations which could be a hidden state of the system due to the incompleteness of sensor data. In summary, the state estimation problem can be described as: given a model and incomplete observation data, how to obtain the hidden state of a system. One popular solution to this problem is the Kalman Filter illustrated in [24], Kalman filter solve the state estimation problem by recursively producing the estimate of

system's current state and uncertainty matrix and then compare the measurement to the estimate to update its uncertainty matrix. However, Kalman Filter and its extension require that the underlying model to be Gaussian and uni-modal. For a non-linear and non-Gaussian state estimation problem, particle filter[25] is a better solution. It represents the posterior distribution using a set of samples, namely particles. Due to this nature, it does not have strict limitations on the form of target distribution to be estimated. It estimates the system state by recursively updating the hypothesis represented by particles forwarding in time with state transition function and then weigh the particles according to importance function. In this dissertation, particle filter is particularly suitable since the movement behavior of occupants in a building is highly unpredictable and it is difficult to derive equations analytically to describe the patterns of the movements. The major problems of this research is modeling and state estimation, however, there are other problems needed to be solved. For example, while applying particle filter algorithm to estimation the state of occupants, the accuracy of the estimation is an important issue needed to deal with. Particle filter algorithm uses particles to represent the posterior distribution of the state of the system, the number of particles determines how close the estimated posterior distribution is to the real posterior distribution of the system. However, since the computational resources for estimation is usually limited, it is impractical to assume that a large number of particles are always available. As a result, problem arises how to accomplish the estimation utilizing limited number of particles. Furthermore, if the estimation is executed at low abstraction level, the state to be estimated is the positions of all of the occupants. The dimensions of the state space increases as the number of occupants to be estimated increases. Consequently, the number of particles required to cover the vicinity of the true state also increases. How to represent the posterior distribution using limited number of particles becomes an important issue

to address in this research. In the following chapter, this dissertation gives a comprehensive illustration and analysis about the challenges exist in occupancy estimation problems.

1.4 Challenge of occupancy estimation

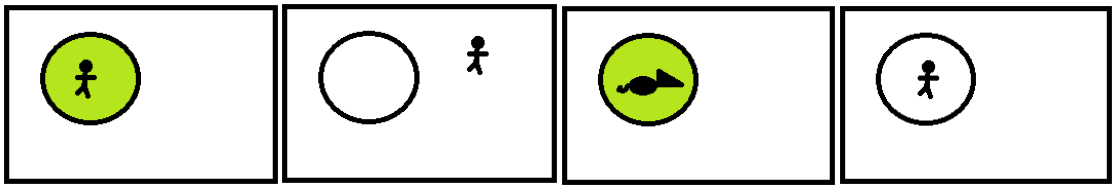
There exist several challenges in the occupancy estimation problem. In the framework discussed in section 1.2, there are three components in a typical solution to occupancy estimation problem: the sensors, the computer models and a method to meld sensors and models to produce outcome. The challenges in occupancy estimation comes from all of these three components.

First, as mentioned in section 1.3, sensors are imperfect. Therefore, the data collected by the sensor are not reliable and cannot be directly used as the output of the estimation. According to a research conducted in [26], the ignorance of the observation from sensors can be categorized into three classes: incompleteness, impression and uncertainty. The ignorance of incompleteness covers the cases that part of the information of the subject is missing. For example, for the case that we want to obtain the weather in San Francisco and Seattle, the statement "It is raining in San Francisco" reveals a precise and exact measurement of the weather in San Francisco but lacks the information about Seattle and is therefore incomplete. As a result, the measurement revealed by this statement is subject to the incompleteness. The ignorance of imprecision covers the cases that the information is given without desired precision. For example, for the case that we want to measure the length of a desk, the statement "This desk is about 80 to 120 centimeters long" contains complete information about the length of the desk but the value measured is a range and lacks required precision. As a result, this measurement is subject to imprecision. The ignorance of uncertainty covers the cases that the provided information is complete and precise, but its correctness cannot be easily verified. For example, for the case that we want to know the location of Sam, the statement "John said that Sam is in his house" is complete, precise but is

uncertain because whether John is lying cannot be verified. Consequently, the measurement obtained from John about Sam is subject to uncertainty. In this dissertation, the data collected from the sensors are also subject to all of the three categories of ignorance of the observations. In terms of incompleteness, there are two challenges.

First, according to different functionality of the sensors, the completeness of the information is different. In the best case where a camera sensor is utilized, the location information of occupants in its detection area can be measured precisely and completely. However, if low resolution sensor such as a gas sensor is utilized, the occupancy can only be inferred from the density of the gas, the information about presence or absence of the occupants is actually missing. Therefore, the observation is subject to incompleteness when using low resolution sensor. Second, sensors have its detection range, the occupancy can only be measured within the sensors' detection area. However, the occupancy sensors such as camera sensors or passive infrared sensors (PIR) are typically deployed sparsely in the building, leaving occupants in areas that are not covered by the sensors' detection range undetectable. When occupants enter the area where no occupancy sensors are deployed, their positions become latent. As a result, data collected by the occupancy sensors miss the information about these occupants and are subject to incompleteness. In terms of imprecision, the major challenge this research faces is that the output of the low resolution sensors is ambiguous. Unless a camera sensor or ultrasonic sensor is utilized, the sensors such as PIR sensors or gas sensors does not have the ability to distinguish the occupants or count the number of occupants. Furthermore, these sensors are subject to the environment clutter and may recognize other object (such as a mouse) as occupant. Therefore, the occupancy measured by these sensors is subject to imprecision. In terms of uncertainty, the major challenge comes from the noise of sensor itself. The measurement of the

sensors always contains noise, for example, the PIR sensor produces reading “1” if it detects an occupant in its sensor detection area and produces reading “0” otherwise; however, due to the noise of the sensor, it may produce reading “0” when detecting an occupant and produce “1” when there is no occupant in its detection area. When this happens, the sensors provides a precise and complete measurement, but the measurement is wrong. As a result, the sensors with noise are subject to uncertainty. These challenges must be addressed and solved in the underlying study. The ignorance of incompleteness, imprecision and uncertainty is represented in figure 1.3. In these figures, the circle represents the sensor’s detection area, the sensor is a PIR sensor and report “1” (represented by green color) if there is an occupant in its detection area.(a) shows a correct reading where the information is complete, precise and certain. (b) shows the case where the occupant is outside sensor’s detection area and the information collected by sensor is subject to incompleteness. (c) shows that the sensor reading is generated by environmental clutter (the reading is generated by the presence of a mouse) and therefore subject to imprecision. (d)shows that the occupant is inside the sensor’s detection area , the sensor is supposed to report “1” (turned to green) however due to the noise of the sensor, it produces a “0” reading. As a result, the data collected by the sensor in this case is subject to uncertainty.



(a) Correct Sensor reading (b) Incompleteness (c) Imprecision (d) Uncertainty

Figure 1.3 Illustration of ignorance of observations

Second, the model need to be developed to infer the occupancy when data collected from sensors are not reliable. There exist two major challenges while developing models to simulate

occupants' behavior. The first challenge is the choice of the models. The details (abstraction level) covered by different models are different. For example, at the low abstraction level, an agent-based model may model each individual occupant as an agent and defines the rules for occupant's behaviors and interactions; the model describes the occupancy dynamics in a bottom up manner and contains most details. Such a model is suitable when the number of occupants in building is moderate, individual agent's behaviors are important and there are interactions between agents. However, in the scenario that massive occupants are involved, whether agent-based model still works is questionable. A model at higher abstraction level can be more helpful in such scenario where massive occupants are involved. For instance, a graph model whose node represents each zone in the building and the edge represent the connection from one room to another can be used to model the occupancy dynamics. This model performs better than the agent-based model in a sense that the dynamics of occupancy is more similar to flow over the graph rather than aggregated behavior of individual occupants while a large number of occupants get involved. The choice of model should be based on purpose for which the model is designed. The second challenge for modeling is the design of the model. We need to consider what details should be put into the model. For example, in developing the agent-based model, a simple equation to update the agent's position with velocity and acceleration can be used to simulate the occupants' moving behavior. However, such an equation based function cannot model complex behaviors of agents such as navigation and avoiding each other. These complex behaviors must be modeled separately in order to capture the true patterns of the occupant movement.

The third challenge comes from the methodology we utilized to meld the observations and models together to estimate the occupancy. As mentioned in section 1.3, the state estimation problem can be solved using method such as particle filter and Kaman filter. At low abstraction

level where agent-based model is utilized, the system state is represented by the position of agents. At high abstraction level where a graph model is utilized, the system state is represented by number of occupants in each zone. In both cases, the system state is composed of multiple components (multiple agents and multiple zones). Therefore, the system state is high dimensional. Consequently, the high dimensionality of state space will have impacts on the estimation performance. For example, in particle filter algorithm, the number of particles required to converge to the vicinity of true state grows exponentially with respect to the number of dimensions of the state space. It is a great challenge to overcome this growth of complexity and represent the sample space with less and limited particles.

1.5 Methods to overcome challenge

The challenges described in previous section can be solved in different ways. The first challenge, the ignorance of sensor data, can be solved by melding the sensor data and computer models together with an estimation framework. Note that, a basic assumption here is that the ignorance of the sensor data is bounded. In terms of incompleteness, the assumption is that sometimes the information provided by the sensors are still complete and the incomplete information can be inferred from the complete information. For example, if an occupant is observed by a sensor at location l at.time k, if at time $k+1$ the occupant moved out of the sensors' detection area, since we know its position at time k , we can make a hypothesis that its position at $k+1$ is near l even the occupant is not detected by any sensor. In terms of imprecision, the unreliability of the error or ambiguity of the sensor is assumed to be able to be handled by computer models. For example, while counting number of occupants inside a sensor's detection area, based on a computer model and patterns of the sensor reading, we can make hypothesis about number of occupant in that area even the sensor does not have the ability to count the

number of occupants. In terms of uncertainty, the assumption is that the unreliability of the sensor can be represented by a model and historical data of sensor reading can be utilized to recognize whether the sensor report is uncertain. For example, in a scenario where a PIR sensor is utilized to measure occupancy, assuming that the sensor has 20% rate to report a false reading,(That is, 20% chance to report 0 when there is occupant in its detection area and report 1 other wise) when the sensor continuously report "1" in consecutive 10 time steps, it is more likely that the sensor is working without error because according to the error rate of the sensor, the probability for the sensor to continuously report false reading is low. On the other hand, if the sensor report three "0"s and then report one"1" and another six "0"s, it is most likely the reported "1" is an error and is incorrect information. In summary, the challenges brought by the ignorance of the sensor data can be solved by establishing appropriate models for both the occupant and sensor, then use these models to make hypothesis about occupants' state while estimating. In this dissertation, agent-based model and graph based model are utilized to serve this purpose. While creating the agent-based model and the graph model, the challenges of the choice of the model and specification of occupants' behaviors arise. To choose the appropriate model for the occupancy estimation framework, several factors need to be taken into account. The first factor is the sensor resolution. The sensor resolution refers to the minimum change a sensor can detect on the subject to be measured. When high resolution sensor is utilized, the data collected by sensors contains more information and thus allow a model at high abstraction level to work properly in the estimation procedure. When low resolution sensor such as PIR sensor or gas sensor is utilized, the data collected by these sensors contains less information. As a result, the model needed to be designed with more details in order to infer the movement behavior of occupants when the sensor data is ambiguous, noisy and incomplete. A model at low abstraction level such

as an agent-based model can suffice this requirement. Despite the sensor resolution, the second factor needed to be taken into account is the deployment of the sensors. If densely deployed, the data collected by the sensors are complete and precise since large portion of the state space (the space of the building floor) are covered by the sensors. Consequently, the volume of the information collected by the sensors is high and therefore a high abstraction level model can be utilized in estimation. On the other hand, if the sensors are sparsely deployed, the data derived from sensors contains limited information because there are areas that are not covered by the sensors' detection area. As a result, a low abstraction level model such as an agent-based model containing more details about the underlying system must be utilized in order to infer the occupants' positions when the occupants enters the area that are not covered by any sensor. The third factor needed to be taken into account is the purpose of the estimation. In other words, how the outcome of the estimation will be utilized. If the purpose of the estimation is to obtain the exact position of each of the occupants, then the system state is the position of occupant and thus an agent-based model which treats positions of the occupants as its system state will be more appropriate. On the other hand, if the expected outcome of the framework is the number of occupants in each zones in the building (in other words, we do not care about each occupant's exact locations), then a graph model is a better choice because in graph model we count number of occupants in each vertex of the graph. In summary, to choose the appropriate model, the resolution of sensors, the deployment scheme of the sensor and the purpose of the simulation are all important factors to consider. After the choice of models is made according to the consideration of the three factors, the next step is to develop a method to combine the observation data and computer models in real time to estimate the location information of occupants. The challenge existing in this step is the high dimensionality of the system state space

as mention in previous sections. Specifically, in this dissertation, we meld the observation data and the model using particle filter algorithm, it uses a set of samples (particles) to approximate the posterior distribution of the underlying system. The particles are associated with importance weights to represent its probability to be selected to reproduce at each estimation iteration. A selection step is applied to eliminate particles with lower weights and multiply those with higher weights. When the number of dimensions of the state space increases, the number of possible combinations of values of the state dimensions increases too. As a result, the number of particles required for the algorithm converging to the posterior distribution increases exponentially with respect to the number of dimensions of the state space. To overcome this challenge, we proposed a novel method to increase the diversity by breaking the combinations of values of dimensions of state space. This is done by dividing the state of the particles into sub dimensions, namely components, and reconstruct new particles from the set of components. We name this method as component set re-sampling, it efficiently increases the diversity of the samples while they are being re-sampled and the ability of particle filter algorithm to represent the posterior distribution using limited number of particles. The details of this method is discussed in section 5.3.

1.6 Agent-based model

In this dissertation, agent-based model serves as a low abstraction level model in the estimation framework. According to [27], agent-based model is referred as "a computational method that enables a researcher to create, analyze, and experiment with models composed of agents that interact within an environment."[27]. The agent-based model has several features which make it a useful and prominent tool in studying people's movement behaviors in a building structure. First, the agent-based model has the feature of "ontological correspondence"[27]. An agent in the model can be directly corresponded to an entity of the real

world target system. In the case of the model of people's movement in the building, each agent can represent an individual occupant moving inside the building. This makes it easier to represent the whole dynamics of the occupants' movement from each individual's behavior. Second, the agent-based model can have "heterogeneous agent"[27]. This means each agent can have its own preference of behavior or even have its own rule to perform actions. In the case of the model of people's movement in the building, each agent update its position according to its own destination and velocity. The aggregation of individuals' choices of behaviors form the overall dynamics of occupants. Third, the agent-based model can represent the environment [27] where the agents perform their actions. In the case of the model of people's movement in the building, the environment is modeled with free spaces, obstacles and sensors and the agent are designed to interact with these objects. A way point graph is also modeled according to the structure of the building to assist the agent for navigation. Forth, the agent-based model can model the interactions among agents [27]. Typically, the interactions among agents can be modeled as a set of rules to define what messages or actions are passed from one agent to another. This is particularly important in simulating occupants' behaviors because the dynamic of occupants' movements includes collective behaviors such as avoiding each other. Such behavior must be captured by the underlying simulation model in a convenient way. In the agent-based model proposed in this dissertation, the interactions among agents, i.e. the avoidance behavior, is implemented using a set of rules. By following these rules, the agents avoid each other to avoid collision and resolve congestions.

Agent-based model is widely applied in studying people's movements in different environment. Pedestrians on a street or occupants in a building can be easily modeled as an agent. By defining rules of behaviors of agents, the pattern of the movement of people can be

modeled. In [28], a typical agent-based model of pedestrian behavior is presented. This model is featured with two categories of agents' characteristics, the social-economic characteristics and behavioral characteristics. The social-economic characteristics of the agent include the individual properties such as sex, income and are used to create a planned route passing through a sequence of destinations on the map. The route is created using shortest path algorithm on a series of waypoints. It simulates each individual's pre-defined plan based on their social-economic characteristics. The behavioral characteristics include agent's attribute of activities such as speed, visual range and fixation. The speed defines how fast the agent can move in the environment. The visual range defines the area an agent can observe using virtual sensor. And the fixation defines how stick the agent will on its pre-defined route. The agents' moving behavior of this model is derived from the two categories of characteristics of the agents. In another work described in [29], author developed an exosomatic visual architecture system for guiding the natural movement of agents. It uses a grid graph to compute the visual range of an agent. Based on the visual field of the agent, the agent uses the decision making process to determine the movement of the agents. The two works presented above demonstrate two fundamental essences for modeling pedestrians' movement behavior, that is, the behaviors originated from social-economic aspect such as navigation and behaviors originated from the nature movement of people such as the rule-based decision process to avoid obstacles. The agent-based modeling work in this dissertation borrows the ideas from these works. It models the navigation behavior and avoidance separately in order to capture the basic dynamics of occupants' movement. The details of the agent based model is described in chapter 3.

Except being used as tool in studying people's movements, agent-based model also finds its application in social science. In social science, one of the major challenges that scientist may

face is that the experiment is difficult to conduct. Conducting an experiments on the research subject requires certain treatment to particular social group. The outcome of social group being treated is then compared with the social group that is not being treated to study the difference. Unfortunately, in many cases, such treatments face ethnical issue and cannot be easily carried out. Agent-based model provide a viable solution to this problem. Because the social entities are modeled as agents and executed within the context of a computer program, ethnical issues are naturally avoided. However, agent-based modeling and simulation in social science is different from its application in natural science in a sense that it does not aim at predicting the future. Rather, the purpose of application of agent-based modeling and simulation in social science is to reproduce, understand social phenomena and to develop social theories. It is normally used as an offline tool without incorporate real time data. This is the fundamental difference between applications of agent-based modeling in social science and applications of agent-based modeling in other disciplines. The application of agent-based model in social science is out of the discussion of this dissertation. However, there exist similarity between the procedures of building the agent-based model in social science and the agent-based model of people's movement behaviors in building.

1.7 Graph Model

Graph model serves as a high abstraction level model in the building occupancy estimation framework proposed in this dissertation. The basic idea is to abstract the building structure to a graph. The vertices of the graph represent the zones of the building and the edges of the graph represent the connectivity between zones. The occupancy is represented by the number of occupants in each zones. And the movements of occupants are modeled as flows of the occupants from one zone to another. The motivation of developing graph model to model

occupancy dynamics is that when the number of occupants to be estimated is large, using agent-based model will be computationally expensive. The computation complexity of simulations grows proportionally to the number of agents in an agent based model and grows proportionally to the number of zones in a graph model. Therefore, in the case that the number of zones in a building is significantly smaller than the number of occupants, the graph model yields higher efficiency than the agent-based model. However, the graph model has some drawbacks. First, the scenario of congestion cannot be easily modeled because the occupants in graph model are treated as homogeneous. The solution to this problem is to employ the queuing theory and implement each vertex of the graph model as a queue. The work presented in [30] gives an example on this approach. In [30], the author proposes a method that models the floor plan of the building as a graph with the vertex of the graph as queue. This allows occupants to be queued when they arrived at the vertex. The first occupants in the queue will wait for a period of time and then exits to the hallway. The occupant is then removed from the queue and the next occupant enters the queue and start to wait. The author utilizes a series of simulation tools to explore the parameters of queues such as evacuation time or congestion delays. This work gives some basic analysis on how graph model can be useful in modeling occupant behaviors. Another similar research is proposed in [19]. In [19], to investigate the emergency evacuation problem, the author uses a queuing network to represent the building floor and flow of pedestrian. The nodes of the queuing network are categorized into different classes. The first tier classes of the nodes of a network are dwelling nodes and circulation nodes. The dwelling nodes can be further categorized into origin nodes and destination nodes while the circulation nodes can be further categorized into horizontal circulation nodes, vertical circulation nodes and transitional nodes. The author then shows how a building floor structure can be evolved to a close queuing network

and presents the mathematical equations that capture the pedestrian flows in the network. The queuing network is implemented using mean value analysis algorithm which is a technique to compute the parameters of the queuing network. This work illustrates how queuing network theory can be utilized to model occupancy dynamic including congestion delays in a building.

The second drawback of graph model in occupancy estimation is that the interaction between occupants is difficult to represent. Since the state of the system is represented using the number of occupants in each zone, the concept of individual has been eliminated from the model. As a result, there is no way to represent the interaction among individual occupants. Fortunately, under certain circumstances, the interactions among occupants is not significantly a factor affecting the overall movement pattern of occupants. Especially in the scenarios that massive occupants have involved, the movement pattern of occupants is more like flow of particles over the zones of the building. The effect of interactions among occupants can be ignored.

In this dissertation, the graph model we utilized in the estimation framework has the simplest form. This dissertation does not intend to model every aspect of the occupancy dynamics such as congestion. Rather, this dissertation intends to build a high abstraction level model that captures the basic rules of occupants' movements so that the computational cost of the model can be minimized. The details of the graph model is discussed in section 6.

1.8 Particle filter algorithm

The algorithm used for data assimilation in this research is Particle Filter (PF). PFs are a set of sample-based 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 [25]. It has the advantage that it does not assume Gaussian property of a system and it allows the targeting posterior distribution to be multi-modal. In a standard bootstrap filter

algorithm [25], the proposal distribution where samples are drawn from is chosen to be the transition prior, which is the agent-based simulation model and graph model in our work. While the bootstrap filter algorithm can support data assimilation for agent-based models, it also faces a major challenge due to the high dimensional system state associated with multiple agents. In PF, the probability density represented by particles converges to posterior distribution when the number of particles is large enough. The number of particles required for convergence increases exponentially with respect to the number of dimensions of the state space and the degree of freedom of each dimension. In agent-based simulation of smart environment, each agent has its own state and the overall system state is the combined state from all agents. As the number of agents increases, the dimension of the system state increases too. To support effective data assimilation in agent-based simulation, there is a need to develop advanced method for dealing with the high dimensional state space associated with multiple agents. Otherwise, the estimation procedure will suffer from deprivation brought by combinatorial explosion, a typical problem for PFs. Several methods have been proposed in the literature to deal with the deprivation problem, including diversifying the particles [31], adjusting number of particles [32], and dealing with high dimensionality of state space analytically [33]. In this work, observing that the high dimensional system state is composed of states of individual agents (each agent's state variables can be thought of as one component of the overall system state), we propose a new method named component set re-sampling in PF. In the proposed component set re-sampling, instead of re-sampling each particle as a whole as in standard particle filters, we divide the state space into sub dimensions (named as components) and resample from the component set to "construct" new particles. By doing this, the combinations among tracked agents in each particle are broken when being re-sampled and therefore the same number of particles can represent more possible

combinations of state variables. We show how the component set re-sampling can work together with the standard re-sampling to support data assimilation for agent-based simulation of smart environment. The details are given in chapter 4, chapter 5 and chapter 6.

1.9 Organization of the dissertation

The rest of this dissertation work is organized as follows: Chapter 2 gives an overview literature review about related research area. The related work of building occupancy estimation, various models for simulating occupant dynamics, the application of data assimilation, particle filter and target tracking are presented. Chapter 3 describes the smart environment and agent-based model of the smart environment. The concept and configuration of smart environment is introduced. A framework of agent-based model to simulate occupants' movement behavior in smart environment is presented. Two important behaviors, the navigation behavior and the avoidance behavior, are defined. The functionality of the sensor and the deployment of the sensor is also discussed. Chapter 4 introduces data assimilation on the designed agent-based model. An introduction on the concept and method of data assimilation is presented. How particle filter is applied with the agent-based model for state estimation including the state transition function, measurement model and re-sampling algorithm is demonstrated. This chapter also presents some experiments to investigate the impact of number of particles on the estimation performance. Chapter 5 explores the improvement of particle filter algorithm on high dimensional state space. In this chapter, two problems of particle filter, the sample impoverishment and particle deprivation, are demonstrated. Due to these two problems, the number of particles required for convergence increases exponentially with respect to the number of dimensions of the system. To solve this problem, a novel re-sampling algorithm with the name component set re-sampling is proposed in this chapter. Experiments are carried out to illustrate

the impact of state dimensions on the estimation performance. Also the comparison of performance between components set re-sampling and standard re-sampling is given in another experiment to demonstrate the advantage of using component set re-sampling. Chapter 6 gives an introduction on the graph model. In this chapter, how a building can be modeled as a graph is presented. Simulation of occupants based on the graph model is carried out. Several experiments are carried out to validate the method. Chapter 7 gives the conclusion for the dissertation work and discussed possible future extensions on this research.

2 RELATED WORK

2.1 Occupancy estimation

Building occupancy estimation involves utilizing data collected from a set of sensors to compute the location information of occupants in a building. The application of building occupancy estimation varies based on the how the result of the estimation will be utilized. There are majorly three categories of applications that the results of building occupancy estimation are used for: energy efficient building HVAC control, evacuation planning and activity discovering. In this section, we give a literature review on each of these three categories of applications.

2.1.1 Occupancy estimation for energy efficient applications

Building occupancy information is important in designing energy efficient applications. In[5], author investigates the energy consumed by lighting device in 158 rooms for two weeks and conclude that people are not diligent to control the light of some specific type of rooms. As a result, using occupant sensor to control wasted energy consumed by light can save up to 60% energy. In[6], the author carried out research on energy consumption of two buildings in the MIT campus to investigate the relationship between occupancy level and energy consumption. The

study is conducted in a two week time frame and the occupancy is measured using the number of WIFI connections. The result shows that there is strong correlation between occupancy and the overall electricity consumed in the building. In[7], the authors address the problem of using high resolution sensor such as camera sensor and ultrasonic sensor to measure the occupancy of a building. These sensors provide relatively high volume of information but may violate the privacy of occupants and is also difficult to deploy. Due to this reason, the authors propose to utilize door sensor and PIR sensor to measure the occupancy. Comparing to high resolution sensors, these sensors are low cost and will not violate occupants' privacies. When using door sensor and PIR sensor to measure occupancy, a room is marked as occupied only when the door sensor is marked as "open" or the door is marked as "closed" but the PIR sensor reports "1". The estimated occupancy information are used as input for a building energy simulation program and up to 15% energy can be saved using this system. This research does not intend to estimate the exact locations of occupants, rather it only measures whether there exist occupant in specific locations. In [8], the author builds a model for simulating the energy consumption by occupant controlled heat, ventilation and conditioning air (HVAC) system and run simulation. The behavior of an occupant is represented using a random process. The result of the simulation shows that occupant controlled HVAC system offers significant energy saving and utilization of occupancy sensor can play important roles in offering even more energy savings. The details of the model of the thermal environment and the model of the occupant behavior in [8] are described in [10]. In[10], the environment is modeled as an aggregation of cell grid with uniform distributed temperatures. The occupant behavior is modeled as a tree structure representing the probability of presence and absence and the probability for occupants to turn off the HVAC equipment when they are leaving the office. This paper also mentioned that using occupancy

sensor can dramatically reduce energy consumption. In [9], author addressed the drawbacks of using PIR sensors and CO2 sensors that the PIR sensor can only detect whether there is occupant in the room but not the number of occupants in the room and CO2 sensor is too slow for regulating CO2 in time. In [9], the author uses a sensor network of cameras to measure the occupancy. The cameras are deployed at the transition boundaries of the building structure. The transition boundary is defined as doors between rooms and rooms or rooms and corridors. The data collected from the camera data are used to train a Markov Chain model representing the occupancy transition of the building. Each state of the Markov chain represents a distribution of occupants in each area and new state is reached by transit occupants from one area to another. To prevent the discontinuity of the state, the author proposed closest distance Markov chain method. This method finds the state that requires minimum transitions to reach from an original state and use this state as the occupancy state if there is no transition probability in transition matrix for the original state. Another method proposed by author to overcome the challenge of discontinuity of the state is blended Markov chain, it blends the transition matrix to single state space so that each state of each transition matrix can be included in the newly constructed transition matrix. This work also integrates the model proposed into sensor network to make strategies of energy savings and controls. In [34], the author proposed to reduce the energy consumption in the building by improving the functionality of sensors. Specifically, the sensor are designed to having the ability of learning from the activity of occupants it detects and adjust its time delay variable. The time delay (TD) variable defines how long the light will be turned off after last motion of occupants are detected. The author uses two method for the smart sensors to adjust its TD variable. In method one, the author utilize a series of rules to determine whether TD will be extended or reduced based on the behaviors of the occupants. In method two, the author models

the inactivity period of the sensor's detecting area as a statistic model and use that model to determine TD. The author claims the sensors developed using proposed methods can save up to 5% of the energy consumptions. In [35], the author uses PIR motion sensors door sensors combined with a hidden Markov model to determine the state of occupant in a room and when to turn off or turn on the HVAC system. Three different hidden states are defined for occupancy: away, active and sleep. The state "away" defines that the home is not occupied at all. The state "active" defines that the home is occupied and at least one occupant and the occupants are not sleeping. The state "sleep" defines that all the occupants are sleeping and are inactive. The observation for the hidden states include features such as the time, the sensor firings of PIR sensors and firing of door sensors. These observation features are used to train the transition probability of the hidden Markov model. The trained model is used to determine when the HVAC or heating system of the home should be turned off. Furthermore, the author utilizes historical sensor data and the pattern of arrival of residents to determine when to turn on the HVAC and heating system to preheat the home so that thermo comfortableness of the residents can be increased. In [36], the author builds system that can generate user profiles of energy consumption based on sensors deployed in a wireless sensor network. The system is featured with two modes: off-line mode and real-time mode. In off-line model, the data are collected to establish profiles of energy consumption by users. This profile includes presence profile, temperature profile and light profile. In real-time model, the system collects value of physical parameters of the environment so that the system can perform automatic adjustments on the HVAC system. Based on the user's profile, the pattern of energy consumption of the user can be predicted and effective energy management strategy can be automatically carried out to reduce energy consumption. In[37], user proposed to use RBF artificial neuron network to establish the

model used in estimation and use data collected from various sensors to train the neuron network model. The data collected comes from various sensors including a motion sensor, a light sensor, a CO2 sensor, a sound sensor, a PIR sensor, a humidity sensor and a temperature sensor. The accuracy for the trained neuron network for estimating occupancy can reach as high as 80% in self estimation case (estimating the occupancy in a room using model trained from data collected from the same room) and can reach approximately 60% in cross estimation case (estimating the occupancy in a room using model trained from data collected from the a different room). The result of the estimation can be used to support demand-driven HVAC operations. In [38], the authors develop an application framework to control the HVAC energy consumption based on the measured occupancy of the room. It is featured with three subsystems: the monitor system, which contains a set of sensor node to measure the internal and external conditions of the building; these conditions include the occupancy, the temperature, the humidity of the building and the light condition of the exterior of the building; the actuation system, which determines how the conditional air will be supplied in the building; and the main controller, which is responsible for getting input from the sensors and arrange the supplement of conditional air, this is done using a control algorithm that takes current occupancy of a zone, occupancy of the neighbor zone and occupancy pattern as input and produce the desired air flow as the output. The author claims that the proposed system yields 50% improvement on saving the energy consumptions. In [39], an environment deployed with various sensors such CO2 sensor, humidity sensor, temperature sensor, acoustic sensor and occupancy sensor are utilized to obtain the occupancy information of the building environment. A set of methods such as support vector machine, artificial neuron network and hidden Markov model are utilized to estimate the occupancy. One of the purpose of this research is to investigate what features detected have the

highest correlation with the occupancy level of the building. Result shows that CO2 and acoustic have the highest correlation with the occupancy level.

Note that most literature in this section provides a connection between possible energy saving applications and the occupancy of the building. However, most of these approaches are purely sensor based or aiming at using model built from machine learning technology to estimate the occupancy, which is not reliable when the sensor data suffers from imprecision, incompleteness and uncertainty or there is no accurate model to describe the occupants' behavior. The method proposed in this dissertation intend to solve this problem by utilize both sensor data and prior knowledge of a model and meld them using particle filter algorithm for estimation.

2.1.2 Occupancy estimation for evacuation planning

Occupancy estimation is useful in the case of evacuation of the occupants in the building. When an emergency such as fire or disaster occurs in the building, occupants in the building stop their activity and try to move to the exit to escape from the situation. During the evacuation of occupants, the location information or the position distribution of occupants becomes important to retrieve. This information can be utilized by first responder such as firefighters to realize the situation in the building and deploy rescuing resources more efficiently. In [17], the author proposed an approach with multiple models to estimate building occupancy during the evacuation of occupants from the building. This approach is based on three different models: an agent based model, which includes the detailed description about each individual occupant's velocity, behavior and trajectory; a coarse model, which represents the building structure and traffic dynamics using a graph; a kinetic model which models the congested areas of the building as queues to simulate the situation of congestion in the building. With the model developed, the

author uses Extended Kalman Filter to estimate the zone level occupancy predicted using the coarse model and kinetic model. One important conclusion draw from the experiments by the author is that the estimation result based on sensor combined with model is significantly better than the estimation result based purely on sensor measurement. Another conclusion is that agent-based model is not suitable for real time estimation due to the computational cost while the coarse and kinetic model are more efficient and can be used in real-time estimation. The framework of the method proposed in this work is similar to the framework of the method proposed in this dissertation that is, given a set of sensors where various observation data can be derived from and a set of models representing the dynamics of the occupants in the building, an estimator is utilized to meld the model and observation to estimate the occupancy state. In [40] the author proposed a simulation model for simulating emergency evacuation. The model is featured with a GIS component and a C++ simulator. It uses the concept of micro-simulation to simulate the detailed movements and positions of the moving vehicles. The positions of the vehicle is modeled as locations which are road segments the vehicles can move into. A set of locations becomes arc and form the road network. The vehicles can move only when its next location has free capacity available. This paper gives an example on how micro-simulation can be used in helping emergency evacuation planning. In [41], the authors propose to utilize a wireless sensor network to detect hazard in the building and navigate occupants to avoid the hazard. This approach is based on a navigation graph and a hazard spread graph of the building. The navigation graph represents the time for the occupants to arrive at each node of the sensor network and the hazard spread graph represents the time for the hazard spread to each of the nodes so that the region covered by that node becomes hazardous. The safety of each scheduled path is calculated based on the difference between the safety of previous node and the navigation

time of the current node. In [42], a model called SIMULEX is proposed to simulate occupants' evacuation behaviors in large buildings. It first uses the program of DRAWPLAN to input the floor plan of the building including rooms, furniture, walls and exits. It then uses program GRIDFORM to create a distance map of the building. In the distance map, the floor plan of the building is divided into blocks and each block is assigned with a value to indicate the shortest distance from this block to the nearest exit of this block. SIMULEX uses three circles to represent the shape of the human body, this makes it easier to calculate the distance between two persons that are in close proximity and to calculate the overtaking of one person to another. In[43], the author build an agent based model to simulate occupants evacuation in large public building. This model is featured with two different sub models- the spatial environment model and the agent decision model. The spatial environment model handles the structural information of building environment, the layout of the building, the source of the fire and the dynamics of toxic gas generated by the fire. The agent decision model handles the agents' behaviors according to the environment and other agents. This includes the route planning, decision of the speed and how to resolve confliction between agents when two agents intend to enter the same area simultaneously. In[44], author build an evacuation model based on a graph with each vertex representing a room, a segment of corridor or hall ways and each edge representing a pipeline that occupant can transport on. The transportation of occupants on the edge of the graph is determined by factors such as social affiliation, access visibility, tenable time, speed, flow rate and distance between rooms.

Most of the methods listed in this section are specifically designed for the investigation of occupancy dynamics under the scenario of emergency such as a fire or other hazard. Some of the methods utilize sensor to detect the hazard and then make plan for the evacuee, some of the

methods establish dynamic model for investigating the evacuation of occupants, some of the methods utilize both models and data derived from sensor to estimate the occupancy then provide the result of the estimation to first responders to make evacuation plan. Intuitively, the method that utilizes both sensor data and simulation model such as the one proposed in [17] yields best result for estimating the state of occupants in a building.

2.1.3 Occupancy estimation for activity discovering

One important application of building occupancy estimation is activity recognition. By exploring the patterns of occupancy dynamic, researchers are able to investigate what is happening or what occupant is doing in the building. In [11], author develops a framework to identify occupants' activity using evidence theory. The author utilizes a directed acyclic graph (DAG) to represent the model and the hierarchy of the knowledge of the sensor and the activity. The DAG is featured with three layer of inference. At the bottom most layer is the sensors including door sensor, PIR sensor and so on. The sensor readings produce context values which is the middle layer of the graph. The context value depends on the type of the sensors. For a PIR sensor the context value would be 0 or 1 to indicate whether there is an occupant in the sensors' detection area. For a keyboard sensor, the context value will be active or inactive to indicate whether the keyboard is being used. From the context value, the topmost layer of DAG can be reached. That is the situation layer. The situation layer represents the actual activity of the occupant. For example, from the context value keyboard is inactive, the situation that the occupant is not at the desk can be inferred. In another example, when the context value keyboard is inactive and the occupant is detected at the coffee maker, the situation that occupant is taking a break can be inferred. The context value beliefs in the DAG are treated as evidences. The algorithm first compute the evidences from the sensor reading, then using a mass function to

calculate the context value beliefs. The evidences are combined using Murphy's combination rule and Dempster's rule to infer the belief of the activity of the occupants in the building. This method improves the recognition accuracy by 70% according to author's experiments. In [12], the authors apply a different methodology on identifying occupant's activities. In their work, the authors divide the information gathered by various sensors into two different groups, that is, the information gathered about the object such as furniture or human posture and the information gathered about the locations. For example, the activity of sweeping the floor can be inferred when a broom is utilized and the activity of cooking can only happen when occupant is detected in the kitchen. The activities are modeled by a set of description logic rules including the object used in the activity, the location of the activity and the human postures. When the sensors fulfilling the rules fired, the activity can be recognized. In [13], the author utilizes data mining technology to discover the activities of occupants in smart homes. It uses historical data to generate time intervals for each activity so that activities will be not be considered at the time that is out of the time interval associated to them. This method improve the performance of the algorithm by 70%. In[14], the author uses Latent Dirichlet Allocation model to discover the occupancy patterns in a building. The sensor used in this research is binary sensor. The author divides the day into 9 time intervals and uses a verbal system to represent the data obtained from sensors. The verbal system is then used in building a Latent Dirichlet Allocation model to detect long term pattern of the occupancy dynamic. In [15], the author investigates the problem of learning from limited historical data obtained by sensor that is not fully reliable to recognize the activity of occupants in a smart home. The author categorizes the attributes representing activities of occupants into four classes, which are person and activities of daily living, time and episode. Person represents the profile of individuals performing activities in the home; activities

of daily living represents activities such as sleeping, cooking and walking performed by occupants; time represents the time the activities take place; and episode represents sequence of sensor behavior. The data model is used as basis in the maximum likelihood estimation to find the model that can best fit the data in the learning procedure. In[16], the author gives an overview about the sensor technologies utilized to build smart environment. The author especially mentions the usage of camera sensors and microphone sensors as static sensors to achieve ubiquitous sensing. The author also addresses the needs to develop calibration, networking and distributed computing technologies to aid the design of the ubiquitous sensing system for smart environment. Moreover, the author lists some path way that can lead to ubiquitous sensing such as identifying occupants, tracking their location, discovering activities, tracking their poses, exploring their gestures or facial expressions and recognize speech in the environment. In [45], author develops a method to discover activities of occupants in a smart home application. In this work, the author addresses problem of previous researchers on discovering patterns of sequence of activity that the pattern they discovered cannot be discontinuous. This means in an example of activity of preparing coffee, the sequence of activities of adding sugar and then adding cream and the sequence of activities of adding cream and then adding sugar will be recognized as two different sequence of activities. This is however not the case. To solve this problem, the author proposed a method that can recognize discontinuous sequence of activities and uses clustering algorithms to group the sequences of activity discovered together as a set to represent that activity. The clustered set are later used in training a hidden Markov model to recognize the labeled activities. In [46], the author first addresses the challenges posed by activities of humans, that is, “concurrent”, meaning that the humans can perform several activities concurrently; “interleave”, meaning that some activities

can affect each other; “ambiguous”, meaning that the interpretation of the activities can be different, some certain phenomena can be interpreted as caused by multiple different activities; and “multiple occupants”, meaning that the activities are performed by different individual in the environment and therefore generates difficulties to recognize who performs the observed activities. The author then introduces the general approach of using hidden Markov model to recognize activities and addresses its limitation on detecting concurrent and interleave activities. This issue can be solved using conditional random field model and emerging pattern approach. Furthermore, this paper discussed several approaches to discover activities such as those which are similar in [14] that uses a verbal system to model the activities where the patterns of activities are treated as similar as topics in documents. The result of activity discovering can be used as basis for establishing models in activity recognitions. In[47], author proposed a method that uses naive Bayesian classifier to discovery and recognize occupant’s activity in a smart home environment. Two types of classifier are implemented, one is that in which the activities are mutually exclusive and another is that in which the activities can exist concurrently. This system is featured with three components, the sensor to gather information about the usage of objects, the experience sampling tool handled by subject to label their activities and the algorithm to recognize and classify the activities. Data are collected from the sensors attached to various objects in the home and are labeled by the occupants using a hand hold device. Temporal relationships are incorporated into the naive Bayesian classifier for better recognition performance. An important advantage of this system is that it is investigated in the home of normal people instead of researchers, therefore, the result is more sound and proving comparing to many existing work.

Note that, in most of the approaches listed in this section related to activity discovering with occupancy information, it involves a procedure of collecting data from sensors, labeling the data with activities and utilizing machine learning technology or classification technology to learn and recognize the patterns of activities from the labeled data. Although these works are related to our work in a sense that both the problems of activity discovering and occupancy estimation involve utilizing sensor data to obtain the contextual knowledge about the occupants in a building, this dissertation does not give an insight on problems of activity discovering. Readers are encouraged to read literature such as [48] to explore more in the domain of activity discovering and recognition. However, the result of occupancy estimation can be useful in activity recognition since the location information as the output of occupancy estimation is related to the activities of occupants in buildings.

2.2 Models for simulating behaviors of occupants

An important component in the occupancy estimation framework proposed in this dissertation is the model which serves as prior knowledge of the patterns of occupants' movement so that location of occupants can be inferred when data from sensors are not reliable. There are majorly two categories of models that can serve this purpose. An agent based model models each individual occupant as an autonomous agent with its own rules of performing behaviors and interacting with others. A coarse model models the occupants as entities flowing in a network of nodes, where each node represents a room or a corridor and each edge represents the connectivity among the rooms and corridors. In this section, we give a literature review on the work carried out by previous researchers on both of the two types of models.

2.2.1 Agent-based models

Agent-based model is a useful and prominent tool to study people's movement behavior in various environments. Numerous research on building such model have been conducted by previous researchers. In [49], the author builds an agent-based model that simulates the movement behaviors of pedestrian in an urban environment with facilities such as church, park, monument, post office and library. The agent-based model is featured with two different behaviors. One is random walk, meaning that the agent moves on the street casually without a clear purpose; the other is purposely movement, meaning that the agent has a predefined destination to move towards to. Experiments show that in the random walk case there is strong correlation between local integration and movements of pedestrian where this correlation is weak in the purposely walk case. In [50], the author builds an agent-based model based on psychological factors. Specifically, the psychological factors are classified into two classes, psychophysical factor and psychosocial factor. The psychophysical factors include the range of the agent's vision and the variation of their moving speed. The psychosocial factor include the goal for agent's movement, for example, to reach a predefined destination or to avoid other agents; and the preference the agent chooses to accomplish his goal as fast as possible. The behaviors of an agent are further categorized into three phases, one is strategic phase, another is tactical phase and the final one is the operational phase. In strategic phase, a global destination is determined for each of the agent so that agents always try to move towards their destinations. In tactical phase, the agents try to avoid other agents and obstacles based on the psychological factors introduced before. In the operational phase, the agent calculates the speed and direction based on decision made by previous two phases to determine its movement. This model can realistically simulate the movement behavior and the interactions among agents. In[51], a model

for simulating crowd at Tawaf area is presented. This model is featured with two levels of movement of pedestrian, the macroscopic movements and microscopic movements. The macroscopic movements define the way finding behaviors or navigation behaviors of pedestrians while the microscopic movements define the interactions among pedestrians when they are performing macroscopic behaviors. Such interactions include avoiding each other or help each other, etc. The microscopic movements are carried out using a cellular automata and the macroscopic movements are implemented using path tables. This work is still at its early stage and author plans to implement several simulator in the later stages of the proposed work. In [52], author proposes a layered approach to model the dynamics of pedestrian crowd. The surface of the 2D environment is divided into different layers to indicate the occupancy, the position of static obstacles and possibly the dynamic obstacles situated in the environment. These layers contain the necessary information for agent to consult for navigation purposes. Finally, the agent utilizes the layered environment and uses Markov decision process and semi-Markov decision process approach to find the correct move given its occupying cell. In [53], author builds an agent based model based on cellular automata to simulate pedestrian's behavior. This approach uses a matrix to represent the agents' preferred movements on all directions and uses dynamic floor field to model the trace of the pedestrians' movement. It uses static field to represent attractive point such as exit on the map. To model the comfortable distance the pedestrian seeks to keep from others, the author uses the concept of repulsion field which defines the force the pedestrian agent puts upon its neighbor cells. These concepts together determines the transition probability an agent moves from one cell to another. In [54], author proposes a model for simulating the pedestrian crowd dynamics. This model is featured with an environment model and an agent-based model. The environment mode is divided into three different layers. At the

top layer, a route map is utilized to represent the structure and connectivity among regions in the map. The second layer consists of a navigation map describing the paths to round over obstacles in each region. At the lowest layer, the description about the information of each objects is presented. The route map is a topological graph implemented with entries and exits in the regions representing the nodes of the graph. The distance between the nodes of the graph becomes the weight of the edge of the graph. To decide a path of an agent, the agent's destination and position are added to the topological graph and a shortest path algorithm is utilized over the graph to find a suitable path for the agent. This approach is similar to the approach to plan path and navigation for the agent proposed in this dissertation. Author in [54] also utilizes social force theory to model the avoidance and interaction between agents.

In most of the work listed in this section, the agent-based model is constructed with two different components, one is a model for the environment including obstacles and topological structure of the environment, the other is a model for the decision and action of the agent including macroscopic behavior such as navigation and microscopic behavior such as avoiding each other. Inspired by these works, in this dissertation, the agent-based model proposed as a high-resolution and low abstraction level model also contains a model for describing the properties of the environment and a model to define the agent's behavior such as navigation and avoiding each other. Furthermore, we adopt a rule-based approach to define the agents' decision on how to perform the behaviors under different circumstances.

2.2.2 Coarse Models

Coarse models in pedestrian simulation include those model the pedestrian moving dynamics as a queuing network and those model the environment as a graph. It yields great computational performance advantage over the agent-based models or other microscopic models.

Some previous research have been conducted in building and utilizing coarse models. In [55], the author discusses both of the drawback of using merely coarse grained network models and merely fine grained network models to simulate the occupants' behaviors in evacuation scenario and proposes to combine the two approaches together to increase the accuracy of the model and at the same time reduce computational cost. In the simulation, the unprotected area are modeled using fine network. In the fine network modeling approach, the whole area is divided into a number of grid cells. Each cell can be occupied by exactly one occupant. By doing this, the author aims at simulating the interactions among occupants and the impact of the interactions on evacuation speed of the evacuees. The protected area is modeled using coarse network, where each node of the network represents a room or a segment of the corridor. Since movements of evacuees in the protected network are more orderly organized, simulation of interactions among evacuees are not necessarily needed. Instead of using a coordinate to indicate the occupant's position as in fine node, the coarse node uses a one dimension value named "distance" to represent the position of the occupant. Experiment shows that the integrated approach has better computational performance than using just fine network and can represent more details than using just coarse network approach. In [56], the author develops a model based on network to reproduce the pedestrian's behaviors such as collection, spillback and dissipation. The target environment is divided into grid cells with hexagon shapes. A variable called "potential" is utilized to indicate the distance of each cell to the exit and the minimum number of movements an occupant must perform to reach the exit. The grid cell representation of the building structure is then translated into a graph whose vertex represents each cell in the hexagon grid and edge represents the pedestrian flow from one cell to another. The directed edges of the graph are generated in such a way that the edges always point from a vertex of cell that has a higher

potential to a vertex of cell that has a lower potential. The result of simulations reveals that this model can realistically simulate the behavior of collection, spillback and dissipation of pedestrians. In[57], author proposes to use state dependent queuing network to represent and simulate the flow dynamics of the pedestrians. The author proposes to divide modeling procedure into three different stages: the representation stage, analysis stage and synthesis stage. In the representation stage, the network topology of the underlying building or facility is created with the rooms and corridors being treated as the nodes in the network. In the analysis stage, the performance characteristics of the queuing network is determined. In the final stage of synthesis, once the performance characteristics are determined in the analysis stage, the performance of a queuing network can be assessed and evaluated to solve various design problem.

Coarse models listed in this section normally involves modeling the environment as a graph with edges and vertex or a network with routes and nodes. The benefit of doing this is to reduce the computational complexity of the simulation. Coarse model is suitable in the condition that a large number of pedestrian or occupants are involved. Under such circumstances, agent-based model may require more computational resources which sometimes is impossible. In this dissertation, we proposed a simple graph model which models the building structure as a set of vertex and edges, and models the occupants as flows in the network. This model is more efficient than the agent-based model when large number of occupants get involved.

2.3 Applications of data assimilation

In this dissertation, we use data assimilation method to meld the observation derived from sensors and the prior knowledge represented by the models. Data assimilation involves incorporating the observation into the computer model to obtain the best estimate of the hidden state of the system and use the estimated state as the starting point for subsequent simulation to

improve the accuracy of the simulation. In [58], the author introduced the application of data assimilation in earth science by utilizing the data provided by hydrologic remote sensing. Although all of the data assimilation approaches are to combine observation and the model to get best estimate of the system state, the algorithms used to serve this purpose are different. The method suggested by the author includes a simple data assimilations system and variational data assimilation system. The simple data assimilation system minimizes a cost function that represents the difference between the estimated state and true state of the system. It derives the calculation of a gain based on the model variable uncertainty and observation uncertainty. However, when the dimension of the system to be estimated is high, the simple data assimilation system will fail to produce the estimate since the analytic solution is impossible when the evolution of system state becomes non-linear. To solve this problem, the author introduces the variational data assimilation techniques including 3DVAR (three dimensional variational), 4DVAR (four dimensional variational) methods, Kalman filter and its extended version. In [59], author applied the data assimilation approach to estimate the rainfall in urban area. The approach of data assimilation proposed in work of [59] involves several steps. In the first step, the model evolved from time step t-1 to time step t and the observation is available. In the second step, the prediction derived from the model is compared with observation to calculate a residual error between the prediction and the model. In the third step, the model state is corrected using the residual error calculated from the second step. The corrected model state is then used as initial condition for the subsequent simulations. In [60], authors apply data assimilation in wild fire spread simulations to estimate the state parameters of wild fire. The authors utilize particle filter algorithm to estimate the fire front from sensor deployed in the area of the wild fire. The particle filter algorithm is a Monte Carlo based method to recursively estimate a system's hidden state

using available observations. It uses samples drawn from a proposal distribution to approximate the posterior distribution of the target system. Each samples are associated with an importance weight to indicate its probability to be resampled. In the research carried out in [60], the proposal distribution of the particle filter algorithm is chosen to be the transition prior defined by a discrete-event simulation model of fire spread, therefore, the weight of each particle is calculated based on the likelihood of the particles. The particle filter algorithm introduces re-sampling step to eliminate the samples with low weight and multiply samples with higher weight. This step solves the degeneracy problem where weights of all but several particles become 0 after several iteration. Experiments carried out in [60] demonstrate that using particle filter, the accuracy of the simulation can be improved. In [61], author utilizes ensemble Kalman filter for global oceanic data assimilation. Ensemble Kalman filter is a Monte Carlo variation of the original Kalman filter, it is similar to particle filter algorithm that it uses an ensemble set to represent the random samples of the prior distribution. Upon calculation, the ensemble Kalman filter replace the covariance matrix in Kalman filter with the sample covariance. This method is proved to have better performance than particle filter algorithm when it is suitable to use, although it requires that the noises of the system and observation follow normal distribution. In[62], author proposes a data assimilation framework for on-line modeling of water distribution system. Data assimilation method in this work is used to predict and correct the state of the system represented using a numeric model. The water demand forecast model is developed to be a multivariate time series model. The methods used to assimilate the observation data into the developed model are Kalman filter and ensemble Kalman filter. The author compared the results in experiments for both of the methods and results show that ensemble Kalman filter provides better estimation accuracy than the Kalman filter while Kalman filter has its advantage in computational cost.

In most of the literatures, the dynamic model used in the data assimilation framework is numeric model and data assimilation is normally applied in atmospheric, oceanic or earth science. There are no literatures about the application of data assimilation in context of agent-based simulation. In this dissertation, we apply data assimilation in an indoor environment and assimilate the observations generated from occupancy sensors into an agent-based model to estimate the position of occupants. This is considered to be a novel application of data assimilation.

2.4 Particle filter algorithms

Data assimilation involves combining the observation with dynamic model to produce estimate of the system. A number of methods can be utilized to serve this purpose. A classical solution is Kalman filter and its extensions[24]. Kalman filter is an optimal solution for estimation problem of linear and Gaussian system. In order to utilize Kalman filter for estimation, it is required that the target system can be written in linear equations and the processing noise and observation noise follow Gaussian distribution. These requirements greatly limited the applicability of Kalman filter in estimating systems that are non-linear, non-Gaussian and have multimodal distribution. Although researchers developed various extensions for Kalman filter such as extended Kalamn filter, unscented Kalman filter and ensemble Kalman filter[63] that can handle non-linearity to some extent, most of the work still require that the system noise follow Gaussian distribution. On the contrast, particle filters [25] do not have any assumption about the linearity and distribution of the target system. The target system can be linear or non-linear, continuous or discrete, and the distribution of the errors of the system does not necessarily need to be Gaussian distributed. Furthermore, it allows the target distribution of the system to be multi-modal. In this dissertation, due to the fact that people's movement is

highly irregular and unpredictable, it is difficult to express the movements of occupants using analytic form. Therefore, we use agent-based model to simulate the occupants' movement in the building. Agent-based model better captures the system dynamics of occupants in the building; however, it also brings difficulty to the estimation without having analytic form. This problem can be solved naturally using particle filter algorithm. This is because, as mentioned previously, particle filter uses a set of samples to represent the posterior distribution and does not have assumption about the target distribution to be estimated. In this dissertation, we intend to show how particle filter algorithm help the estimation of occupancy dynamics.

Particle filter algorithm has a long history, it has been widely used in problems such as target tracking and signal processing. It has a number of variations. Most of these variations aim at solving two major problems of the particle filter, namely sample impoverishment and deprivation. Sample impoverishment occurs when after several steps, all the particles are multiplied from a small set of samples and becomes identical. This happens due to the nature of re-sampling which keeps eliminating particles with lower weight. Particle deprivation occurs when there are no particles are in the vicinity of the true state of the system. This happens when the number of particles is not large enough to cover the state space comprehensively. Methods to solve these problems include moving the particle around its original position at state space [31], adjusting the size of particles utilized for estimation dynamically [32], and dealing with high dimensional state space analytically[33]. In [31], the author proposes a method to increase the diversity of the sample space. This is done by introducing a re-sample move step at each iteration of the algorithm. The re-sample move step first sample new particles according to the weight of each of the particles and then move the particles to some new positions in state space. By doing this, the diversity of the sample set has been increased because each particle has been moved

randomly. As a result, the coverage of the samples on state space has been enlarged. In [32], the author proposes a method that dynamically adjusts the size of the particles according to Kullback-Leibler distance between the distribution estimated and the true distribution. This is done by dividing the state space into multiple bins. The particles that fall into a bin make that bean non-empty. After counting how many non-empty bins are there during estimation procedure, the number of non-empty bin is used to calculate desired number of samples using equation derived from Kullback-Leibler distance. The algorithm keeps inserting new particles until the number of particles equals to the number of desired particles. In [64] and [33], authors propose to use Rao-Blackwellized particle filter to deal with the high dimensional state space of the target system. The core concept of Rao-Blackwellized particle filter is to decrease the number of particles required to achieve same performance as regular particle filter. The basic idea is to divide the state of each particle into root nodes and leaf nodes which are linked using a dynamic Bayesian network. Since the leaf node can be derived from the root node analytically, we only need to sample and estimate the variables at the root nodes. By doing this, the dimension of the system need to be estimated reduces. Besides these method, some recent work reveal new methodologies to combat the deficiency of particle filter. In [65], the authors proposes a new method based on Genetic Algorithm to improve the performance of particle filter and to deal with the particle impoverishment problem. The basic idea is to treat each particle as chromosome and to apply cross over and mutation operators on the chromosomes. The cross over operation generates new chromosomes from exist chromosomes so that the diversity of the sample space is increased. Consequently, the sample impoverishment problem has been relieved. In [66], authors address the problem of tracking multiple target that large number of particles are required in order to cover the high dimensional state space and propose a method to deal with the high

dimensionality of the state space. This is done by dividing the state space into sub dimensions and apply a particle filter on each sub dimension of the state space. In other words, instead of calculating the posterior distribution of the target system as a whole, the method aims at calculating the marginal distribution of sub dimensions of the state space. Specifically, in the application of multiple target tracking, the sub-dimension is each target and the problem is transformed to track each target using a single particle filter. Simulation and experiments show that this method has advantage in performance over the standard particle filter algorithm. In[67], the author introduces an auxiliary variable during the sequential importance re-sampling phase. This variable is an intermediate variable and only assists the simulation, therefore it is called auxiliary variable. And the algorithm is named as auxiliary particle filter. The auxiliary variable improves the adaptability of the particle filter algorithm and overcome several deficiency of the sequential importance re-sampling. In [68], the authors proposed to use unscented Kalman filter to generate proposal distribution for the particle filter algorithm. This method generates a better proposal distribution that can handle the condition that the likelihood is narrow and can have large overlap areas with the posterior distribution. Experiments are carried out to show that particle filter algorithm with unscented Kalman filter has better performance than other sequential estimation algorithms.

The works listed in this section aim at improving the efficiency of particle filter algorithm. The approaches include enlarging the state space covered by the samples, adjusting the number of particles, dealing with high dimensional state analytically, introducing cross over and mutation operator from genetic algorithm, dividing the state space into sub dimensions, introducing auxiliary variable and incorporating Kalman filter to generate proposal distribution. Inspired by these methods, in this dissertation, we proposed a novel method to increase the

diversity of the samples. This is done by dividing the state space into component set and re-sampling from the component set instead of re-sampling from the entire sample set. This method efficiently increases the ability of the particle filter algorithm to use limited number of particles to cover larger portion of the state space and deals with high dimensionality of the state space.

2.5 Target Tracking

To get location information of occupants in a building, the most direct method is tracking. When agent-based model is utilized in estimating the position of occupant, it is very similar to what is investigated in the field of multiple-target tracking problem; that is, given observations and movement model of targets, calculate the target's trajectory. In this section, we give a literature review on the methods and applications of multiple target tracking problem. Traditionally, multiple-target tracking problems (MTT) are mostly investigated under the scenario that observations are from sensors with strong functionalities such as radar or sonar. Data obtained from such sensors normally provides accurate but ambiguous information of each single tracked target; MTT is then divided into two different sub problems: 1) Data association problem, which links a specific target to the observation data it generates. 2) Single target tracking problem, which identifies a target's trajectory in clutter environment [69]. In [70], author discussed the application of multiple hypothesis method applied in target tracking algorithm. The multiple hypothesis method derives a series of hypothesis according to the position of the target in previous time step and then uses upcoming observation to update the data association hypothesis. The observations are used to generate new hypothesis of the tracks of the targets and these tracks are evaluated lately to determine which tracks (data association) should be kept and which tracks should be deleted. In [71], authors utilize sequential Monte Carlo method (particle filter) to track unknown number of targets. In this article, authors address

that the main difficulty for tracking multiple target is from assigning measurements to each of the targets. The authors apply a combination of Gibbs sampler-based estimation with standard particle filter to estimate the assignment from measurement to each of the targets. The method is also extended to add an appearance test and disappearance test to test whether there are new targets appears in the surveillance area. Finally, the method is extended to estimate under the assumption that multiple data receivers get involved. In [72], the author propose a method called joint probabilistic data association to solve the data association problem, the basic idea is to enumerate all possible association between predicted target position and measurement. The algorithm then calculate an association probability for each of the association. This probability is then used as a weight for the state estimate using some filtering algorithm. The state of the system is a sum of all the weighted state estimates.

These set of work is built based on the assumption that sensors with strong functionalities such as radar or sonar are utilized. Unfortunately, they are difficult to be applied to indoor environment. The burst development in wireless sensor network technology provides a feasible solution for target tracking in smart environment. A variety of methods has been proposed to solve the target tracking problem under an office or indoor WSN setting using some well functional sensors [73][74][75]. The recent contribution of [76][77] reveals a fundamental analysis and lower bound of tracking multiple targets with binary proximity sensors in both 1D and 2D situation without any assumptions of the behavior of moving targets. Since binary proximity sensors are the least functional sensors, it provides a good fundamental analysis of the extendibility to similar tracking approach with more components. In the work of [76][77], the sensors are deployed relatively close to each other; the distance between consecutive sensors is set to be less or equal to their detection radius, making these sensors actually overlapping with

each other. The effectiveness of the developed tracking algorithm under sparse sensor deployment is not investigated in detail. On the other hand, the performance of the algorithm mentioned in the paper has certain limitations in 2D space due to the increasing dimensions while tracking multiple targets.

The target tracking problem is very similar to what is investigated in this dissertation. The major difference comes from the model used in the estimation procedure and the assumption about the measurement data. In traditional target tracking research, since the sensor used to obtain measurement is normally assumed to be radar or ultrasonic sensor, the trajectories of targets are relatively easy to follow. As mentioned previously, the main difficulty is then to associate the measurement data to the origins of measurement. In this dissertation, the sensor used to obtain measurement is assumed to be binary proximity sensor, it is a passive infrared sensor that can only detect whether there is an occupant in its detection area. As a result, the major research challenge comes from the estimation problem itself, i.e. how to capture the dynamic of targets' movement and estimate the positions of the target using incomplete observations.

3 SMART ENVIRONMENT AND AGENT-BASED SIMULATION OF SMART ENVIRONMENT

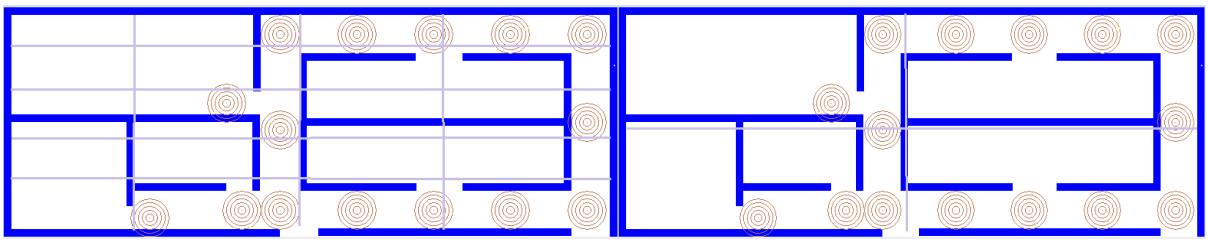
3.1 Smart Environment

A smart environment is any design that utilizes knowledge of the environment's context to aid accomplishing various tasks such as navigation, scheduling energy consumption, notification of activities and evacuation planning. The term "smart" refer to the ability of the environment to automatically detect and analyze the contextual information of the environment

to make decisions or to provide valuable information. The contextual information is defined as features of information of any entity in the smart environment. For example, the location of people, the gesture or posture of an individual, the states of certain objects such as doors or computers, the events ongoing and the activities performed by occupants are all considered to be contextual information of the smart environment. To collect contextual information, a typical smart environment is equipped with various sensors, including those detect the density of the gas, those detect whether there is an occupant inside the detection area, those detect various states of the objects and those can measure the position of occupants directly. The data obtained from these sensors are fused in a central control system for handling configurations of energy supply, air conditioner or providing information related to the current context to the occupants. Consider a scenario that a smart home application is capable of controlling the cooling system. We assume the activity of occupants can be labeled as three different categories: sleeping, awake and absent. From various sensors deployed in the home, the application uses some algorithm to discover the activity of the occupants. For example, if the occupancy sensor in the bedroom reports that there is no individual in there, then the application can conclude there is 90% probability that the occupants are in the state of being awake or absent. By knowing this, the smart home application turns off the cooling system in the bedroom to save energy consumption. Moreover, by applying machine learning technology and providing a set of training data to the smart home application, the application is able to predict when occupant will perform certain activities. In the bedroom example mentioned above, it is possible for the smart environment application learn from the data and discover that the occupant normally goes to bed at around 10:00PM. To utilize this information, the smart environment application turns on the cooling

system at around 9:30PM so that when the occupant goes to bed, the air is already cooled and his comfortableness is maximized.

The information gathered by sensors in a smart environment can be categorized based on two different resolution criteria: spatial resolution and logical resolution. The spatial resolution defines the spatial granularity of occupancy problem to be investigated. Under high resolution, each individual occupant is considered to be the research subject while under low resolution, a group of individuals in a specific area of the building are considered to be the research subject. The logic resolution defines the logic granularity of occupancy problem to be investigated. Under high logic resolution, answers to questions such "where is A" are the outcome of the solution to the occupancy problem. Under low logic resolution, answers to question such as "what is happening in the building" are the outcome of the solution to the occupancy problem. The purpose of this dissertation is to build an estimation framework that is capable of estimating the building occupancy at different spatial resolutions so as to handle different situations. On the other hand, the possible extensions on integrating questions from different logical resolutions can also be added. The different resolutions of occupancy problem are illustrated in figure 3.1 and 3.2 where 3.1 presents the spatial resolution of the occupancy problem and 3.2 presents the logic resolution of the occupancy problem.



(a) Spatial high resolution (b) Spatial low resolution

Figure 3.1 Spatial Resolution

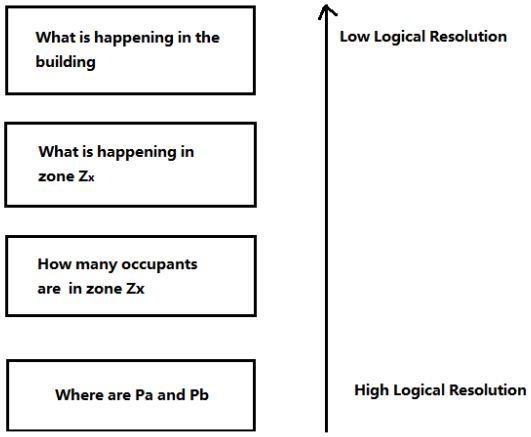


Figure 3.2 Logic Resolution

Among various information that can be derived from sensors deployed in a smart environment, the most important one is the location information of occupants, i.e, the occupancy of the building. This is because occupancy directly related to many aspects of contextual information of a building such as energy consumption and occupants' activity. In terms of energy consumption, occupancy directly informs which parts of the building are in need of energy and which part of the building do not need energy at all so that the central control system can effectively handle the energy supply. In terms of activity discovering, occupancy is the key factors to recognize the ongoing activities of occupants. For instance, when the occupants are holding a meeting in the building, the detected occupancy in the conference room will be dramatically increased, in this case, the pattern of occupancy change directly indicates the activity performed by occupants. Because of the importance of location information, in this dissertation, we focus on deriving location information of occupants from sensors equipped in the building. Other research topics related to smart environment such as activity discovery, activity recognition and learning from data to building occupancy model are out of the scope of this dissertation. In the work of using agent-based model to estimate the position of occupants,

we assume that the smart environment is equipped with binary proximity sensors, which reports 1 when one or more occupants are within its sensing area and reports 0 otherwise. These sensors have low resolution in detecting the movement of occupants and have the following features in terms of resolution, accuracy and ambiguity:

1. The sensors provide anonymous position information of occupant. Binary proximity sensors do not provide measurement of occupant's identity.

2. Location information obtained is ambiguous when there are multiple occupants in the sensing area of a sensor. The binary proximity sensor does not have the capability of identifying the number of occupants in its detection area.

3. Data collected are subject to sensor errors and environmental noise.

4. Sensors are deployed sparsely. The occupants' locations become latent variables while being outside a sensor's detection area.

These features make it impossible to directly measure occupants' positions using binary proximity sensors. To overcome the difficulties brought by these features, we build an agent-based model to simulate the occupants' movements and use particle filter algorithms to assimilate sensor data into the agent-based model dynamically. The data assimilation provides estimations of occupants' states in real time, which can be used to supply initial conditions for future simulations to improve simulation results.

3.2 Concept of agent-based simulations

Agent based model is useful for simulating people's movement in smart environment. It is a modeling paradigm that models a system by modeling its individual components as agents. An agent is an autonomous entity that performs certain actions and interacts with other agents in an environment. Each agent is assigned with a set of simple rules to define what actions the agent

will perform and how will they interact with each other. The dynamic of the system is the sum of dynamics of all of the agents in the system. The philosophy of agent-based model is "the whole is greater than the sum of parts" because complex system behaviors can be generated from simple rules of each agent. Typically, an agent-based model consists a set of agents, a set of rules each agent will follow, a set of interaction topology connecting one agent to another and a non-agent environment. For example, an agent-based model that models the social network would contain a set of agents where each of the agents represent a single person. Each agent is assigned with the behavior such as talk and argue. Assuming that agents are all interconnected with each other and a variable called "relationship" represents the weights of the connections. When two agents talk to each other, the weight of the connection between the two agents increases. When two agents argue with each other, the weight of the connection between the two agents decreases. It would be interesting to see how the social network will evolve given certain intitial distribution for the agent to perform different behaviors. This is a typical agent based model. The agent-based modeling and simulation has many advantages over the traditional system dynamic modeling paradigm. The most important features are probably that the agent-based model can represent the heterogeneity of the entity of the system and the ability of agent-based model to simulate the interaction among different agents. These two features are otherwise impossible to implement using a top-down approach. The heterogeneity of the entity of system means that each agent can have its own preference of performing behavior or even its own behavioral rules to follow. Without agent-based model, the entities of a system have to be modeled in a homogenous manner, which is not practical in many real world scenario. (For example, each person should have its own rules of performing behavior and therefore, entity in

social system should be modeled in a heterogeneous manner). Also, the interaction among entities of a system is difficult to specify without a proper agent-based model.

Because of the features of agent-based modeling, it is particularly suitable in simulating people's movement behavior. The movement of people involves a process that is highly heterogeneous and interactive. Each individual can have his own choice of route, speed and destination and acts according to its own preference to move to a destination. When individuals are in close proximity, they will have to interact and perform maneuver to avoid each other. Agent-based model captures the two features of the movement behavior of occupants naturally.

A number of researchers have conducted their work on building models for peoples' movement behavior in building environment [20][78]. Besides homogeneity and interaction, agent-based modeling and simulation also has other features. For example, each agent can be considered as a self-contained entity with its own attributes. These attributes can be added, removed or shared with other agents easily and define the identity of the agent. With these attributes, the agent can be recognized by other agents and the environment. An agent is also autonomous, which means that its action or behavior can be easily defined using computer program. Their functionalities are independent of environment and other agents. By observing the environment and other agents, an agent determines its behavior to perform. The agent's behavior can be of any form. It can be as simple as a single equation or it can be complex procedure. Furthermore, an agent has its own state, which determines the behavior it will perform. The aggregation of the state of each agent becomes the state of the whole agent-based model. The state of the system at some time step is determined by the state of system at previous time step and determines the behavior of the system as a whole. Moreover, the agent can have complex mechanism such as learning and adaption. This means the agent can adapt to adjust its

behavior and action rules according to a learning procedure. The learning procedure involves utilizing accumulated experience of the agent to decide its preference of behavior. This requires that the agent has some sort of memorizing mechanism to store its perceived information from the environment and other agents. Finally, an agent can be goal directed. This means that each agent can be assigned with a task. The behaviors of the agents are designed specifically for accomplishing this task. This is particularly useful in simulating movement behaviors of people because people normally move according to a predefined route which can be considered as a goal directed behavior. [85]

In the previous research of agent-based modeling and simulation, agent-based simulation is usually used as an offline tool to help the system design without incorporating observation data. However, the trend of utilizing observation data in design and application of agent based models has received much attention in recent years. For instance, the work of [79] carries out micro-simulation and initializes an agent based model using contextual data. The work of [80] discusses the potential to combine data mining technology and agent based model together so that the two technologies utilize each other to solve problems in their own domains. A framework to integrate data into the procedure of modeling, validating, calibrating and analyzing of agent-based models is proposed in [81]. Another framework focusing on integrating data into the modeling procedure of agent-based model is presented in [82]. Furthermore, how agent-based models can work with data and observation models from other discipline such as GIS is discussed in [83][84]. Nevertheless, most of these works focus on incorporating observation data into modeling or validation procedure; none of them utilizes real time data to calibrate the simulation and to estimate of the system's state under study. In this dissertation, we construct an agent-based model to simulate the movement of occupants in building and develop a data

assimilation framework to incorporate observation data into the model to estimate the positions of occupants in the building.

3.3 Framework for the agent-based model of smart environment

Agent-based modeling and simulation is widely applied in emulating people's movement dynamics in different scenarios. It is usually used as an offline tool for investigating the behavior pattern of the movement of people. By modeling microscopic behavior of individual agent, defining their decision making process and describing their interactions with each other, macroscopic pattern of the target set of people to be emulated can be emerged. In building environment, the agent-based model is constructed by modeling each occupant as a single agent and treating their movement as deterministic. Comparing to simply modeling the movement of each occupant as linear procedure with noise, an ABM can better describe the real non-linear dynamics existed in the movements of a building's occupants such as avoiding each other and irregular maneuver. Furthermore, while applying ABM in the data assimilation framework, adding complex human behavior model as an assist to increase the quality of transition prior becomes possible.

The developed agent-based model includes four basic categories of components: zones, obstacles, occupants and sensors. Zones are defined as the locations which are available for users of the building to occupy. Obstacles are locations occupied by an object (wall, desk, chair, etc). These two categories of components are modeled as stationary polygons in a 2D rectangular environment. To model occupant as autonomous agent, the simplest way is to describe each single agent *i* using a state space containing merely two elements and follows a linear position update procedure:

$$\{l_k^i, v_k^i\}$$
$$l_k^i = l_{k-1}^i + v_k^i \qquad (3.1)$$
$$v_k^i = g(l_k^i) + Q$$

, where p_k^i , v_k^i represents the position of agent *i* at time *k* and the velocity. And g is some function to determine the occupant's velocity according to its current location and *Q* is other factor contributed to determine the occupant's velocity.

The limitation of this simplicity is that it is hard to determine the value of v_k^i formally under certain circumstances, if random speed vector is used, the simulation will end up with the situation that all of the agents are performing random work which makes the simulation unrealistic.

We notice that people in commercial building usually move rationally; their movements are highly associated with clear purposes. For example, occupants going for lunch (schedule) may set its destination to the kitchen room and move there directly (route); when a meeting is hold (event) in the conference room, people are more likely to stop current activity and set the conference room as the destination. In short, there exists a structure with multiple layers by which the occupancy dynamic is driven. Combing with (3.1) as movement model, we describe the four layers of the occupant model as following in figure 3.3, note that each schedule-route-move procedures represents the inner logic of agents in the system:

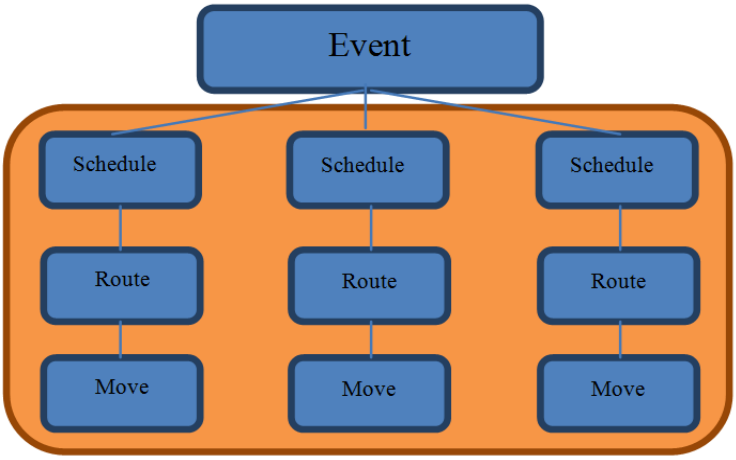


Figure 3.3 The four layer model of occupant model

Each layer of this model is explained as follows:

Event: Event in a building determines the potential trends of the occupants' activities. For example, in normal situation, occupants follows their own schedules; however, if there is a fire emergency, occupants will stop current schedule and make a plan to rush to the exit .

Schedule: Schedule defines the sequences of occupant's moving behaviors according to their daily activity. An employee in the building probably has schedules such as going kitchen for lunch, staying in his office for some time or meeting his employer in another office. An event occurring in a building environment can change the schedule of the occupants.

Route: In our model, each single plan in an occupant's schedule can be abstracted to an initial position and a destination. Once an occupant determines to execute some certain plan, a route from initial position to the destination will be generated according to the prior knowledge about the floor map.

Movement: Movement of the occupant will follow the route he/she generates. The position of the occupant is updated using (3.1).

Implementing different layers of this model can produce different outcome. For example, to model an event happening in the office, say, a meeting in the conference room, the simulation first generates an event and then dispatches it to agents that will be affected by the event. The present schedule of the agents will be terminated and a new schedule called "move to conference room" will be generated for each agent affected by the even. The agents then set conference room as its destination and generates a route from its present position to the location of conference room. By moving along the route it generated, agent performs a moving to destination behavior responding to the event. The hierarchical agent-based model makes it easy to add new behaviors to the agents so that they can better emulate the occupant dynamics in building environment.

While the benefit to implement each level of the proposed hierarchical agent-based model is obvious (making the simulation more realistic), it adds complexity to the models. In this dissertation, the goal of constructing an agent-based model is to work with the particle filter algorithm and assimilate data so that occupants' states can be estimated. Since we do not intend to incorporate the personal schedules of agents into the model, we only implement the model at the bottom two levels. That is, we implement the routing level and the movement level of the proposed model. In our implementation, we only care about how agents generate a path from its current position to a destination and how agents move along its generated path. We do not explicitly investigate how the agents choose its destination, which means the destinations of the agents are generated randomly or according to some predefined rules. Although future work can be done so that the destination of the agents are generated according to survey data related to occupants' daily schedule and data collected from sensors, this is beyond the scope of this dissertation.

3.4 Behavior control of the agent-based model

The agent-based model models individual occupants' moving behavior such as navigating and avoiding each other. The model runs in a time step fashion. In general, an agent's position at time *k* is updated based on its position and velocity at time *k-1*. A simple equation-based model describing the physical law of position update can be used to simulate the movement of each agent. However, such a model does not specify how agents interact with each other to perform maneuver such as avoidance. Furthermore, it makes it difficult to model "planning" type of behavior, such as moving to a pre-determined destination and moving according to a schedule. To overcome the deficiencies of simple equation-based models, this work adopts behavior-based control for modeling agents' movement. The behavior-based control employs mechanisms, such as a rule-based system, in which behaviors that match a current world condition may fire subject to conflict resolution among competing alternatives [86]. The behavior-based model makes it easy to model different behaviors and the triggering condition associated to these behaviors. Specifically, in this dissertation we model each agent to have two basic behaviors in the smart environment: avoidance and move-to-destination (Navigation). We note that the main focus of this dissertation is on the data assimilation aspect, thus we do not model other behaviors such as emergency evacuation. Those behaviors can be added into the agent-based model if needed, and the data assimilation framework presented in this dissertation still works if more complex agent models are used.

The goal of the avoidance behavior is to enable an agent to perform appropriate maneuver when it is blocked by other agents or obstacles. Since the corridors and rooms in a typical building structure usually contain limited free space, it can result in congestion when multiple occupants are in vicinity of each other. When there is congestion ahead, occupants stop

moving to their destinations and try to follow some rules to resolve the congestion first. In our model, an agent performs its avoidance behavior according to three basic rules: keeping comfortable distance, moving sideways and re-planning a temporary path. The first rule, keeping comfortable distance, defines that an agent will always try to keep some distance from other agents. It moves to the opposite direction of another agent if its distance to this agent is smaller than a threshold. The "side way move" rule defines that if an agent is blocked by other agents on its heading direction, the agent scans its side ways to seek for a direction that is clear of obstacle and move towards that direction. The third rule, re-planning motion, is applied when the agent does not find a direction that is clear of obstacles while applying the second rule; when this happens, the agent tries to re-schedule a new route so that it can pass the obstacle. This set of rules helps agents to avoid each other while moving and when congestion happens.

The goal of the navigation behavior is to navigate an agent to a pre-defined destination so that the "planning" type of behavior can be modeled. We notice that occupants in commercial buildings usually move rationally; their movements are usually associated with clear purposes. For example, occupant going for lunch may set its destination to the kitchen room, then schedule a path from its current location to the kitchen room and move there directly; when a meeting is hold in the conference room, occupants are more likely to stop their current activities and set the conference room as their destination. In short, instead of walking around casually, an occupant usually determines a destination before starting to move. Once the destination is determined, the moving direction and speed of the occupant can be determined based on its current position and its destination. In our work, we implement the move-to-destination behavior using a waypoint graph.

3.4.1 Navigation behavior

With the bottom two levels of the hierarchical model being implemented, it is possible to model a basic behavior of occupants' activity: navigation. The goal of navigation behavior is to guide the agents' movement so they can arrive their destinations. To model the navigation behavior of the occupants, one must utilize the information of the building structure so that the agents know how to generate paths to arrive its desired location. A building structure can be abstracted to corridors, rooms and obstacles such as walls. Agent can move in rooms and corridors but cannot reach the regions occupied by walls. The problem of path generation is described as follows: given a starting position and a destination and a map of building structure, find a path from starting position to the destination without engaging any obstacles. To solve this problem, we abstract the structural information of the building as a way point graph. The way point graph defines a series of positions and connections between these positions that can represent the general structure of the building. The way points in the way point graph are set to the intersections between corridors and corridors as well as intersections between corridors and rooms. By doing this, two features of the way point graph are guaranteed: first, from arbitrary way point in the way point graph, there is a path to another arbitrary way point in the way point graph; second, since we choose the intersections between corridors and intersection between corridor and room as the way point, from arbitrary position location on the building map, there is at least one way point can be connected to without engaging any obstacles. The procedure of generating a path from position R1 to position R2 is shown as follows:

Assuming we have an office environment as shown in figure 3.4:

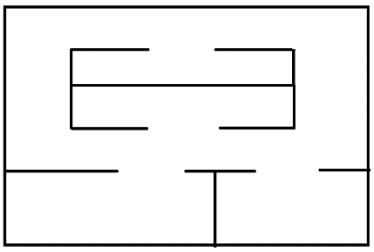


Figure 3.4 Example of office environment

While constructing way point graph, we set intersections between corridors as well as intersections between corridors and rooms as the way points, represented by black circles in figure 3.5. We connect each waypoint to its nearest waypoints to form a graph.

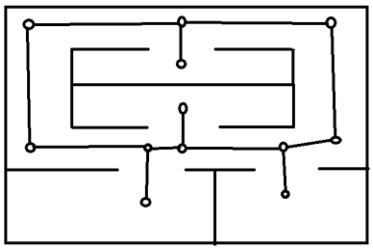
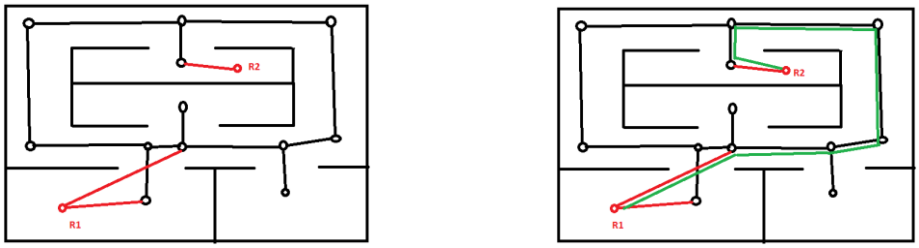


Figure 3.5 Example of waypoint graph

To generate the route we first connect the starting position R1 and destination R2 to all the way points in the way point graph. We then delete the connections that engage the obstacles. The modified way point graph is shown as red circles in figure 3.6(a). With the modified graph, find the shortest path from R1 to R2 to generate a route, which is shown by green lines in figure 3.6(b)



(a)Add starting positions to graph (b) Generate shortest path

Figure 3.6 Generation of the navigation path

Formally, let G represent a waypoint graph:

$$G = \{V, E, W\}$$
$$W = w(E)$$

, where *V, E, W, w* are sets of the vertices, edges, distance of the edges and function to return the distance of an edge respectively. For each element in *V*, a coordination is associated to indicate the position of waypoint. To generate a route from an arbitrary initial position p_1 to an arbitrary destination p_2, the waypoint graph is modified as follows:

$$I_1 = f(G, p_1)$$
$$I_2 = f(G, p_2)$$
$$V' = V \cup \{p_1, p_2\}$$
$$E' = E \cup e(I_1, p_1) \cup e(I_2, p_2)$$
$$W' = W \cup w(I_1, p_1) \cup w(I_2, p_2)$$
$$G' = (V', E', W')$$

, where *f* is a function to return all vertices in G that directly connected (i.e. without engaging any obstacles) to a given position *p* ; *e* is a function that returns all of the edges connecting vertices in *I* with *p* ; After we connect the position and destination of an agent to this waypoint graph , Dijkstra algorithm is applied to the new graph $G' = \{V', E', W'\}$ to find the one to one shortest path from p_1 to p_2. The returned vertices (waypoints) from the algorithm will then

be stored in a vector as sub-destinations of the generated node. The route of occupant d*i* is then described as a vector:

$$d_i = \{p_1, m_{x_1,y_1}, m_{x_2,y_2}, m_{x_3,y_3}, m_{x_4,y_4}, ...p_2\}$$

, p_1 represents the agent's current position; p_2 represents the destination and $m_{x,y}$ is the intermediate node in the route as guidance for the occupants' movement. Once the route is determined, the agent set its velocity to update its position so that it always moves to $m_{x,y}$ subsequently. Once the agent arrives an intermediate route point, the route point will be removed from the route vector and agent will set the next element in the vector d_i as its destination.

In summary, an agent generates its route to a specific destination from its current position with following rules:

1. Add starting position and final destination of the agent to the waypoint graph by connecting them to all the possible vertices such that the new edges do not cross any obstacles (walls in this particular example).
2. Find a shortest path from the starting position to the destination over the modified way point graph.
3. Store the route point on the found path as a vector so that agent set its velocity to navigate to the final destination through the intermediate destinations in the vector.

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 first element of the route point vector. When it arrives at that route point, the route point will be removed from the list and agent navigates to the next route point in the vector.

Note that the implemented navigation behavior assumes that occupant selects shortest path to arrive its destination. While this is sometimes not the case in real world scenario

(sometimes people does not select the shortest path to its destination), the behavior of select non-shortest path can be modeled by assigning a sequence of destinations to the agent to be modeled and let the agent arrive these destination sequentially.

The navigation behavior makes it easy to abstract real world scenario using this model. Since all the activities in a building can be described by change of occupants' positions, modeling these activity is simply a procedures to assign a set of destinations to the occupants. For example, to model an activity of conference meeting, we can simply assign the location of the conference room to the destination of each of the agent and let the agents navigates to that location using method mentioned above. To model agents' daily activities, we can assign different locations such as kitchen and office as agents' destinations to model the lunch and work activity. In short, with the method described above for navigation behavior, the activity and the movements of occupants can be easily described and modeled.

3.4.2 Avoidance behavior

Navigation is the core behaviors for modeling occupancy dynamic. However, occupant does not always perform this behavior. For example, if the number of occupants in a small area is large, it is more likely that the occupants will be blocked by other occupants while trying to move to their destinations. When this happens, the occupant should stop keeping moving to its desired destination and try to avoid each other first. This comes to another important behavior in modeling occupancy dynamics in a building: avoidance. The goal of avoidance behaviors is to resolve the congestion caused by aggregation of large amount of occupants in a small area in the building. Specifically, we developed a set of rules for each individual agent that captures the avoidance maneuver of occupants while moving inside a building structure. The rules for agent to avoid each other is defined as following:

1. Keeping comfortable distance:

   Let $d_{a,b}$ denotes the distance an agent a is from its nearest neighbor b within a range specified by r. If $d_{a,b}$ is smaller than a threshold, then agent a will move towards the opposite direction to b with speed s.

2. Reschedule route

   If the distance to the nearest obstacle on the agent's heading direction is larger than a threshold, then the agent moves to its next destination, this destination is generated using waypoint graph and is stored as a queue.

   If the distance to the nearest obstacle on the agent's heading direction is smaller than the threshold. Then the agent scans an angle in the range $-\pi/8$ to $-3\pi/8$ and $\pi/8$ to $3\pi/8$ from its heading direction. By doing this, the agent seek for a direction that is clear of obstacle and move towards that direction.

   If the agent fails to find a direction by scanning, it wait for one time step.

   If waiting time of the agent exceeds a threshold, it adds k points on the controversial direction of its heading direction to see if it can move to sideways on these points; if it finds one point where a side move to right or left can be scheduled, it add both that point and a point to the right or left of that point to the temporary route. The agent will always move towards route point of the temporary route first; after all the destination of temporary route is reached, it moves to the destinations generated using way point graph.

   If the agent fails to find a point among the k point on the controversial direction of its heading direction that it can move to some direction that is clear of obstacle, it means this agent is deeply blocked by other agent; it then scan an angle in the range $-\pi/2$ to $-\pi$ and $\pi/2$ to π from its heading direction to try to move backward.

This set of rules guaranteed that the agents avoid each other while moving even when congestion happens.

The navigation behavior and avoidance behavior together define the behavior model of the occupants. These are two basic behaviors that can represent the occupancy dynamics in the building environment. In the simulation procedure, agent switches between these two behaviors according to different situations. We notice that in the real world scenario, when a congestion occurs in the building, occupants usually stop moving towards their original destination and try to resolve the congestion first. This is because without resolving the congestion and make the path clean of other agents, it is impossible to perform navigation behavior properly. Therefore, in our effort for building the agent-based model, we set avoidance behavior has higher priority than the navigation behavior so that when there is congestion ahead, agents stop performing navigation behaviors and instead try to avoid each other so that they can make their way clean. This is done by employ behavior selection mechanism that based on whether the agent is blocked by another agent ahead. If it is blocked, then the agent choose to perform avoidance behavior. The navigation behavior is only performed when there are no other agents on the moving direction of the agent.

In this dissertation, the agent-based model represents the occupancy dynamic of the building in most details. It has lowest abstraction level because it models agents' attributes such as speeds, locations, intentions and interactions exactly as same as real world entities. While using this model for estimating occupancy dynamic, the advantage is that since it is a high resolution model, it is possible to also estimate occupancy at high resolution. However, there is a problem that the resolution of estimation is sometimes limited by the resolution of the sensing device used to generate observations. For example, if a gas sensor is utilized to estimate the

occupancy level in the building, using agent–based model will not provide expected estimation outcome since gas sensor only measures the density of the various gas. It can only indirectly indicate the distribution of people in the building rather than directly measure the positions of the occupants. As a result, estimation using agent-based modeling and simulation may not produce expected outcome and only adds to the complexity of the estimation. Furthermore, in the scenario that massive occupants are getting involved, tracking exact position of each single agent becomes impossible; it makes more sense to estimate the number of occupants in each specified zones instead of estimating the position of each occupant. Therefore, low resolution models which represent the occupancy dynamics at higher abstraction level are needed for considerations mentioned above. In this dissertation, we build a coarse model for estimation at higher abstraction level to deal with the scenario of large number of occupants get involved. Instead of modeling the location information of occupant as each individuals' exact position, the coarse model divide the environment into different regions and model the state of the system as number of occupants in each region.

3.5 Senor Model

A typical smart environment is equipped with various sensors. The agent-based model for smart environment must also include a model to describe the functionality and deployment of the sensors. Although complex sensor such as camera sensor can produce high volume information about the position of occupants, in this dissertation, we assume the sensors utilized to detect occupancy are low resolution sensors. Specifically, we assume the sensors to be binary passive infrared sensor (PIR). The PIR sensor has basic ability to detect the motion of occupants. It emits infrared light beans and detects whether there are objects engaging these infrared light beans. If an engagement is detected, the sensor reports a high voltage on its output port, otherwise the

output port continuously produces low voltage. These sensors are assumed to be mounted at the ceilings. The field of view of the sensor ranges from 3 meters to 10 meters. In this research, we define the field of view of the sensor (in other words, the diameter of the sensors detection area) to be 50 unit distance in simulation which is approximately 5 meters. Within the field of view of the sensor, if there is an occupant, then the sensor report a "1", otherwise it report a "0". We assume that the sensors are aware of the locations and identity of themselves; therefore, the data collected from these sensors are labeled with the location and identity of the reporting sensor. The collected data are sent to a central computer for processing. The communication overhead during the transmission of the sensor data is assumed to be ignored. In other words, we assume that the data obtained from sensors are perfectly received by the central computer, knowing exactly which data are generated from which sensors. Furthermore, we model the error reported by sensors using a uniform distribution. The probability that a sensor produces an error reading (report "1" when there is no occupant in the detection area and report "0" when there is occupant in the detection area) is modeled as a variable P_{error} , where P_{error} is smaller or equal than 1 but larger than 0. At each time step, the simulation system evaluates whether there is at least one occupant in each sensor's detection area. The system then generates a random number between 0 and 1 following uniform distribution. If this number is smaller than P_{error}, then it means the sensor will produce an error reading, in which case the sensor produces "1" if there is no occupant in the sensor's detection area and produces "0" if there is at least one occupant in the sensor's detection area. If the number is larger than P_{error}, then it means the sensor will produce correct readings, in which case, the sensor produces "0" if there is no occupant in the sensor's detection area and produces "1" if there is at least one occupant in the sensor's detection area.

In the simulation, the sensors are deployed sparsely in the building. The distance between arbitrary two sensors are set to be larger than 2 times the diameter of the sensor. As a result, the sensors do not have overlapping areas with each other. Furthermore, one assumption is that there are areas that are not covered by any sensors' detection area. In this case, the agent's position becomes latent when it enters these areas and its position can no longer be measured by the sensors. This brings certain problem for estimation of occupants' positions. The problem brought by the latent state of the system can be defined as state estimation problem, that is, given incomplete observation and a model of the system, how to obtain the latent state of the system. In this dissertation, this problem is solved using particle filter.

4 A DATA ASSIMILATION FRAMEWORK BASED ON PARTICLE FILTERS

4.1 Concepts and methods of data assimilation

Data assimilation involves incorporating observations to estimate a system's state given some prior knowledge of the system represented by a model. The estimated state of the system can then be used as initial state of the subsequent simulations. These simulations produce forecast of the system state. Data assimilation is important for state estimation in a sense that the observations of the system are normally erroneous and incomplete; therefore, they cannot be used directly as the measurement of the system state. On the other hand, the model used to simulate the system is usually not perfect and using model along is not sufficient to predict the system state accurately. However, combining the incomplete observation data and imperfect model gives us a chance to utilize information provided by both of the two components and produce relatively accurate estimate of the system.

A typical data assimilation system is composed of three different components: the observations, the dynamic model to describe the system and an algorithm to meld the observations and the dynamic model. The observations arrive regularly or irregularly and can be sparse. It is sometimes indirectly related to the variable of the model. Consequently, the state estimation requires some background information derived from previous estimate of the system and it is better that this information can be accumulated in time and adopted by all of the variables of the model. The model component in a data assimilation system defines the physical constraints and prior knowledge about the underlying system to be estimated. The model used in data assimilation can be of different forms. In atmosphere and oceanology, the model is a set of mathematic equations to describe the fluid dynamic of the system; in social science, the model can be non-linear form such as agent-based model. The model defines how the system state evolves from previous time step to current time step and thus can serve to forecast the future state of the system. With the model of system, the incompleteness of the observation data can be overcome because the model defines how the system will evolve in the time interval where no observation arrives.

The schemes to combine observations and the dynamic model can be categorized into two different classes. If the estimation problem is to estimate the system state given historical observation up to current time, then the estimation problem is a filtering problem. If the estimation problem is to estimate the system state given future data, then the estimation problem is a smoothing problem. In this dissertation, we focus on the filtering problem. The smoothing problem is out of the scope of this dissertation. Among various melding scheme to combine observation and models, the most basic one is least squares estimation. Least squares estimation is built on top of a set of hypothesis, such as the observation function must be linear, background

error covariance matrix and observation error covariance matrix are positive definite matrices. It calculates a gain using observation matrix, covariance matrix of background error and covariance matrix of observation error. The gain is then used to calculate analysis error covariance matrix with observation matrix, covariance matrix of background error and covariance matrix of observation error. The analysis is equivalent to a problem that minimize the cost function calculated by summing background term and observation term. From least squares estimation, optimal interpolation methods can be derived. In optimal interpolation method, each observation is assigned with weight based on the statistical property of their errors. The goal of optimal interpolation method is to minimize the observation errors by determine the value of the gain. This is similar to the three dimensional variational assimilation and four dimensional variational assimilation method which aim at minimize a cost function and its gradient represented by observations weighted plus analysis and short term forecast by their accuracy. The difference between three dimensional variational assimilation and four dimensional variational assimilation technology is that in three dimensional variational assimilation method, only observation at one time is included in the cost function and in four dimensional variational assimilation method, observations distributed at multiple times of a time interval are included in the calculation of the cost function. Variational assimilation technologies primarily focus on exploring the initial value of the model, they do not aim at adjusting the current state of the system at analysis time. To adjust system state at analysis time, Bayesian filtering is a viable method. Bayesian filtering methods estimate a system's state from noisy observations. It is derived from Bayes theorem. Given two events H and T , according to Bayes rules , the following equation holds for any p(T) that is not zero :

$$P(H \mid T) = \frac{P(T \mid H)P(H)}{P(T)}$$

This can be applied in the recursive Bayesian filtering problem, given system evolution model for calculating the transition prior:

$$\mathrm{x}_k = g(\mathrm{x}_{k-1}, \mathrm{v}_k) \sim p(\mathrm{x}_k \mid \mathrm{x}_{k-1})$$

And observation model for calculating the likelihood:

$$\mathrm{y}_k = h(\mathrm{x}_k, o_k) \sim p(\mathrm{y}_k \mid \mathrm{x}_k)$$

Assuming that the system is Markovian, with initial distribution of the system state and historical observation vector we can estimate the posterior distribution of the system using Bayesian theorem:

$$p(x_k \mid y_{1:k}) = \frac{p(y_k \mid x_k)p(x_k \mid y_{1:k-1})}{p(y_k \mid y_{1:k-1})} \qquad (4.1)$$

, where a prediction can be calculated when an observation arrives using system transition prior and the Bayesian belief of previous time step:

$$p(x_k \mid y_{1:k-1}) = \int p(x_k \mid x_{k-1})p(x_{k-1} \mid y_{1:k-1})dx_{k-1} \qquad (4.2)$$

Then the prediction is updated using likelihood and the posterior or Bayesian belief of current time step can be calculated using the prediction and likelihood:

$$p(x_k \mid y_{1:k}) = a \cdot p(y_k \mid x_k)p(x_k \mid y_{1:k-1})$$

, where *a* is the normalization factor represented by:

$$p(y_k \mid y_{1:k-1}) = \int p(y_k \mid x_k)p(x_k \mid y_{1:k-1})dx_{k-1} \qquad (4.3)$$

The equation of (2) and (3) form the foundation of recursive Bayesian filtering. The propagation of posterior distribution is just a conceptual solution to the optimal Bayesian filtering problem. There is no analytic solution. However, there exists a set of method that can

approximate the solution. These methods include Kalman filter and particle filter. Kalman filter aims at estimating the state of a linear system that has following form:

$$x_k = Ax_{k-1} + v_{k-1}$$

with state transition matrix *A*, and observation function:

$$y_k = Hx_{k-1} + o_{k-1}$$

with observation matrix H, where system noise *v* and observation noise *o* follows Gaussian distribution,

$$p(v_{k-1}) \sim N(0,Q)$$
$$p(o_{k-1}) \sim N(0,R)$$

The Kalman filter has two steps. In the first step, state at previous time step and system transition function are used to calculate the a priori estimate error covariance matrix of the system state. This step is called prediction. It involves using the system transition matrix to calculate a prior state estimate from previous state and then derive priori estimate error covariance matrix from posterior error covariance matrix of previous timestep:

$$x_k^- = Ax_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

In the second step, the Kalman filter utilizes observation to update the predictions to obtain the posterior state estimation and a posterior error covariance matrix:

$$x_k = x_k^- + K_k(y_k - Hx_k^-)$$

$$P_k = (1 - K_k H)P_k^-$$

, where the Kalman gain K_k is calculated as:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

By recursively performing the prediction and update, the state of the system can be estimated. Kalman filter has been proved to be an optimal filter for linear and Gaussian system. However, it has several drawbacks. The major drawback is that it can only be applied to system whose transition function can be written as linear forms with system noise and observation noise to be Gaussian distributed. Furthermore, it requires that the target distribution to be unimodal. Consequently, Kalman filter is not appropriate in non-linear, non-Gaussian, multimodal state estimation problems. To overcome the limitation of the standard Kalman filter, researchers have proposed several variations of the standard Kalman filter. For non-linear state estimation problem, the extended Kalman filter has been proposed. Extended Kalman filter can deal with system whose state transition function is non-linear. The core concept of extended Kalman filter is to linearize the non-linear state transition function. This is done by calculating the Jacob matrix of partial derivatives of the non-linear state transition function and observation function, and use the result to construct the linear form of the state transition function and observation function. By doing this, the prediction step and update step of the standard Kalman filter can be applied to estimate the system state using the new constructed linear representation of the non-linear system functions. Extended Kalman filter is able to deal with non-linear system, however, it still requires that the system and observation noise to be Gaussian. Furthermore, when the system to be estimated is highly non-linear, extended Kalman filter performs poorly because the calculation of Jacobian matrix would be complex. Unscented Kalman filter is proposed to alleviate the deficiencies of extended Kalman filter. The state distribution in Unscented Kalman filter is specified by a minimal number of chosen samples called sigma points. These points are chosen to be around the mean and propagated using the nom-linear system transition function. The estimated mean and covariance can be then calculated from the non-linear system transition

function. The unscented Kalman filter can possibly estimate a system whose noise is not Gaussian distributed and often yields better performance than extended Kalman filter. However, when it comes to system with large amount of variables, the standard Kalman filter and its extended variations performs poorly because large amount of computations are required to deal with the high dimensionality. To alleviate this problem, researchers develop another Kalman filter based estimation technology called ensemble Kalman filter. The ensemble Kalman filter using a set of ensemble members to approximate the distribution of the state of the target system. Instead of calculating covariance matrix as in standard Kalman filter, the ensemble Kalman filter calculate the covariance of the ensemble members, and then use it to update the estimate of each ensemble members. The ensemble Kalman filter performed good in applications where large number of variables are involved. However, it assumes the system noise and observation noise to be Gaussian. Although Kalman filter and its variants are computational efficient, they normally require that the system must be written in analytic forms; furthermore, some of these methods requires that system to be linear (standard Kalman filter), some perform poorly when the system is strongly non-linear (extended Kalman filter, unscented Kalman filter), some requires that the system noise and observation noise follow Gaussian distribution (ensemble Kalman filter, extended Kalman filter, standard Kalman filter) and some cannot handle multimodal distribution. Therefore, these methods are not suitable for the people tracking problem defined in this dissertation. In this dissertation, we choose particle filter algorithm as the melding scheme to combine observation and the model to produce the estimate of the system. The details of the particle filter algorithm is presented in section 4.2.

4.2 Particle filter algorithm

Data assimilation involves incorporating observations into a computer model to obtain best estimate of the system state. Since we assume that sensors in the smart environment are deployed sparsely, once an occupant moves into a "blind" area that is not covered by the sensors, its position becomes latent and can only be inferred from the agent-based model. On the other hand, because occupants' movements in buildings are generally unpredictable, deriving occupants' position only from the agent based model without incorporating any observation data is infeasible. Data assimilation combines both information provided by observation data and the agent-based model to estimate occupants' positions. In this work, we carry out data assimilation using Particle Filters (PFs).

Formally, particle filters work with a state-space model that has a generic form containing a state transition function and a measurement function:

$$x_t \sim p(x_t \mid x_{t-1})$$
$$y_t \sim p(y_t \mid x_t)$$

,where x_t represents the system state at time t evolved from state at previous time step $t-1$; and y_t represents the observation at time t given system state at time t. In particle filters, posterior distribution of the system state is represented by a number of weighted particles. The weights of these particles are calculated based on the state transition, proposal distribution and likelihood of the particles. The particles are updated and reweighted at each estimation step. Formally, particle filter is derived from sequential importance sampling (SIS), where the posterior distribution is approximated by:

$$p(x_t \mid y_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta(x_t - x_t^{(i)})$$

,and w represents the weight and is updated using:

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{q(x_t^{(i)} \mid x_{x-1}^{(i)}, y_t)} \qquad (4.4)$$

However, sequential importance sampling suffers from the phenomena of degeneration, where weights of all but several particles become 0 after some iterations. To solve the problem, an additional re-sampling step is applied for eliminating particles with low weights and multiplying particles with high weights. Theoretically, by constructing an accurate and easy-to-draw proposal distribution and utilizing sufficient number of particles, the estimated probability density of system states will eventually converge to the posterior of the target system. In the related work proposed in this dissertation, we draw samples from the transition prior using the agent-based model. At each step, the samples are drawn from the proposal distribution $q(x_t^{(i)} \mid x_{x-1}^{(i)}, y_t)$. When this distribution is chosen to be the system transition prior $p(x_t^{(i)} \mid x_{t-1}^{(i)})$, equation 4.4 can be transformed to:

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{p(x_t^{(i)} \mid x_{t-1}^{(i)})} = w_{t-1}^{(i)} p(y_t \mid x_t^{(i)})$$

, which means the update of the weight depends only on the likelihood of the particles.

Overall, data assimilation using particle filters consists of three major steps. The first step is the sampling step, which involves drawing samples (particles) from proposal distribution; in our work, the proposal distribution is chosen to be the state transition function, i.e., the agent-based simulation model. More specifically, this is done by running the simulation to the next observation time according to the agent-based model. The second step is to calculate the likelihood, which involves weigh a particle based on the probability for the particle to generate measurement that matches the ground truth measurement; to do this we need to define the observation model and calculate the distance between the ground truth measurement and measurement of each particle. The third step is re-sampling, which involves multiplying particles

with higher weights and eliminating those with lower weights. In this research, besides the standard re-sampling method we also developed a new re-sampling method named as the component set re-sampling. This method effectively increases the diversity of the sample space and is capable of representing a larger sample space using limited number of particles. To apply particle filters for data assimilation, the state transition function that captures the state evolution and the observation model that maps system state to observation data need to be defined. Below we describe these components in detail.

4.3 State and state transition function

In this research, the state transition function is used in the particle filter algorithm to draw new samples in each data assimilation iteration. We define the state of the system as the positions and destinations of all the agents in the smart environment. Notice that since agents' velocities and intermediate destinations are calculated from agents' positions and destinations, we do not include them explicitly in the definition of the system state. Formally, the state of the system is defined as:

$$S_t = < A_t^1, A_t^2, A_t^3, A_t^4, ..., A_t^m >$$

, where m is the total number of agents and A_t^i is the state of agent *i* at time *t*,

$$A_t^i = < l_t^i, d_t^i >$$

, where l_t^i, d_t^i represent the current position and final destination of agent *i* at time *t*, respectively. The system transition function is then formulated as follows.

$$S_{t+1} = ABMTransition(S_t) + Q_t \qquad (4.5)$$

where Q_t is the processing noise and *ABMTransition* represents the state transition function. This function defines how the state of a particle evolves from time step *t* to time step *t+1* and is based on the agent-based model described in chapter 3. The position update of an

agent is based on its velocity, which is calculated from the agent's current location and its next intermediate destination and is affected by agent's avoidance behavior. Once the agent reaches its current final destination, a new destination is generated. In this work, to support data assimilation using particle filters, an agent's new destination is generated stochastically based on a distance that is measured by how many waypoints away from the agent's current destination in the waypoint graph. Specifically, we use the following method to generate a new destination once an agent reaches its present final destination:

1. Determine the heading direction, denoted by *a*, of the agent.
2. Generate a waypoint distance from the current destination. Let $lr' = |GP(lr, 1)|$ where *GP* is a Gaussian process with mean to be *lr*, which is a constant and represents the average waypoint distance from the current destination, and variance to be 1. *lr'* is the number of way points that the new destination will be set away from the current destination.
3. $d=a$, $g=A$ where *a* is the heading direction of the agent and *A* is the position of the agent.

   For i=0 to lr'

   Randomly select a direction *c* based on direction *d*, this is done by connecting position *g* to all of its nearest waypoints in the waypoint graph and then select a direction along the edges generated by these connections. The probability of selecting direction for opposite direction of *d* is set to be a constant p_b and the probability for the agent to select other direction is set to be $1-p_b$. Find the closest waypoint *w* near position *g* on direction *c*.

   $d=c$, $g=w$

   End For
4. Randomly pick a position between g and its nearest way point. This position is considered as the new destination that the agent will move to.

4.4 Observation Model

The observation data used to calculate importance weights of particles is collected from a set of binary proximity sensors. The sensor produces 1 if there is at least one occupant in its detection area and produce 0 otherwise. The error rate of the sensor is set to P_{error} so that the probability for a sensor to report incorrect reading at a particular time step is P_{error}. Assuming there are *m* sensors deployed in the smart environment, the output of the measurement is then a vector containing *m 0s* or *1s*.Formally, given a set of occupants:

$$O = \{o_1, o_2, o_3, o_4, o_5 \ldots o_n\}$$

, a set of sensors, SE, that are deployed sparsely in the building:

$$\text{SE} = < se_1, se_2, se_3, se_4, se_5, ..., se_m >$$

For each sensor se_i, its location is defined as l^s_i and its detection area is specified by a distance rs_i , which is the radius of the area it is able to observe. The observation generated by se_i at time *k* is then defined as follows:

$$se_i = \text{Observ}(O, l^s_i) + m_{i,k}$$

, where

$$Observ(O, l^s_i) = 1 \quad if\ \exists o_j \in O, | l_{oj} - l^s_i | < rs_i$$
$$Observ(O, l^s_i) = 0 \quad Otherwise$$

In this research, since the transition prior $p(x_t|x_{t-1})$ is chosen to be the proposal distribution, the weight of the particle is calculated based on the likelihood $p(y_t|x_t)$. Since the measurement is a vector, like in many Kalman Filter applications, we define that the likelihood as a multivariate Gaussian distribution. In our work, the sensors are independent with each other. Thus the importance weight (likelihood) $w_k^{particle}$ of a particle at time *k* can be calculated from

the multiplication of the likelihood of individual sensors. Specifically, $w_k^{particle}$ is calculated as follows:

$$w_k^{particle} = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}} e^{-\frac{d_i^2}{2}}$$

, where d_i is the distance between the measurement of the particle and ground truth measurement for sensor i. We define that d_i is calculated as follows:

$$\begin{cases} d_i = 0 & se_i^{real} = 1, se_i^{particle} = 1 \\ d_i = 1/a & se_i^{real} = 0, se_i^{particle} = 0 \\ d_i = a & else \end{cases}$$

In our work, a has value of 2, thus $a > 1/a > 0$. For each sensor, we assign different values to d_i in different situations to indicate the different information it carries in computing the importance weight of a particle: 1) when the measurement of a real sensor and the measurement of its corresponding sensor in a particle both have value 1, this means the two observations match and carry strong information (the sensor detects occupant movement within its detection range), and thus d_i has the a value of 0 to indicate the distance between measurement of this sensor and the ground truth measurement is smallest; 2) when both have value 0, this means the two match but carry less information (the sensor does not detect occupants within its detection range), and thus d_i has a larger value but is still smaller than 1; 3) when the two mismatch, d_i has a value greater than 1. The importance weight represents likelihood of state of a given particle with respect to the observations obtained from the target system.

4.5 Standard Resampling

The particle filtering algorithm follows the standard sequential importance sampling with resampling (SISR) procedure [25]. It first initializes the particles so that they are randomly distributed over the state space. After that, samples are drawn by advancing the state of each

particle using the state transition function described in section 3.2 for a period of time Δt (Δt depends on how often the observation data is collected and assimilated). The weight of each particle is calculated based on the comparison between the observations generated by the sensor model for each particle and the real observation data collected from the smart environment. The weights are then normalized so that in the re-sampling step, particles with higher weights have greater chances to be selected and reproduced.

Re-sampling in particle filters deals with the degeneration problem where all but several particles' weight decrease to 0 after several iterations.[25] It avoids the problem by eliminating particles with lower likelihood and multiplying those with higher likelihood at each estimation step. It contains two steps: first, *normalization* so that the sum of weights of all particles is equal to 1; second, *selection* so that a particle's normalized weight becomes its probability of being selected during re-sampling. The new set of particles is generated by performing selection step N times (where N equals to the total number of the particles). The procedure of re-sampling is summarized as follows:

Standard Re-sampling

Re-sampling step:

Input: The set of particles $\{\tilde{x}_{1:t}^{(i)}; \mathrm{i} = 1,..., \mathrm{N}\}$ with associated weights $\{\tilde{w}_t^{(i)}; \mathrm{i} = 1,..., \mathrm{N}\}$;

Output: A new set of particles $\{x_{1:t}^{(i)}; \mathrm{i} = 1,..., \mathrm{N}\}$

1. Normalize the weights according to:

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}}$$

2. **For** $k = 1$ to N

   Select particle $x_{1:t}^{(k)}$ from the set :

   $\tilde{X} = \{\tilde{x}_{1:t}^{(i)}; i = 1,...,N\}$ according to $W = \{w_t^{(i)}; i = 1,...,N\}$

   End for

Figure 4.1 Algorithm for standard resampling

Although avoiding the degeneration problem, the standard re-sampling method brings two potential issues: sample impoverishment and particle deprivation. The sample

impoverishment phenomena occurs when after several iterations, all the particles are generated by multiplying a small number of particles (because these particles have dominant weight or because other particles have a much lower likelihood.). As a result, the diversity of the particles decreases after re-sampling and particles tend to collapse to a single point or several identical points on the state space after some steps. At this stage, due to loss of diversity, if the states of particles are all far away from the true state of system, there is no way for particles to rewind and recover. The other issue of particle deprivation occurs when there is no particle in the vicinity of true state; as a result, it is impossible for the algorithm to converge to posterior. Particle deprivation is caused by many reasons. One of the reasons is loss of diversity of particles introduced by sample impoverishment. Another reason is combinatorial explosion of possible states due to high dimensional state space, which means that the number of possible combinations of state values increases exponentially as the dimension of system state increases. For high dimensional state space, a large number of particles are required in order to cover all possible combinations of state values to prevent deprivation, thereby significantly increasing the computation cost. In this work, we aim to estimate occupants' position on a 2D space. Since each agent has two state variables, the number of dimensions of the entire state space is 2**n* where *n* is the total number of agents. As the number of agents increases, the high dimensional state space of the system leads to deprivation caused by combinatorial explosion and loss of diversity. To alleviate this problem, we propose a new re-sampling method named component set re-sampling as described in Chapter 5.

4.6 System noise

To improve the quality of data assimilation, how to model the process noise Q_t in equation (2) is also important. In this paper, we model the process noise from two aspects: the

noise added to the position of the agents and the noises added to the destinations of the agents. For agents' positions, we simply add a random position noise in uniform distribution. We also add noise to agents' destinations so that particles re-sampled from the same parent can have different destinations. The method for adding noise to agents' destinations is similar to the method for generating new destinations described in section 4.3 except that *A* is initialized with the destination instead of the position of the agent and *d* is initialized with the heading direction of the destination (directing from the second last intermediate destination to the final destination). Fig 2 shows an example of adding noise to the destination:

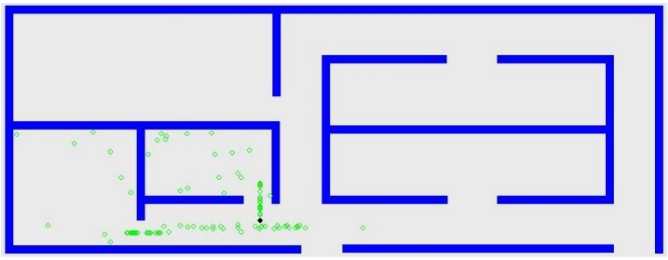


Figure 4.2 An example of adding noise to destination

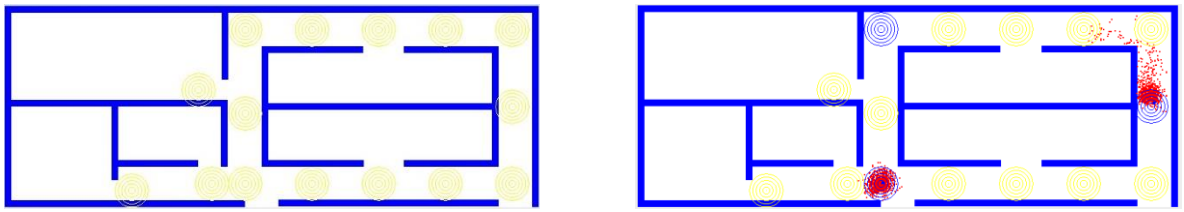
In Fig 4.2, the black circle represents the original destination and the green ones are different instances of destinations generated by adding noise. Totally 100 instances of new destinations (green circles) are generated by randomly adding noise to the original destination (black circle) using the method described above. In this example, *lr* is set to 1; the heading direction of the original destination is 180 degree (from right to left); the probability for generating a new destination on the opposite direction is 30%. As can be seen, the new destinations are generated around the original destination, with most of them are in the forward areas along the heading direction of the original destination, and the majority of them are within 1 waypoint distance from the original destination.

4.7 Impact of number of particles

The performance of particle filter algorithm relies heavily on the number of particles utilized. With more particle, larger area of the state space can be covered and there is a higher chance that there are particles in the vicinity of the true state. To validate this we have conducted a series of experiments.

We use the identical twin experiment to evaluate our method, which is commonly used in data assimilation research. In the identical twin experiment, we first run a simulation and record the corresponding data. These simulation results are considered as "real", therefore, the observation data obtained here are regarded as coming from the "real" system. Consequently, we estimate the system state from the observation data using particle filters and then check whether these estimated results are close to the "real" results. In this dissertation we do not assimilate real world sensor data, which is considered as future work.

In all the experiments, the floor structure and sensor deployment are shown in Fig 4.3(a), where the space is a 800*300 2D space; the blue lines represent the walls and yellow circles represent the sensors' location and detecting areas. The error rate of the sensors is set to 5%, which means the sensor has 5% probability to report 1 when it is supposed to report 0 and vice versa. At the initialization stage of the particle filtering algorithm, agents' positions and destinations are distributed uniformly on the space (except the locations occupied by walls). Fig 4.3(b) shows an illustration of data assimilation results for tracking two agents, where the blue dots are the location of the two real agents, and the red dots are estimated locations of the two agents from all particles. As can be seen, the estimations from the particles converge to the "true" state of the real agents. Note that in this research the posterior is represented by the aggregation of all of the particles instead of a single particle with highest weight.



(a) Floor structure of the smart environment (b) Result illustration (with 2 agents)

Figure 4.3 Experiment setup

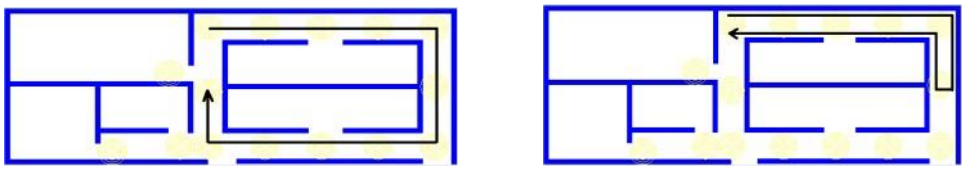
We first carry out experiment to evaluate the robustness of the particle filter-based data assimilation by assimilating sensor data to estimate the position of a single moving agent. The goal is to show that the state estimation converges to the "true" value, that is, the error between the true position and the estimated position of the agent maintains below a certain threshold, as the data assimilation proceeds. To evaluate the performance of the data assimilation, in each step of the data assimilation we calculate the error between the agents' real positions and estimated positions from particles, and then use the averaged position error for all particles to measure the estimation accuracy. Formally, the averaged position error at time t is calculated as below:

$$E(\mathrm{t}) = \frac{\sum_{k=1}^{N_p} | \mathrm{l}_t^k - \mathrm{l}_t^{real} |}{N_p}$$

, where N_p is the total number of particles, l_t^k represents the location of the agent in particle k at time step t and l_t^{real} represents the location of the agent in the real system at time step t. In this experiment, we set N_p to 800 so that 800 particles are utilized for the data assimilation. The total simulation time is 800 steps. The sensor data are assimilated every 4 steps, thus the total number of steps of data assimilation is 200.

In this experiment, we evaluate the data assimilation for two cases where the agent has different moving patterns. In the first case, the real agent is designed to move forward through a sequence of destinations. The route configuration is shown in Figure 4.4 (a). In the second case,

the real agent is designed to move straight forward through a sequence of destinations, then turn back to move to its initial position. The route configuration for the second case is shown in Figure 4.4 (b). The two cases share the same initial position for the agents and the same sensor error rate. Fig 4.5 shows the estimation errors of the two experiments. Result at each data assimilation step is calculated by averaging results from 10 experiment runs.



(a) Regular route (b) Turning back route

Figure 4.4 The configuration of the routes

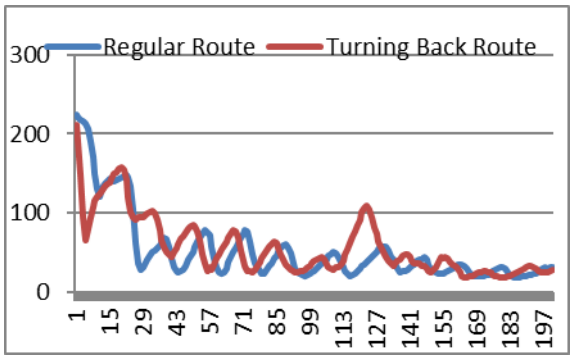


Figure 4.5 Estimation error for the regular route and turning back route

In Figure 4.5, the vertical axis represents the error and the horizontal axis represents the data assimilation steps. In both cases, there are oscillations for the estimation errors. This is because the error increases when the agent enters areas where no sensors are deployed and decreases when the agent enters the sensors' detection areas. As a general trend, as time increases the estimation error decreases and eventually maintains at a low level. This is expected and shows that the particle filter algorithm is working. Fig 8 also shows that in the turning back route configuration, the error increases significantly between estimation step 113 and 127. This is because the agent is turning to an opposite direction and is staying in an area where no sensors

are deployed. After the agent enters the sensor detection area, the error decreases rapidly and maintains at a low level similar to that in the regular route configuration. This series of experiments show that the standard re-sampling algorithm is robust in estimating the position of a single agent.

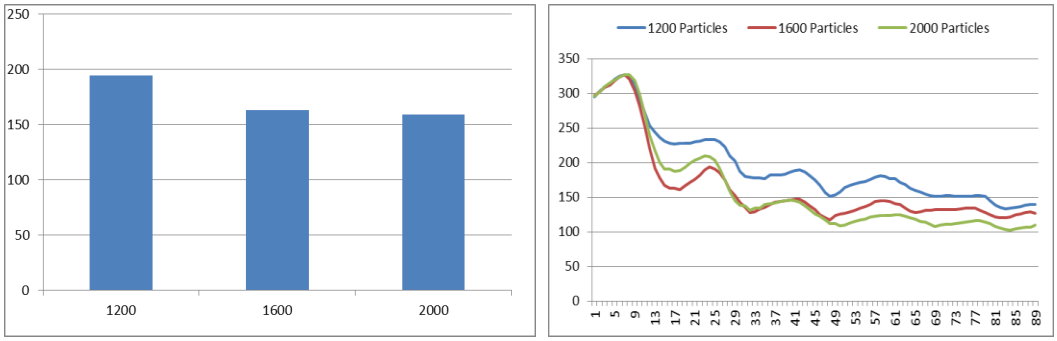
We then conduct a series of experiments to compare the effectiveness of data assimilation for two agents using different number of particles. The purpose of these experiments is to investigate the impact of number of particles on estimation performance using standard re-sampling. In general, we expect to see improved data assimilation result when the number of particles increases. In other words, the estimation error decreases as the number of particles increases. The estimation error is calculated as follows:

$$E(\mathrm{t}) = \frac{\sum_{k=1}^{N_p} \frac{1}{P} \sum_{j=1}^{P} | l_t^{(j)k} - l_t^{(j)real} |}{N_p}$$

, where N_p represents the number of particles and P represents the number of agents in the experiments; $l_t^{(j)k}$ represents the position of agent j in particle k at time t and $l_t^{(j)real}$ represents the position of the pairing agent in the real system at time t , we apply a matching procedure to match an agent in a particle and an agent in the real system uniquely so that the distance between them represent the true error between a particle and real state. The pairing procedure is needed because sensors carry no identity information thus we need a way to associate an agent in the particle with the corresponding agent in the real system for comparison purpose. The paring procedure works in the following way. Given a particle with agents $<a_1\ a_2\ \ldots\ a_n>$ and the real system with agents$<\ b_1\ \ b_2\ \ldots b_n>$, we first calculate the distances between all possible pairs $<a_i,b_j>$ for all i = 1 to n and j=1 to n. We then sort the pairs $<a_i,b_j>$ in ascending order based on the distance between a_i and b_j and store the sorted pairs in a list L. After that, we recursively

select the first element $<a_x,b_y>$ in L (which is a pairing of a and b that has the smallest distance) , store $<a_x,b_y>$ in a list T (referred to as the *pairing result list*), and then delete all of the pairs in the list L that contains a_x or b_y. This continues until L is empty. The paring result is stored in list T, where each element of T is a unique pairing of elements in $<a_1\ a_2\ \dots\ a_n>$ and elements in $<b_1\ b_2\ \dots b_n>$.

The sensor data is assimilated every 4 steps, which means there are approximately 90 data assimilation steps. The number of particles N_p varies from 1200, to 1600 and then to 2000. The number of agents P is 2. The result at each data assimilation step is calculated by averaging results from 20 experiment runs. The result is shown in Figure 4.6.



(a) Average error (b) Error per data assimilation step

Figure 4.6 Error using different number of particles

In Figure 4.6(a), the vertical axis represents the estimation error by averaging the error for all data assimilation steps. The horizontal axis represents the number of particles used in data assimilation. It shows that with increasing number of particles, the overall estimation error decreases. Figure 4.6(b) shows the estimation errors over time for the three cases. In general, it can be seen that as the number of particles increases, the error per time step decreases. The results presented in Fig 9 show that the estimation accuracy is influenced by the number of particles used in data assimilation. This is expected, because when the number of particles

increases, larger area of state space can be covered by the sample set and it is more likely that there exist samples whose states are close to the “real” state.

5 COMPONENT SET RESAMPLING FOR HIGH DIMENSIONAL STATE SPACE

5.1 Problem of particle filter

As mentioned in previous chapters, particle filter has some problems that limit its performance. The two major problems are sample impoverishment and particle deprivation problems. When the state of the target system contains multiple dimensions, these two problems are amplified because the number of particles required for convergence grows exponentially with respect to the number of dimensions of the system. In this dissertation, we aim at estimating a system that contains multiple agents, therefore, the target system can be considered as a system with multi-dimensional state space, where each agent represent one dimension of the system. Consequently, we develop a new resampling method to deal with the sample impoverishment problem and particle deprivation problem introduced by the high dimensionality of the system. This method effectively increase the ability of particle filter algorithm to represent large state space using limited number particles. In this chapter, we introduce this approach and analysis its impact on the estimation performance. Before we introduce the new method, we first give an overview about the two problems of particle filter algorithm.

5.1.1 Sample impoverishment

In the particle filter algorithm, the re-sampling step are introduced to deal with the degeneration phenomena of sequential importance sampling procedure. Degeneration happens during sequential importance sampling procedure, when after several iterations, all but one particles have neglected weights. Re-sampling solve this problem by eliminating particles with

lower weights and multiplying particles with higher weights so that the variance of the weight is reduced. However, re-sampling brought new a problem, that is, re-sampling always eliminates some particles. As a result, the diversity of the particles tend to decrease. After running sufficient long, there is a high probability that all particles become identical. This is called sample impoverishment problem. The result of sample impoverishment is that the particles will tend to collapse to several points and the posterior distribution of samples will be represented by a few distinct samples. To solve this problem, we need find a way to increase the diversity of the samples during estimation.

5.1.2 Particle deprivation

Particle deprivation problem happens when there are no particles are in the vicinity of the true state. Consequently, it is impossible for particle filter algorithm to converge to the true posterior distribution of the target system. Particle deprivation problem can happen due to many reasons. One reason is the loss of diversity. Since the variance of particles tend to decrease over time, if the correct state is not covered by the particles initially, there is no way for the particle filter to rewind and recover. Another reason comes from the combinatorial explosion of the value of the states of the particles. As mentioned previously, the number of particles required to cover the state space grows exponentially with respect to the number of the dimensions of the system. It requires large number of particles to cover all possible combinations of values of different dimensions of the state space. If a limited number of particles are utilized, it is more likely that the correct combination of values of different state dimensions will not be covered by the set of particles and thus result in particle deprivation. This problem can be solved by increase the number of possible combinations of different dimensions so as to enlarge the coverage of particles over the state space.

5.2 Component set re-sampling

To alleviate the problem brought by the sample impoverishment and particle deprivation problem we develop a new re-sampling called component set re-sampling. In this section we describe the method developed in detail.

5.2.1 Motivation

In the standard particle filter re-sampling procedure, the entire system state is treated as a whole when being re-sampled. The basic idea of the component set re-sampling is to break the system state into sub-components and carry out re-sampling at the component level. In other words, instead of re-sampling each particle as a whole as in standard particle filters, we divide the state space into sub dimensions (named as components) and resample from the set of components to "construct" new particles. The component set re-sampling makes sense especially for the agent-based simulation model because the system state is composed from states of individual agents, and each agent's state variables can be treated as a component of the whole system state. For example, for an agent-based model with two agents *agent1* and *agent2*, one component would be *agent1*'s position and destination; another component would be *agent2*'s position and destination. With the component set re-sampling, the combinations among agents in each particle are broken when being re-sampled. By reconstructing new particles with components from different particles, the diversity of the sample space increases with the same number of particles, because new combinations of components can be generated. We note that a particle filtering algorithm with the component set re-sampling differs from the standard particle filtering algorithm only in how particles are re-sampled. It does not change the weight calculation of particles, that is, a particle is still treated as a whole when calculating its importance weight (same as in the standard weight calculating procedure). After weight

calculation and normalization, we divide a particle into different components, all of which inherit the same weight from the particle. For a particular component category (e.g., agent1' state), the corresponding components from all particles form a component set. When reconstructing a new particle, we select a component (according to components' weight) from each component set for all component sets to form a new particle.

5.2.2 Method illustration

To illustrate the component set re-sampling method, formally, let:

$$x^i = < r_1^i, r_2^i, r_3^i, r_4^i, ..., r_m^i >$$

represents the system state of particle *i*, where *m* is the number of components in the system state (in our work, *m* is the number of agents in the system), and

$$r_j^i = < l_j^i, d_j^i >$$

represents agent *j*'s state in particle *i*, i.e., agent *j*'s position and destination. Overall, the states of particle *i* consists of components $r_1, .r_2 , r_3, \ldots, r_m$. Each component r_j^i has the same important weight that is inherited from the particle's importance weight.

$$w_t^{r_j^i} = w_t^i = p(y_t^i \mid x_t^i)$$

The component sets of N particles are defined as:

$$R^1 = \{ r_1^1, r_1^2, r_1^3, r_1^4, ..., r_1^N \}$$

$$R^2 = \{ r_2^1, r_2^2, r_2^3, r_2^4, ..., r_2^N \}$$

$$R^3 = \{ r_3^1, r_3^2, r_3^3, r_3^4, ..., r_3^N \}$$

$$\ldots \ldots$$

$$R^m = \{ r_m^1, r_m^2, r_m^3, r_m^4, ..., r_m^N \}$$

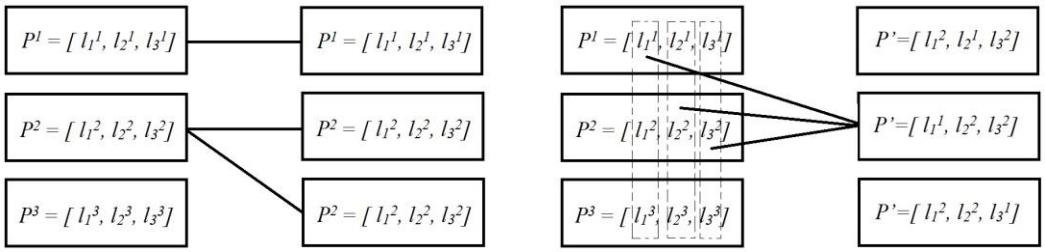
, where each r_j^i has an associated $w_t^{r_j^i}$. With these components sets, a new particle is constructed as:

$$\tilde{x}_t^i = < \tilde{r}_1^i, \tilde{r}_2^i, \tilde{r}_3^i, \tilde{r}_4^i, ..., \tilde{r}_m^i >$$

, where

$$\tilde{r}_k^i = select(R^k)$$

and *select (X)* is the function of multinomial selection to select a component from set *X* according to the weight assigned to the components. Fig 5.1 illustrates how the component set re-sampling works. In this example, there are 3 particles (P^1, P^2 and P^3) where each particle is composed from 3 components l_1, l_2, and l_3 (in the figure, $l_i{}^j$ represents the component l_i in particle P^j). Assuming the weights of the three particles P^1, P^2 and P^3 are WP_1, WP_2, WP_3 respectively, we have $WP_2 > WP_1$, and $WP_1 = WP_3$. In (a), the standard re-sampling is used where the 3 particles are selected according to their weights. In this example, P^1 is selected once and P^2 is selected twice. In (b), the component set re-sampling is used where re-sampling is carried out at the component level. The three vertical dashed boxes represent the component sets for l_1, l_2, and l_3, respectively. As can be seen, each component set is re-sampled vertically and then recombined together to form a new particle. In this example, since P^2 has the highest weight, the components from P^2 yield higher probability to be selected.



(a) Standard Re-sampling (b) Component set Re-sampling

Figure 5.1 Comparison of standard re-sampling and component set re-sampling

The component set re-sampling procedure is summarized as follows:

Component Set Re-sampling

Re-sampling step:

Input: The set of particles $\{\tilde{x}_{1:t}^{(i)}; \mathrm{i} = 1,...,\mathrm{N}\}$ with associated weights $\{\tilde{w}_t^{(i)}; \mathrm{i} = 1,...,\mathrm{N}\}$;

Output: A new set of particles $\{x_{1:t}^{(i)}; \mathrm{i} = 1,...,\mathrm{N}\}$

1. Normalize the weights according to:

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}}$$

2. **For** i= 1 to N

Divide state space into m components

$x^i = < r_1^i, r_2^i, r_3^i, r_4^i, ..., r_m^i >$;

Assign weight $w_t^{r_j^i} = w_t^i = p(y_t^i \mid x_t^i)$ to each component $\mathrm{r_j^i}$;

End for

For each component r_k^i , k=1 to m

create a component set R^k so that:

$R^k = \{ r_k^1, r_k^2, r_k^3, r_k^4, ..., r_k^N \}$

End For

For i=1 to N

For q= 1 to m

select a component from $\mathrm{R^q}$ according to components' weights

$\tilde{r}_q^i \sim \{R^q; i = 1,...,N\}$ for particle $\tilde{\mathrm{x}}_t^i$;

End for

construct the new particle $\tilde{x}_t^i = < \tilde{r}_1^i, \tilde{r}_2^i, \tilde{r}_3^i, \tilde{r}_4^i, ..., \tilde{r}_m^i >$

End for

Figure 5.2 Algorithm of component set re-sampling

The component set re-sampling diversifies the sample space with a limited number of particles. However, it also brings an issue: when dividing the state space into components, a "good" particle with the right combination of components may be broken during the re-sampling procedure. Furthermore, when different components of the system state have dependencies over each other, a newly constructed particle may not hold the dependencies among the sub-components and thus results in "invalid" state from the system point of view. To ensure the robustness and correctness of the particle filtering algorithm, we propose to combine the component-set re-sampling and the standard re-sampling to take advantage of both. Specifically, during the re-sampling procedure, we re-sample one portion of the total particles using the

standard re-sampling and re-sample the rest of particles using the component set re-sampling. The goal of the standard re-sampling is to ensure the robustness and correctness of the algorithm, and the goal of the component set re-sampling is to diversify the sample space with limited number of particles. In our work, we have set that 20% of the particles are re-sampled from the standard re-sampling and 80% of the particles are re-sampled from the component set re-sampling. We note that these percentage numbers are empirically determined based on experiments for the agent-based model. In our application, the dependencies among the different agents are relatively weak and thus the majority particles are re-sampled from the component set re-sampling. We expect for different applications different percentage values should be used in order to achieve best performance. A guideline about what percentage values to use can be established based on the level of dependency among sub-components, and that is considered as future work.

The final re-sampling procedure is named as the *mixed component set re-sampling* and is described below, where we assume the total number of particles is *M*, among which *U* particles are re-sampled from the standard re-sampling and *V* particles are re-sampled from the component set re-sampling. We note that the standard re-sampling procedure can be thought as a special case of the mixed component set re-sampling when *U=M*.

Mixed Component Set Re-sampling

Resample particle set with M=U+V particles

1. **For** particle $x_t^1 \ldots x_t^V$
   Generate particles using **Standard Re-sampling**
   End for
2. **For** particle $x_t^{V+1} \ldots x_t^M$
   Generate particles using **Component Set Re-sampling**
   End for

Figure 5.3 Mixed component set re-sampling

With the component set re-sampling enabled, the overall data assimilation procedure for agent-based simulation of smart environment is performed as follows. First, we initialize the

particles where each particle has the same number of agents as the number of occupants in the real system. Initially, the position of each agent in each particle is randomly distributed over the 2D smart environment. In each data assimilation step, it first uses the state transition function defined in section 3.2 to draw new samples by forwarding the state of each particle for a period of time Δt, where Δt is the time from the last data assimilation step to the current data assimilation step; the particles are then re-sampled using the mixed component set re-sampling. This algorithm is summarized as follows:

Data Assimilation based on Particle Filter

1. *Initialization*. $t = 0$
   For particle $i=1, \ldots, N$,
   randomly assigning agents' positions and destinations in particle i;
   Generate routes
   Set t=1
   End For
2. *Sampling*
   For particle $i = 1, \ldots, N$
   Run Δt steps of simulation based on ABM model
   End For
   For particle $i=1, \ldots, N$
   calculate weights using (3)(4)
   End For
3. *Re-sampling*
   Resample with replacement N particles using the **mixed component set re-sampling**
   Set $t \leftarrow t+1$ and go to step. 2

Figure 5.4 Particle filter algorithm with component set re-sampling enabled

5.3 Impact of number of dimensions

While dealing with high dimensional systems the number of the dimensions have impact on the estimation accuracy of the particle filter algorithm. In this section, we carried out a series of experiments to investigate the impact of number of dimensions of system states on the estimation performance. This series of experiment explore the impact of the number of state space dimension (i.e., the number of components) of the system state on the standard re-sampling and to justify and demonstrate the need of component set re-sampling. As the number

of components (which equals to the number of agents in our work) increases, the number of possible combinations of component values increases too. Consequently, the number of particles required to cover all possible combinations increases. In this experiment, we refer to a particle whose component values completely match all the component values of the real state as a particle with correct combination. As the number of agents increases, we expect to see the number of particles with correct combinations decreases dramatically when using standard re-sampling. However, even though a particle may not have the correct combination for all component values, it can have some component values that match those in the real state. This means from the whole system state point of view, the particle is not "good" because it does not completely match the real state. When using the standard re-sampling, such a particle is likely to be eliminated entirely because of its low weight. However, the particle may still have "good" components that carry useful information and can be utilized, which gives rise to the component set re-sampling. This experiment aims to demonstrate this aspect.

We use two different methods to measure the degree of match between a particle's state and the real system state. In our work, the system state includes agents' positions and destinations. For simplicity, we only consider agents' positions when measuring the degree of match. In both methods, we first apply a paring procedure to pair the agents in a particle and the agents in the real system based on the distance between them so that the degree of match can be measured in a systematical way. The paring procedure is as same as the paring procedure demonstrate in section 4.7. After the pairing procedure, for each pair of agents we calculate the position difference (i.e., the distance) between the two agents. If the distance between the two agents is small (smaller than a threshold; in our experiment, since the diameter of sensor's detection area is 50, we set this threshold to be 50), we consider it as a *correct match*. We name

the first method of measuring the degree of match as the "complete match" method. In this method, we count how many particles whose states completely match the real state (which means all the pairs in *T* have the correct match). Formally, let <*a, b*> represents a pair in a pairing result list *T*, l_a and l_b represent the location of agent *a* and agent *b* for the pair <*a, b*>, N_p is the number of particles, T^i is the paring result list between particle *i* and the real state. Then the measurement of complete match is calculated as follows:

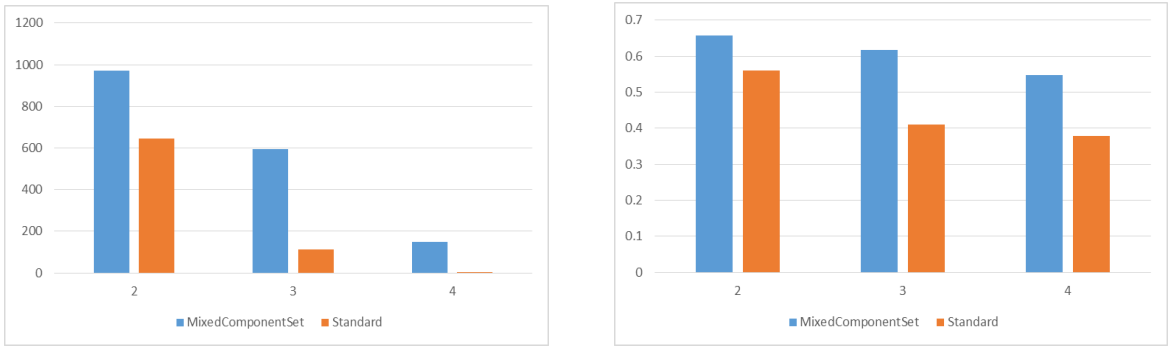
$$E(t) = \sum_{i \hat{I} N_p} g(T^i) \quad where$$
$$g(T) = 1 \; if \; '' < a,b > \hat{I} \; T, |l_a - l_b| < 50$$
$$g(T) = 0 \; otherwise$$

We name the second method of measuring the degree of match as the "percentage match" method. In this method, we calculate the ratio of "good" components in the overall system state for each particle (which means the ratio of pairs in T that has the correct match), and then compute the average for all particles. Formally, let <*a, b*> represents a pair in a pairing result list *T*, l_a and l_b represent the location of agent a and agent b for the pair <*a, b*>, N_p is the number of particle, *P* is the number of agents, T^i is the paring result list between particle *i* and the real state. Then the measurement of percentage match is calculated as follows:

$$E(t) = \frac{\sum_{i \hat{I} N_p} (\sum_{<a,b> \hat{I} T^i} h(a,b)) / P}{N_p} \quad where$$
$$h(a,b) = 1 \; if \; |l_a - l_b| < 50$$
$$h(a,b) = 0 \; otherwise$$

The two methods measure the degree that the particles match the real state at different granularity levels. The complete match measurement represents the match at the whole state level – a particle is a complete match only when its whole state matches the real state. The percentage match measurement represents the match at the component (sub-state) level – a particle contributes to the percentage as long it has one or more "good" components. To carry

out this experiment, we utilize 2000 particles and the total simulation time is 350. The data is assimilated every 4 time steps, thus the total data assimilation steps is 90. We vary the number of agents from 2, to 3 and then to 4, and use both standard re-sampling and component set re-sampling in all cases. The measurement results using the two measurement methods are presented in Figure 5.5:



(a) The result for complete match (b) The result for percentage match

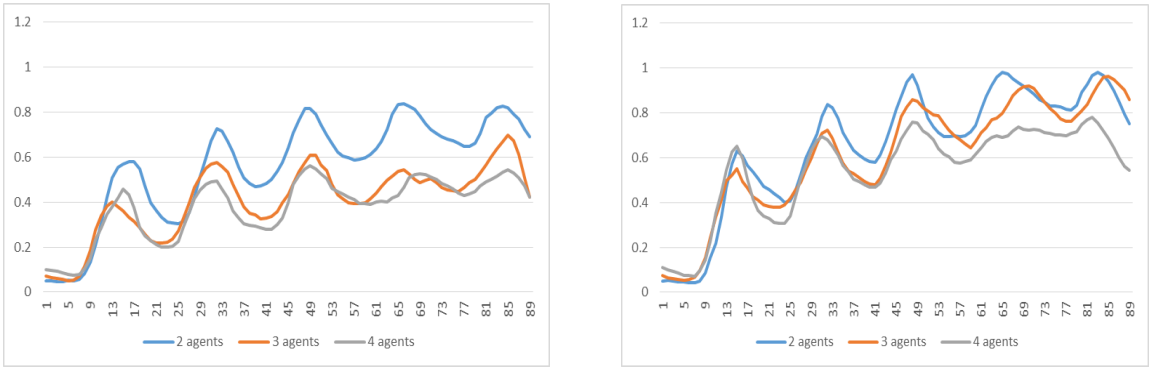
Figure 5.5 Experiment result averaged from each data assimilation step

Figure 5.5(a) shows the result using complete match measurement, where the y axis represents the average number of particles that completely match the real state and the x axis shows the three different cases with different number of agents. Figure 5.5(b) shows the result using percentage match measurement, where the y axis represents the average ratio of "good" components and the x axis represents the three cases. Both results are calculated by averaging results from 5 experiments. For each of these experiments, the final result is calculated by averaging the result at each data assimilation step. Figure 5.5(a) shows that for both standard re-sampling and mixed component set re-sampling, the number of complete match decreases as the number of agents increases. This is expected because as the number of agents increases, it is more difficult for particles to have correct combinations of system state. Figure 5.5(a) also shows that for standard re-sampling the number of complete match decreases dramatically as the

number of agents increases. When there are 4 agents, there is also zero number of complete match for the standard re-sampling. However, Figure 5.5(b) shows that even there are few complete match in the case of 4 agents, the standard re-sampling still have a relative high ratio of "good" components (about 37%). The existence of such "good" components is one of the major motivations for us to develop the component set re-sampling, which divides the whole state space into sub-components and reconstruct particles from component set. By doing this, the good "components" of all particles can be utilized for constructing new particles, and thus lead to improved results. The effect of mixed component set re-sampling is demonstrated in both Figure 5.5 (a) and Figure 5.5 (b). In Figure 5.5 (a), it is shown that the mixed component set re-sampling leads to more complete match in all three cases of 2, 3 and 4 agents. Even in the 4-agents case, the number of complete match reaches as high as 160 using the mixed component set re-sampling (compared to almost 0 for the standard re-sampling). In Figure 5.5 (b), the ratio of correct match of components increases when the mixed component set re-sampling is used. All these cases justify and demonstrate the need of the component set re-sampling.

Figure 5.6 provides more details about the percentage match for both standard re-sampling and mixed component set re-sampling by showing the value of percentage match over time. As can be seen, in both Figure 5.6(a) and Figure 5.6(b), the percentages of matching components increases over time. This indicates as a general trend the particles gradually converge to the posterior of the true state over time. In both two figures, there are oscillations due to the fact that sensors are sparsely deployed: the peaks are situations when the real agents move into the sensor detection areas; the valleys are situations when the real agents move out of the sensor detection areas. Furthermore, it shows that as the number of agents increases, the percentage of matching components decreases. This is consistent with Figure 5.5. Comparing

Figure 5.6(a) and Fig 5.6(b), we can see that the mixed component set re-sampling has better performance than the standard re-sampling, because the curves of the mixed component set re-sampling in all three cases give high values than those of the standard re-sampling.

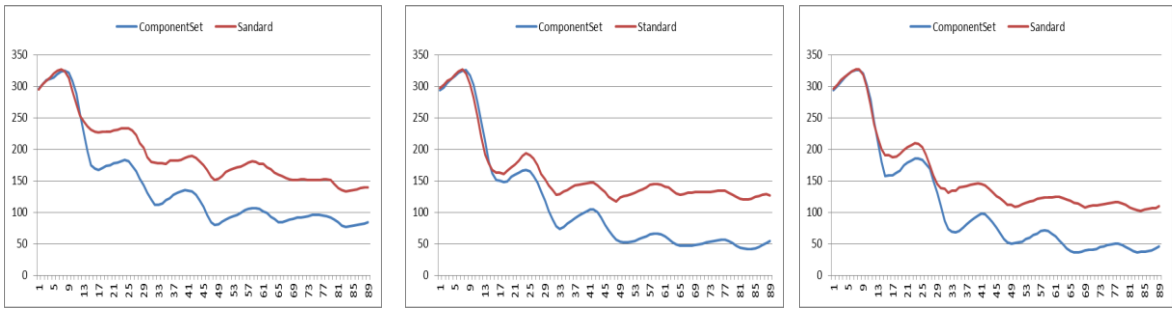


(a) Standard Re-sampling (b) Mixed Component Set Re-sampling

Figure 5.6 Experiments result shown by percentage match over time

5.4 Comparison between component set re-sampling and standard re-sampling

In this section, we conduct a series of experiments using the mixed component set re-sampling and compare their results with standard re-sampling. The purpose of this series of experiments is to demonstrate the advantage of mixed component set re-sampling over the standard re-sampling when the system state has multiple state components. We first consider the case of 2 agents. Similar as experiment discussed in section 4.7, we vary the number of particles from 1200, to 1600 and then to 2000. The estimation error is calculated in the same way as that in experiment 4.7. The results of the experiments are shown in Fig 5. 7(a), 5.7(b) and 5.7(c):

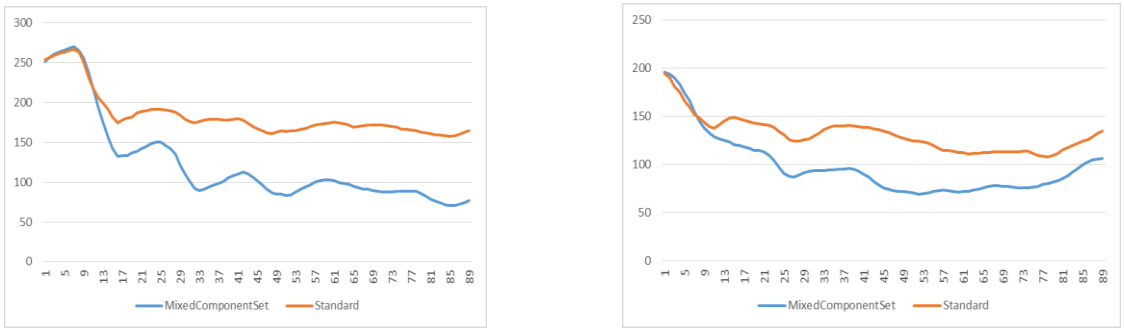


(a) 1200 particles (b) 1600 particles (c) 2000 particles

Figure 5.7 Comparison of component set and standard re-sampling

In Figure 5.7, the vertical axis represents the estimation errors and the horizontal axis represents data assimilation steps. The result is averaged from 20 experiments. Figure 5.7 shows that the estimation errors of the mixed component set re-sampling are lower than those of the standard re-sampling in all three cases of using different number of particles. This confirms the advantage of the proposed component set re-sampling.

To further show the advantage of the component set re-sampling, we conduct two other experiments where 4 and 6 agents need to be estimated, respectively. We use 1200 particles and use both the standard re-sampling and the mix component set re-sampling and compare their performance. Figure 5.8 shows the results, which are averaged from 20 simulation runs. In these experiments, due to the larger number of agents (which means higher dimensional state space), it is more difficult for particles to converge to the real system state. Figure 5.8 shows that the mixed component set re-sampling maintains lower error than the standard re-sampling algorithm in both experiments. The results show that the mixed component set re-sampling increases the performance of particle filter-based data assimilation for multiple agents.



(a)Estimating 4 Agents (b) Estimating 6 Agents

Figure 5.8 Comparison of component set and standard re-sampling

6 GRAPH MODEL-BASED SIMULATION OF BUILDING OCCUPANCY

6.1 The motivation of the graph model

In this dissertation, we utilize agent-based model for data assimilation of the smart environment. Agent-based model is effective in estimating occupants' position using particle filter algorithm. However, an agent-based model models the underlying phenomena in a low abstraction level, most of the detailed information related to the phenomena is thus taken into account while building the model. While the benefit of doing this is that the estimation can be more accurate if observations from sensors are sufficient, the drawback of using an agent-based model in estimation is also obvious. That is, the complexity of the agent-based model increases the computational cost and the agent-based model is not working well when the information volume provided by the sensor is significantly smaller than the information volume of the underlying phenomena. For example, in a scenario where binary proximity sensors are utilized to estimate the occupancy of 1000 pedestrian in a mall, the agent-based model can describe the movement dynamic of each agent in detail. However, the nature of binary proximity sensor is that it can only detect whether there are occupants in its detection area and cannot notify the occupants' actual positions. As a result, the minimum change on occupancy that can be detected is limited. Under such circumstance, the information collected from sensor is not sufficient for estimating people's locations at high resolution, therefore, using a model at high spatial resolution such as an agent-based model will not provide expected outcome and only adds to the computational cost. To handle the scenario that massive number of occupants are getting involved, a low resolution model describing the occupancy dynamics at high abstraction level is more appropriate. In this dissertation, in order to carry out estimation in a scenario that massive occupants gets involved, we design and construct a low resolution, high abstraction level model

to model the occupancy dynamic. This model simulates the occupancy dynamic in zone level. The whole building is divided into zones and is represented using a directed graph. The connectivity of the node of the graph is represented using edge link between zones. The behaviors of occupants in the building is represented using flows among the edge links of the graph. In this chapter, we describe this model in details.

6.2 General structure of the graph model

The graph model consists of two sub models. One is the model of the building, the other is the model of the occupant. The model of the building defines the graph structure of the building and how the occupants will move from a node of the building to another. The model of the occupant describes the properties and attributes of the occupants. In the model of building, the building structure is represented using a graph. The vertex of the graph represents a segment of the building structure. Typically, the segment of the building is a section of a corridor or the zone of a room. And the edges between the vertices represent the connectivity among the segments of the building. For instance, consider a floor plan of a building structure illustrated in figure 6.1.

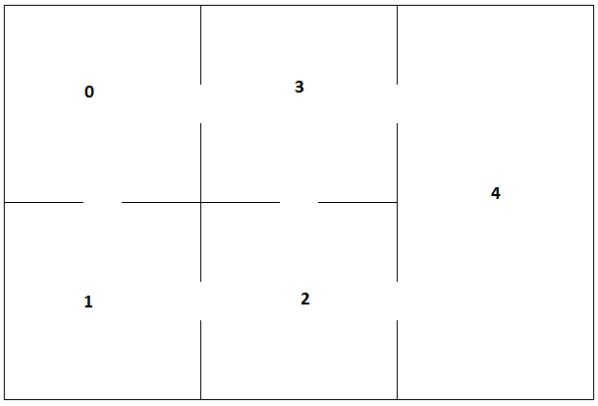


Figure 6.1 A floor plan

In figure 6.1, a floor plan with 5 rooms is presented. In this floor plan, there is a door between room 0 and room 3, room 0 and room 1, room 1 and room 2, room 2 and room 3, room 2 and room 4, room 3 and room 4 respectively. We consider that the rooms connected with doors are linked using an edge directed from one room to another. This means that, for example, there is an edge directed from room 0 to room 3 and an edge directed from room 3 to room 0 because there is a door between room 0 and room 3. The graph representing this floor plan can then be constructed as shown in figure 6.2.

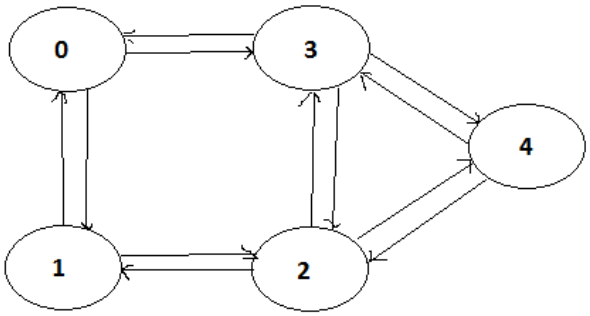


Figure 6.2 Graph representing the floor plan in figure 6.1

In this graph, vertices are connected using directed links. The vertex has several properties: the *id* of the node indicates the identity of the vertex. The *capacity* of the node indicates the maximum number of occupants that can reside in the zone represented by this vertex. The *number of occupants* of the vertex represents how many occupants are currently staying in this vertex. And the *neighbors* is an array of vertices that connected to this vertex with links. To guide the movement of occupants, we use *transition tables* in each vertex to indicate which direction the occupants staying in that vertex should move to. The transition table defines the probability that the occupants move from current vertex to its neighbor vertices. In the example shown in figure 6.2, for instance, the occupants stay in vertex 2 will have 4 choices of movement which are moving to vertex 1, 3, 4 and staying in vertex 2 respectively. The transition

table defines the probability the occupants in the vertex perform each of the movements accordingly. In this example, a transition table can be defined as shown in figure 6.3.

Vertex	Probability
-1	0.4
1	0.2
3	0.2
4	0.2

(a) An example of transition table for vertex 2

	Link 01	Link12	Link30	Link23	Link34	Link24
Link 01	0	30	30	-	-	-
Link12	30	0	-	30	-	30
Link30	30	-	0	30	30	-
Link23	-	30	30	0	30	-
Link34	-	-	30	30	0	30
Link24	-	30	-	-	30	0

(b) An example of link distance

Figure 6.3 Examples of transition table and link distance

In figure 6.3, -1 means staying in the current node; the probability for the occupants to stay in vertex 2 is 0.4; the probability for the occupants to move to the vertex 1 is 0.2; the probability for the occupants to move to the vertex 3 is 0.2 and the probability for the occupants to move to the vertex 4 is 0.2. The transition table in each vertex can be considered as a profile of movement pattern. In different situations, the profile of movement pattern can be different. For example. Consider vertex 4 in figure 6.2 as an exit room, in the normal condition, the probability

for occupants in vertex 2 to move to the vertex 4 can be defined as 0.25. However, when there is an emergency in the building such as a fire or disaster, the probability for occupants to move from vertex 2 to vertex 4 may raise to 0.8 because occupants are more likely to perform evacuation behavior and rush to the exit. Therefore, there are multiple transition tables representing different profiles of occupants' movement patterns in each vertex of the graph. Specifically, the vertex in the model can be represented using a 5 elements variable v, where

$$v = < i, c, n, L', D, E, T >$$

In this equation, i is the id of the vertex, c is the capacity of the room represented by the vertex, n is the number of occupants in the room represented by the vertex, L' is the collection of links that connect with the vertex, D is the distances between the links in L' this distance is specified in a table shown in figure 6.3(b), E is a collection of neighbor vertices of the vertex and T is a collection of transition tables of the vertex.

The link of the graph also consists of several properties. The "*from*" vertex defines the original vertex of the link and the "*to*" vertex defines the vertex that the link pointed to. The *flow capacity* of a link defines the maximum number of occupants that can go through this link at each time step. To model the congestion in the building, for each link, we define a *move queue* to describe the queue formed by the occupants that is moving through this link. At each time step, the occupant decides which room to move to. The occupant then spends a period of time to move to the link connecting the current vertex where the occupant resides to the destination vertex. After that, the occupant registers himself to the move queue of that link. Number of occupants that equal to the flow capacity of the link are removed from the move queue at each time step. After being removed from the queue, these occupants arrive at their destination after being removed from the queue. Besides the property of the link itself, there is also a property *link*

distance which represents the distance between two links. The link of the graph can be considered as the gate between rooms. The link distance is then the distance between the gates of the rooms. As the link distance is known, we can estimate the time required for the occupant to travel in the room in order to arrive its destination. An example of link distance is shown in 6.3(b), where the link distance between two arbitrary links in the same vertex is set to 30. Specifically, the link in the graph model can be represented using a 4 elements variable l, where

$$l =< i, v_{from}, v_{to}, c_f, q >$$

In this equation, i represents the id of the link, v_{from} represent the from vertex of the link, v_{to} represent the to link of the vertex, c_f represents the flow capacity of the link, and q is the move queue of the link.

The model of occupant describes properties of the occupants and how occupant moves from one vertex of the graph to another. The *state* of the occupant defines what profile of movement pattern will be used to determine the moving direction of the occupant. In other words, if the vertex of the graph contains multiple transition tables, the state of occupant determines which transition table will be used to guide the occupants movement. The *location* of the occupant defines the id of the vertex that the occupant is currently staying in. The *id* of the occupant defines the identification of the occupant. The *isInQueue* property indicates whether the occupant has registered himself into a move queue of a link. The *destination* defines the vertex that the occupant decides to move to. The "*from*" link defines the link that the occupant went over to enter its current location. The "*to*" link defines the link that occupant will use to move to its destination. The *speed* defines the distance an occupant move along in a single time unit. The *action* of the occupant defines whether the occupant is moving to other vertex or the occupant is staying in his current location. The *elapse time* defines the time the occupant have

spent in his current location. The *delay* of the occupant is a counter that counts how many time units the occupant will spend in his current location. This counter is reset each time an occupant enters a new vertex. The value assigned to the delay of occupant is calculated based on the link distance between the "from" link and "to" link of the occupant and the speed of the occupant. The counter reduces by 1 at each time step, when it reaches 0, it means the occupant arrives the "to" link and is ready to move to its destination vertex. The occupant then register itself to the move queue of the "to" link. The *action* property defines the action that the occupant is performing; if it is 0, it means the occupant stays in current vertex; if it is 1, it means the occupant is moving to another vertex. Specifically, the occupant in the model can be represented using variable o, where

$$o = < ph, v_{position}, id, l_{from}, s >$$

$$s = \begin{matrix} staying & < t_{elapse}, t_{stay\_delay} > \\ moving & < t_{elapse}, t_{\text{move\_delay}}, l_{to}, spd > \\ in\ queue & < V_{destination} > \end{matrix}$$

In this equation, ph represents the phase of the occupant to determine which transition table to be used to guide occupants movements, $v_{position}$ represents the current vertex the occupant reside in, id represents the identification of the occupant, l_{from} represent the link that the occupant come from. s represents the state of the occupants and has three different values, staying, moving and in queue. Each of the value is associated with several properties. l_{to} represent the link that occupant is going to, $v_{destination}$ represents the destination of the occupant, spd represents the speed of the occupants, t_{elapse} time represent the time elapsed when the occupant stay in current room, t_{stay_delay} represents the time count that the occupant will stay in current vertex, t_{move_delay} represents the time count that the occupant will be moving in current vertex. The collection of m occupants in the model is represented by:

$$O = < o_1, o_2, o_3, ..., o_m >$$

The model of the building and the model of the occupant together define the general structure of the graph model. The graph model evolves according a set of rules to simulate the movement dynamics of the occupants. In the following section, we describe the procedure to simulate occupants' moving dynamic using the graph model in detail.

6.3 The dynamics of the model

At each time step, the graph model evolves according to two sets of different rules. One set of rules defines the behavior of the occupants at each time step including how they decide the destination vertex and how they move to the destination, another set of rules defines the behavior of occupant in the move queue of each link.

The rules to decide the occupant's destination and how they move to the destination is defined as follows. For the model of occupant movement behavior, at each time step, if the delay of occupant is larger than 0, then the delay of the occupant decreases by 1 and elapse time of the occupant increases by 1. The delay of the occupant serve as a counter to count how many time step the occupant will stay in its current location and the elapse time of the occupant represents the number of time step has been staying in current location. If the occupant's delay is set to -1, this means this occupant has just entered its current location and needs to determine its destination and delay. To determine the destination of the occupant, occupant looks up the transition table of the current vertex it resides in and select movement behavior profile according to its current state. The transition table specifies the probability of occupant in a vertex to travel to each neighbor of the vertex. The occupant use a generated random number to determine which vertex to travel to according to the probabilities specified in the transition table. This procedure is similar to the roulette wheel selection. The neighbor vertex of the vertex occupant currently

resides in is assigned to a portion of the roulette wheel while the area of the portion is proportional to the probability of traveling from current vertex to the neighbor vertex. By generating a random number between 0 and 1, roulette wheel spins and the selection point points to some portion of the wheel, then the neighbor vertex assigned to that portion is selected to be the destination vertex of the occupant. Take the transition table illustrated in figure 6.2 as an example, according this transition table, the shares of the portion of the roulette wheel is specified as follows: portion 0~0.4 is assigned to vertex -1 which means staying in current vertex, portion 0.4~0.6 is assigned to vertex 1, portion 0.6~0.8 is assigned to vertex 3 and portion 0.8~1.0 is assigned to vertex 4. To select a destination, the system generates a random number between 0 and 1, say, 0.7. Because 0.7 is within the portion 0.6~0.8, vertex 3 is selected as the destination of the occupant. After the determination of the destination, the next step is to determine how long the occupant will stay in current vertex. The occupant stays in current vertex for two reasons: if the occupant's destination is current vertex, it means it chooses to stay in this vertex for some time; if the occupant's destination is one of the neighbor vertices of current vertex, the occupant will spend some time to travel to that vertex. The delay an occupant chooses to stay in current vertex is determined by two attributes minStayTime and maxStayTime, the actual time an occupant chooses to stay in current vertex is a random value between minStayTime and maxStayTime. The delay an occupant will spend to travel to the neighbor vertex is determined by the speed of the occupant and the link distance between the link that the occupant used to entering its current vertex and the link connecting its current vertex to the neighbor vertex it will travel to. Specifically, it is a random value between the maxDelay and minDelay. The maxDelay is calculated by the link distance divided by the speed of the occupant and the minDelay is either 0 (in the case maxDelay is smaller than elapse time) or maxDelay

minus elapse time (int the case maxDelay is larger or equal to the elapse time). After the delay of the occupant is set, it reduces by 1 at each time step. When it reaches 0, it means the staying time for the occupant has expired or the occupant has finished traveling and ready to enter the destination vertex. If it is the first case, then delay is reset to -1 so that the occupant goes over this procedure again to perform movement behavior. If it is the second case then the occupant is queued into the moving queue of the link that connecting the occupant's current vertex to its destination vertex. The procedure that the occupant moves in the vertex of the graph can be summarized as follows:

function simulate(timeStep)
//System behavior
Decide_system_phase();
// Agent behavior
for each o_i in O
o_i.phase = systemPhase;
if o_i.s=staying
if $o_i.t_{stay\_delay} > 0$
$o_i.t_{stay\_delay} \leftarrow o_i.t_{stay\_delay}-1$, $o_i.t_{elapse} \leftarrow o_i.t_{elapse}+1$
else if $o_i.t_{stay\_delay} = 0$
decision_making(o_i)
end if
end if
if o_i.s=moving
if $o_i.t_{move\_delay} > 0$

$o_i.t_{move\_delay} \leftarrow o_i.t_{move\_delay}-1$, $o_i.t_{elapse} \leftarrow o_i.t_{elapse}+1$

else if $o_i.t_{move\_delay} = 0$

addQueue(o_i, $o_i.l_{to}.q$)

$o_i.s \leftarrow$ in queue

end if

end if

if $o_i.s$=in queue

// do nothing

end if

end for

// Queue Behavior -- remove agent from queue

for each link *l* in the graph

for g←1 to $l.c_f$

if $o.v_{destination.c} > o.v_{destination.n}$

$o \leftarrow$poll($l.q$)

$o.v_{position} \leftarrow l.v_{to}$

$o.l_{from} \leftarrow$l

$o.t_{elapse} \leftarrow 0$

$l.v_{from}.n \leftarrow l.v_{from}.n-1$

$l.v_{to}.n \leftarrow l.v_{to}.n+1$

decision_making(o)

end for

end for

function decision_making(o_i)

transition_table←choose_transition_table($o_i.v_{position}$.T, o_i.ph)

$o_i.v_{destination}$ ←select_destination(transition_table)

$o_i.l_{to}$ ←findLink($o_i.v_{position}$, $o_i.v_{destination}$)

if ($o_i.v_{destination}$= $o_i.v_{position}$) // the case of staying

r←rand(0,1)

s←staying

$o_i.t_{stay\_delay}$ ←minStaytime+r*(maxStaytime-minStaytime)

else // the case of moving

maxDelay←linkDistance($o_i.l_{from}$,$o_i.l_{to}$)/o_i.spd

minDelay←maxDelay- $o_i.t_{elapse}$ if maxDelay>= $o_i.t_{elapse}$

minDelay←0 if maxDelay< $o_i.t_{elapse}$

r←rand(0,1)

s←moving

$o_i.t_{move\_delay}$ ←minDelay+r*(maxDelay-minDelay)

end if

In this algorithm, choose_transition_table is a function that select a transition table according to the occupant's state s, select_destination is a function takes transition_table as input and randomly select a vertex as the destination of the occupant, findLink(v1,v2) returns the link pointed from vertex v1 to vertex v2, rand(a,b) is a function generating a random number that is between a and b. linkDistance(l1,l2) returns the link distance between the link l1 and link l2. addQueue(o, q) add the occupant o to the queue specified by q. The decide_system_phase()

function is a function to determine the phase ph of each of the occupants so that occupants select transition table to guide movements accordingly.

The rules to decide the behavior of occupant in the move queue of each link is defined as follows. At each time step, for each link of the graph, remove number of occupants equals to the flow capacity of that link from the moving queue. The occupants that removed from the move queue enter their destination vertices where the destinations of the occupants become the locations of the occupants; meanwhile, the delays of the occupants are reset to -1, the isInQueue properties are reset to false, the elapse time properties of the occupants are reset to 0, the "from" link properties of the occupants are set to the link that the occupants are removed from.

These two sets of rules together define the transition function of the graph model. In the following section, we describe a case study using the proposed graph model to simulate the occupancy dynamic in a building.

6.4 Discussion

The graph model developed can simulate the dynamic of occupancy at a high abstraction level. There are several potentials for further developing the model. First, in current model, it is assumed that there is only one door (two links) directing from one room to another. However, to make the simulation more realistic, there can be multiple doors connecting the same two rooms. Different doors can be differentiated using the id of the link and the link distance specifying the distance between two doors is assigned with the id of the link. Second, in current model, a door is represented using two links-an outgoing link and an incoming link, the number of occupants go through the door is calculated on the two links separately and each link has its own flow capacity. However, this is not realistic. To improve the model, the two links should share the same flow capacity. To do this, while moving occupants from the queue of the link, first move

one occupant from incoming link, then move one occupant from the outgoing link and repeat to do this until the flow capacity of link is reached. Third, in the current model, the congestion does not affect the flow rate, this is not realistic because in the real scenario, the congestion will slow down the movements of occupants. To model this, we can define the relationship between the flow capacity and the congestion level so that the flow rate is reduced when congestion level increases. Fourth, in current model, the destination of an occupant is selected using select_destination() function of the decision making method and is purely based on the transition table of each vertex. This method cannot be directly used to model the occupant's adaptive behavior. To reflect the occupant's adaptive behavior, the select_destination() function can be made more complex. For example, the occupant can be made to choose a destination with less queue size to avoid congestion or choose the same destination with the majority of occupants to model the herding behavior. A nature way to do this is to define the transition probability in transition table using a function (instead of using a constant number as in current model). By doing this, more specific behaviors of the occupants can be defined. Fifth, in current model, the transition probabilities in transition table are set to be constants; however, in an environment equipped with sensors, the transition probability can be updated in real time using data obtained from sensors so that the model can calibrated. Sixth, the current model that uses transition table to guide the occupants' movements is a top-down approach, it does not model the occupants' individual behaviors, while this is applicable in cases where a massive number of occupants get involved, in the case where the smaller of occupants are involved, it is still useful to model each occupants individual behaviors. This can be done using a bottom-up approach, where the occupant can select a destination purely based on its own profile and the environment (whether there is a door damaged or whether large number of occupants are congested). This is similar to

the traditional agent-based approach. Each occupant is independent and has its own rule of behavior. In this case, the top-down approach such as transition table is not useful. Seventh, the environment in current model is constant, however, in the case such as an emergency, the environment may change due to hazard. For example, a door may be damaged by the fire, in this case, the flow capacity of this door reduces to 0 (no occupants are allowed to pass the door) and the transition table need to be modified so that occupants have less probability to try to move through this door. A changing environment can make the simulation more realistic.

6.5 The GUI

To better demonstrate the simulation, we develop a GUI to show the simulation procedure. We use lines to represent the walls and dots to represent occupants. For occupants are currently inside the rooms but are not in any queue of the links, we distribute them randomly in the rooms; for occupants that are currently inside the queues of the link, we draw them around the doors to simulate the scenario of congestions. The number of occupants that are in the queues of the links are presented near the doors using red string. For each room, we display the capacity of the room, the number of occupants in the room, and the number of occupants that are not inside the queues in the room at the center of the room. The first line of the numbers at the center of each room displays the capacity of the room. The second line of the numbers at the center of each room displays the total number of occupants in that room. The third line of the numbers at the center of each room displays the number of occupants that are not inside the queues in the room. At the bottom of the map, the total time steps of the simulation are displayed. The GUI is shown in figure 6.4.

6.6 Case study

In this section, we demonstrate an example that uses the graph model to simulate a scenario of emergency evacuation. We use the floor structure illustrated in figure 6.1 as the environment. 2000 occupants are deployed randomly in five rooms. The capacity of each room is set to 2000. The maxStaytime is set 10 time steps and minStaytime is set to 3 time steps. The floor structure is transformed to a graph as being illustrated in figure 6.2. The flow capacity of the links in the graph is set to be 5. For each vertex in the graph, there are two different transition tables to define the probability of the occupant moving from the vertex to other vertex. One transition table defines the transition probability of occupant in normal state in which occupants moves randomly in the building, the other transition table defines the transition probability of occupant in evacuation state in which occupants tries to rush to room 4 which is defined as an exit room. The transition table for normal state is shown in table 6.1:

Table 6.1 The transition table for normal state

	Vertex 0	Vertex 1	Vertex 2	Vertex 3	Vertex 4
Vertex 0	0.33	0.33	-1	0.33	-1
Vertex 1	0.33	0.33	0.33	-1	-1
Vertex 2	-1	0.25	0.25	0.25	0.25
Vertex 3	0.25	-1	0.25	0.25	0.25
Vertex 4	-1	-1	0.33	0.33	0.33

Table 6.2 The transition table for evacuation state

	Vertex 0	Vertex 1	Vertex 2	Vertex 3	Vertex 4
Vertex 0	0.05	0.15	-1	0.8	-1
Vertex 1	0.15	0.05	0.8	-1	-1
Vertex 2	-1	0.05	0.05	0.1	0.8
Vertex 3	0.05	-1	0.1	0.05	0.8
Vertex 4	-1	-1	0.0	0.0	1.0

In the transition table illustrated in table 6.1 and table 6.2, the each row defines the transition table for a vertex. The value in row i and column j defines the probability that an occupant moves from vertex i to vertex j. If the value of row i and column j equals -1, it means there is no connection from vertex i to vertex j and therefore it is impossible for an occupant to move from vertex i to vertex j. The link distances representing the distance from one link to another (the distance between the doors) are set uniformly to 30 and the speed of the occupant is set to 3 unit distance per time step. The simulation is separated into two period. In the first period (first 50 time steps), the occupants' states are set to normal, therefore, the occupants move using the transition table illustrated in table 6.1. In the second period (after 50 time steps), we assume there is an emergency in the building and everyone is trying to exit the building, therefore, the occupants move using the transition table illustrated in table 6.2. In the transition table illustrated in table 6.1, the occupant has equal chance to select one of the neighbor vertex of his current vertex as his destination while in the transition table illustrated in table 6.2, the occupant has significantly larger chance to set vertex that is nearer to the exit vertex (which is vertex 4) as its destination. Since the flow capacities of the links are uniformly set to 5 which means only 5 occupants can move through the link at each time step, it causes congestions when occupants are

moving from one vertex to another. The congestion happens whenever the move queues of the links are not empty. We represent occupants using dots, the simulation is shown in figure 6.4, 6.5 and 6.6.

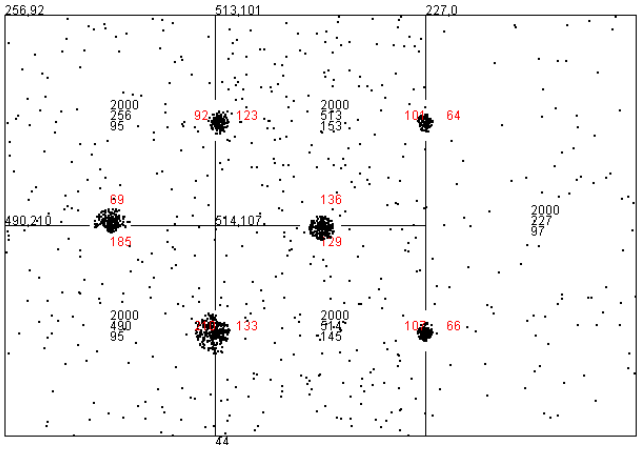


Figure 6.4 Simulation at time step 30

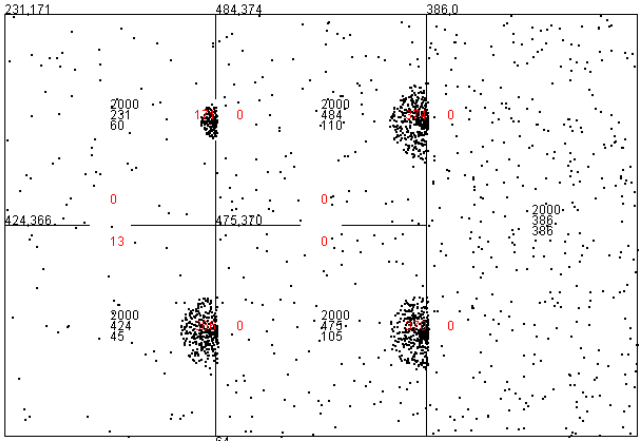


Figure 6.5 Simulation at time step 60

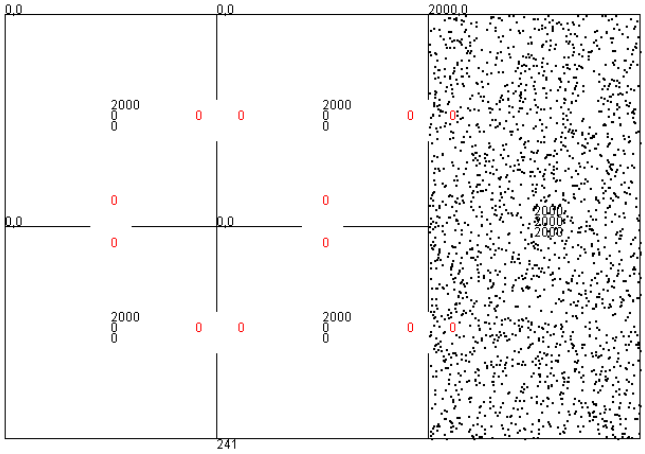


Figure 6.6 Simulation at time step 300

At simulation time 30, the occupants move in normal state, the probabilities for an occupant to choose neighbors of his current vertex as destination are equal, therefore, the occupants moves in the building randomly. Because the flow capacities of the links are set to 5, the move queue of the links are always not empty, therefore, there are congestions (represented by aggregations of the black dots) around each of the links in the graph. At simulation time 60, the occupants move in evacuation state, the occupants have higher probability to move to the vertex that is nearer to the exit vertex, therefore, the congestion happens majorly on the links directing to the exit vertex. At simulation time 240, all the occupants have finished evacuation movement and reside in vertex 4.

6.7 Experiments

In this section, we carried out a series of experiments to evaluate the model constructed. The experiments simulate the scenarios that there is an emergency in the building and the occupants use the evacuation transition table shown in table 6.2 to guide their movements. The floor plan shown in figure 6.1 and the graph for the building structure in figure 6.2 are used as the building environment in these experiments. The minimal staying time for the occupant to stay in a room is set to 3 time steps and maximum staying time is set to 10 time steps. In each of

the experiment, we show three different quantitative measurement of the simulation system: the occupants in room 2, the occupants in room 4 and the occupants in queue of the link 2 to 4.

6.7.1 Robustness of the model

This series of experiments aims at test the robustness of the model. We run 5 simulation experiments using a same configuration of the model to see whether the simulation result is correct. In these simulations, the number of occupants are set to 2000. The capacities of the rooms are set uniformly to 2000. The flow capacities of the links are set uniformly to 5. And the link distances between arbitrary two links are set uniformly to 30. The result is shown in figure 6.7, figure 6.8 and figure 6.9.

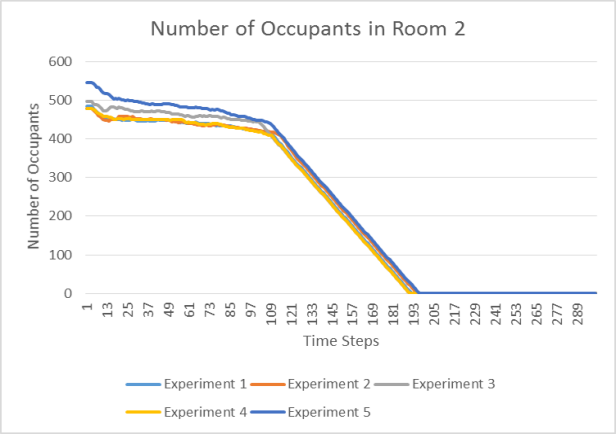


Figure 6.7 Robustness test: occupants in room2

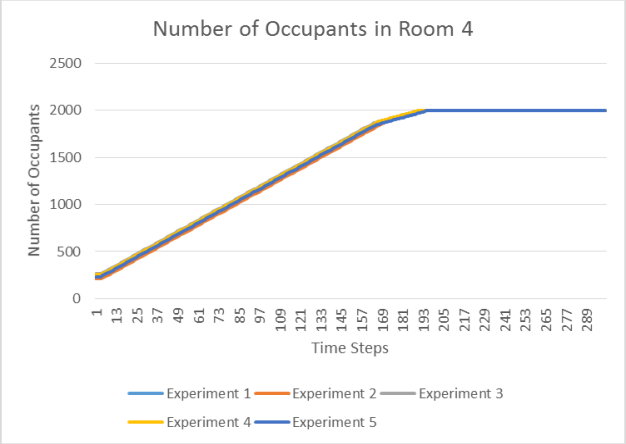


Figure 6.8 Robustness test: occupants in room 4

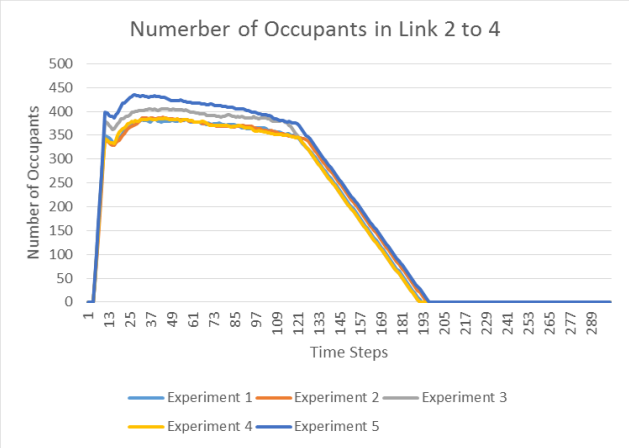


Figure 6.9 Robustness test: occupants in link 2 to 4

In figure 6.7, the curve shows two evolving stages. At the first stage, the number of occupants entering room 2 and number of occupants leaving room 2 are nearly equal, therefore, the curve shows the number of occupants in room 2 does not decrease significantly and maintains at a certain level. At the second stage, the occupants in room 1 and room 3 have almost all moved out, the number of occupants entering room 2 becomes 0, therefore, the number of occupants in room 2 decrease significantly and the figure presents a linear decreasing curve at this stage and eventually reaches 0. In figure 6.8, because room 4 is the exit room, in the emergency evacuation scenario, all of the occupants are trying to entering this room. As a result,

the number of occupants in this room keeps increasing linearly until the room capacity 2000 is reached. In figure 6.9, the number of occupants in the queue of link 2 to 4 also presents with 2 stages of behaviors. At the first stage, the number of occupants moving into the queue is nearly equal to the number of occupants moving out of the queue. Consequently, the curve of the number of occupants maintains at some certain level and does not decrease significantly. After some time, all of the occupants in room 2 have moved out of the room and the number of occupants in room 2 becomes nearly 0, the number of occupants entering the queue becomes nearly 0. As a result, the number of occupants in the queue decrease significantly afterwards and eventually reaches 0. From the figures, we can see that all the 5 experiments carried out have similar curves on all the quantitative measurement, this proves that our model is robust and correctly simulates the underlying phenomena of evacuation.

6.7.2 Impact of number of occupants

This series of experiments aims at evaluate the effect of varying number of occupants in the simulations. The capacities of the rooms are set uniformly to 2000. The flow capacities of the links are set uniformly to 5. And the link distances between arbitrary two links are set uniformly to 30. The number of occupants are set to 1500, 2000 and 2500. The results are shown in figure 6.10, figure 6.11 and figure 6.12.

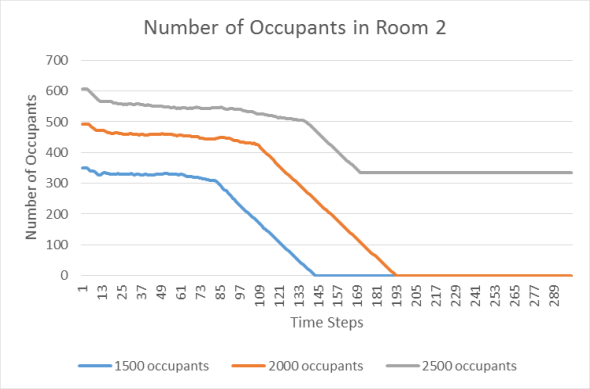


Figure 6.10 Vary number of occupants: occupants in room 2

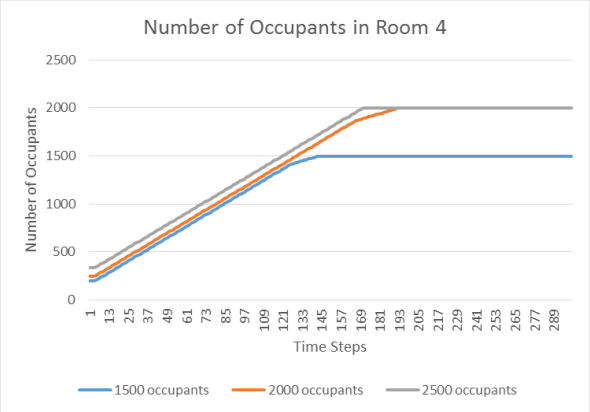


Figure 6.11 Vary number of occupants: occupants in room 4

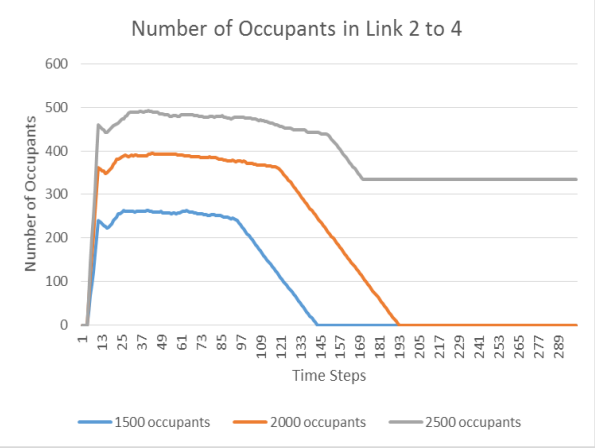


Figure 6.12 Vary number of occupants: occuapnts in link 2 to 4

In figure 6.10, it can be seen that in simulation of 2500 occupants scenario, the number of occupants in room 2 decreases to and maintains around 350 while in simulation of 1500 and 2000 occupants scenario, the number of occupants eventually decreases to 0. This is because the capacity of the exit room-room 4 is set to be 2000. As a result, in 1500 and 2000 occupants cases, all of the occupants in room 2 can eventually enter room 4 while evacuating because the capacity of room 4 is larger than the toatal number of occupants. However, in 2500 occupants case, there are 500 occupants that can not enter room 4 when room 4 is full, these occupants will stay in room 2 and room 3. This is why in 2500 occupants scenario, the number of occupants in room 2 eventually maintains around 350. In figure 6.11, since all of the occupants will finally reach room 4, the number of occupants in room 4 maintains at a certain level. In 2000 and 2500 occupants case, since the capacity of room 4 is set to be 2000, the number of occupants in room 4 will eventually reach this number after all of the occupants moves towards room 4. In 1500 occupants case, the number of occupants in room 4 eventually increases to the total number of occupants which is 1500. In figure 6.12, the number of occupants in link 2 to 4 eventually maintains at around 350 in 2500 occupants case since room 4 is full, while in 1500 and 2000 occupants case the number of occupants decreases to 0 eventually. In both figure 6.10 amd 6.12, as the number of occupants increase, the number of occupants in room 2 and in link 2 to 4 decreases at each timestep. This proves that the model is correct and robust while varying the number of occupants in the building.

6.7.3 Impact of room capacity

This series of experiments aims at evaluate the effect of varying the capacity of one room in the simulations. The number of occupants are set to 2000. The capacity of room 4 is set to 1000,1500,2000. The capacity of the other rooms are set uniformly to 2000. The flow capacities

of the links are set uniformly to 5. And the link distances between arbitrary two links are set uniformly to 30. The results are shown in figure 6.13, figure 6.14 and figure 6.15.

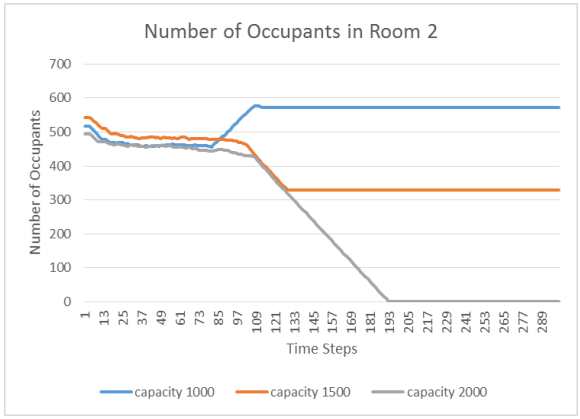


Figure 6.13 Vary room capacity: occupants in room 2

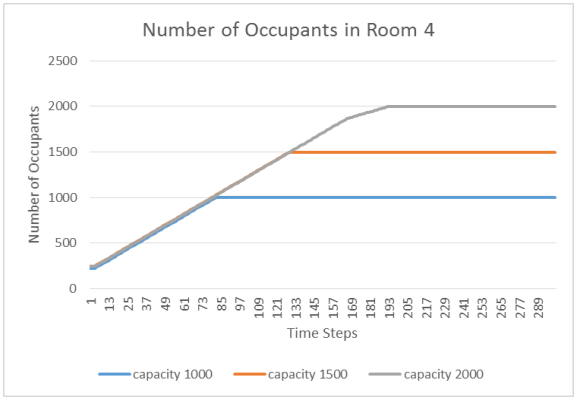


Figure 6.14 Vary room capacity: occupants in room 4

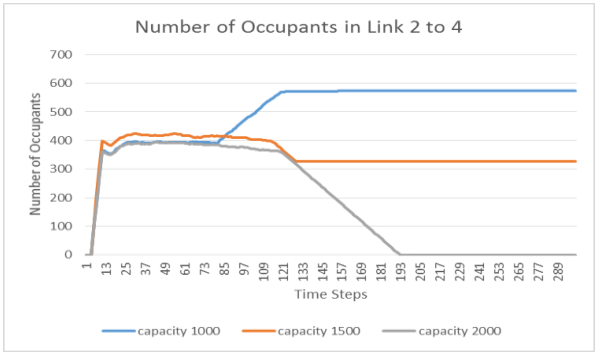


Figure 6.15 Vary room capacity: occupants in link 2 to 4

In figure 6.13, it can be seen that as the capacity of room 4 increases the number of occupants at final simulation steps in room 2 decreases. This is because as the capacity of room 4 increases, more occupants are able to entering room 4 from room 2 and other rooms connected to room 4. When the capacity of room 4 has significantly low capacity such as 1000, a large number of occupants will stuck in room 2 and can not enter room 4 because room 4 is full. In figure 6.14, as the capacity increases, the final number of occupants in room 4 increases too. In figure 6.15, as same as in figure 6.13, the occupants in link 2 to 4 increases as the capacity of room 4 decreases, this is because the room 4 is full and occupants stuck in the queue of link 2 to 4 and are not able to move into room 4.

6.7.4 Impact of flow capacity

This series of experiments aims at evaluate the effect of varying the flow capacity of one link in the simulations. The number of occupants are set to 2000. The capacity of the rooms are set uniformly to 2000. The flow capacity of the link 2 to 4 is set to 2, 5, 20 and 50. The flow capacities of other links are set uniformly to 5. And the link distances between arbitrary two links are set uniformly to 30. The results are shown in figure 6.16, figure 6.17 and figure 6.18.

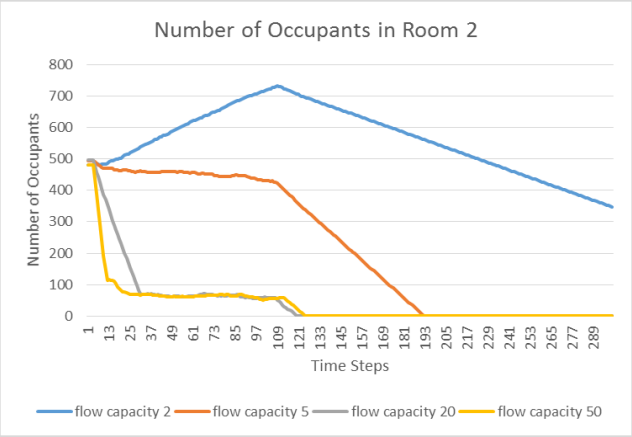


Figure 6.16 Vary flow capacity: occupants in room 2

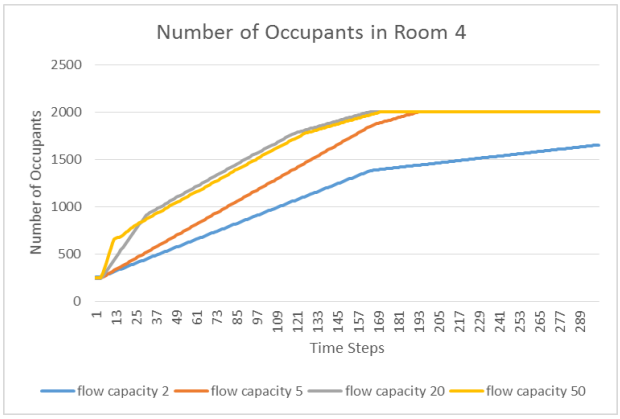


Figure 6.17 Vary flow capacity: occupants in room 4

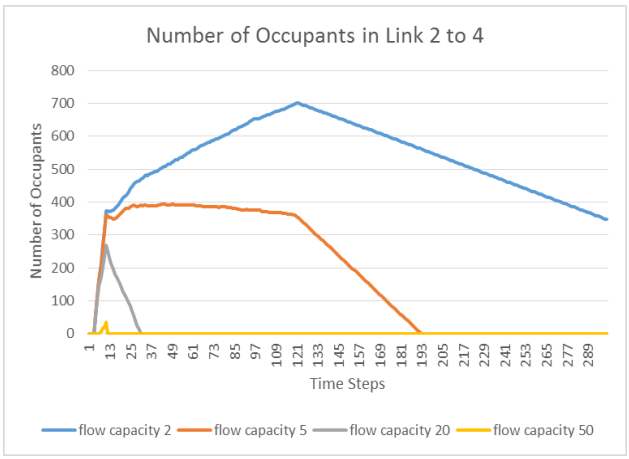


Figure 6.18 Vary flow capacity: occupants in link 2 to 4

In figure 6.16, as the flow capacity increases, the gradient of decreasing curve of the number of occupants in room 4 significantly increases. This is because at each time step, more occupants are allowed to move to room 4 by being removed from the queues of the link 2 to 4. As a result, the congestion caused by the queued occupants in link 2 to 4 dismisses rapidly. Specially, the occupants in room 2 in flow capacity 2 case increases, this is because the number of occupants entering room 5 per time step is larger than the number of occupants exiting room 2. In figure 6.17, as the flow capacity of link 2 to 4 increases, the number of occupants in room 4 increases faster. Because in both flow capacity 20 and flow capacity 50 cases, the congestion in

link 2 to 4 dismiss, so the curves of the number of occupants of both cases overlap with each other but increases faster than flow capacity 5 case. In figure 6.18, the occupants in the queue of link 2 to 4 decreases significantly and reaches 0 faster as the flow capacity increases. This is because more occupants are removed from the queue when flow capacity increases. Again, the number of occupants in queue increases in the flow capacity 2 case, this is because the number of occupants entering the queue is larger than the number of occupants exiting the queue at each time step.

6.7.5 Impact of link distance

This series of experiments aims at evaluate the effect of varying the link distance between two specific link in the simulations. The number of occupants are set to 2000. The capacity of the rooms are set uniformly to 2000. The flow capacities of the links are set uniformly to 5. And the link distances between arbitrary two links are set uniformly to 30 except the link distance between the link 1 to 2 and the link 2 to 4 which is set to be 30, 200, 500. The results are shown in figure 6.19, figure 6.20 and figure 6.21.

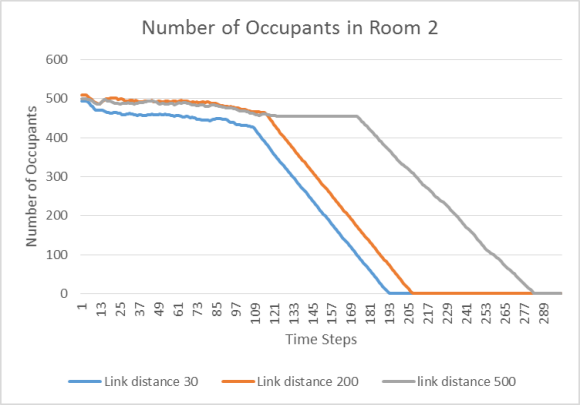


Figure 6.19 Vary link distance: occupants in room 2

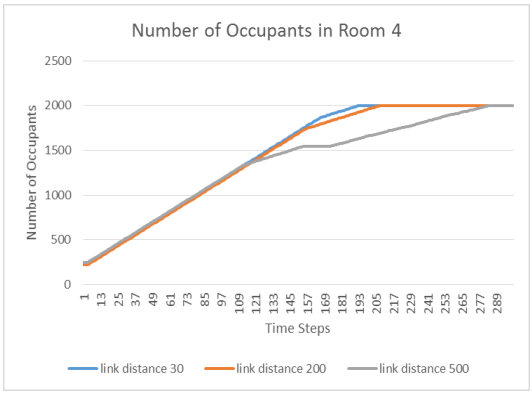


Figure 6.20 Vary link distance: occupants in room 4

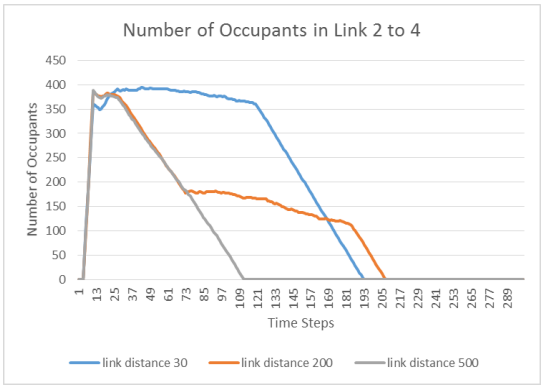


Figure 6.21 Vary link distance: occupants in link 2 to 4

In figure 6.19, the time steps for the number of occupants to reach 0 increases as the link distance between the link 1 to 2 and 2 to 4 increases. This is because the time required for the occupants that enters room 2 from the link 1 to 2 to travel to the link 2 to 4 increases. As a result, all of the occupants entering room 2 from the link 1 to 2 and to the link 2 to 4 will have to stay in room 2 for longer time intervals. This is also the reason behind that in figure 6.20, as the link distance increases, the time required for the number of occupants in room 4 reaches 2000 increases too. In figure 2.21, since the link distances are large in link distance 200 and link distance 500 case, there are few occupants entering the queue at the beginning of the simulation(because occupants have to spend more time on traveling to the link 2 to 4), therefore,

the number of occupants in link 2 to 4 does not maintain at a certain level as in link distance 30 case. After the occupants spend the time to travel to the link 2 to 4, in link distance 200 case, the occupants start to entering the queue of link 2 to 4, so the curve of number of occupants maintains at a certain level. However, in link distance 500 case, since the occupants spend too much time on traveling from link 1 to 2 to link 2 to 4, the number of occupants in link 2 to 4 eventually drop to 0.

6.7.6 Scenario: normal state to evacuation state

This series of experiments aims at simulating a scenario that the occupants firstly move randomly in the building, then an emergency happens and occupants evacuate the building. The number of occupants are set to 2000. The capacity of the rooms are set uniformly to 2000. The flow capacities of the links are set uniformly to 5. And the link distances between arbitrary two links are set uniformly to 30. The transition table is set to be normal state as in table 6.1, and then set to be evacuation state as in table 6.2 after 50 time steps .The results are shown in figure 6.22, figure 6.23 and figure 6.24.

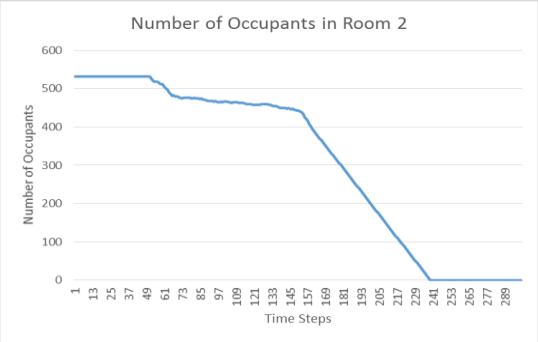


Figure 6.22 Real scenario: the number of occupants in room 2

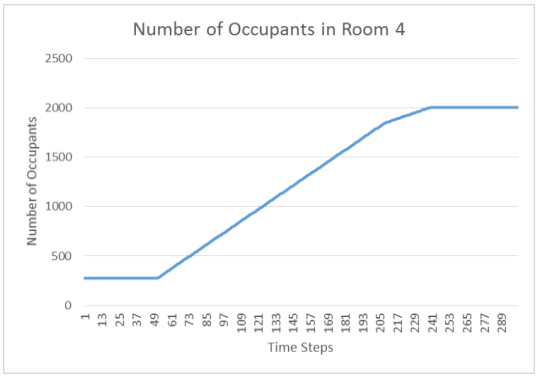


Figure 6.23 Real scenario: the number of occupants in room 4

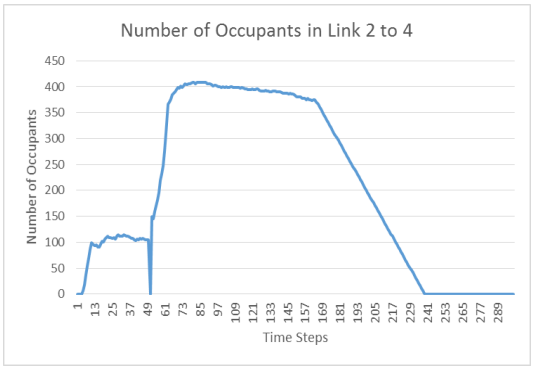


Figure 6.24 Real scenario: the number of occupants in link 2 to 4

In figure 6.22, it can be seen that before time step 50, the number of occupants in room 2 maintains at a certain level, this is because in the transition table of normal state, the probabilities for occupants to move to each neighbor of their current locations are equal. As a result, there are always a number of occupants in each link of the graph. Due to this reason, number of occupants that move into a room and number of occupants move out of a room is equal. Therefore, the total number of occupants in each room stay unchanged. This also applies to the room 4 case, where the number of occupants maintains at a certain level until time step 50. In figure 6.24, in the first 50 time steps, since the probabilities for occupants to move to the neighbors of their current locations are equal, the number of occupants in the queue of the link is significantly smaller than the number of occupants in the queue after time step 50 where the probability for occupants to

travel from node 2 to node 4 is large. At time step 50, the number of occupants in link 2 to 4 drop to 0. This is because in current implementation, the queues are cleared whenever there is a state change of occupants so that all the occupants reschedule their movement when their states change. After time step 50, the behavior of the model is similar to the behavior of the model in experiment 6.6.1.

6.8 Verification

To verify the correctness of our model, we carried out a series of experiments using the agent-based model described in chapter 3 to justify the developed graph-based model. In these experiments, an agent-based model and a graph model are built sharing some common parameters. We run a simulation using the agent-based model and then run another simulation using the graph based model. We then compare the results of the two simulations to validate the graph-based model. The floor map we used in the simulation is presented as follows:

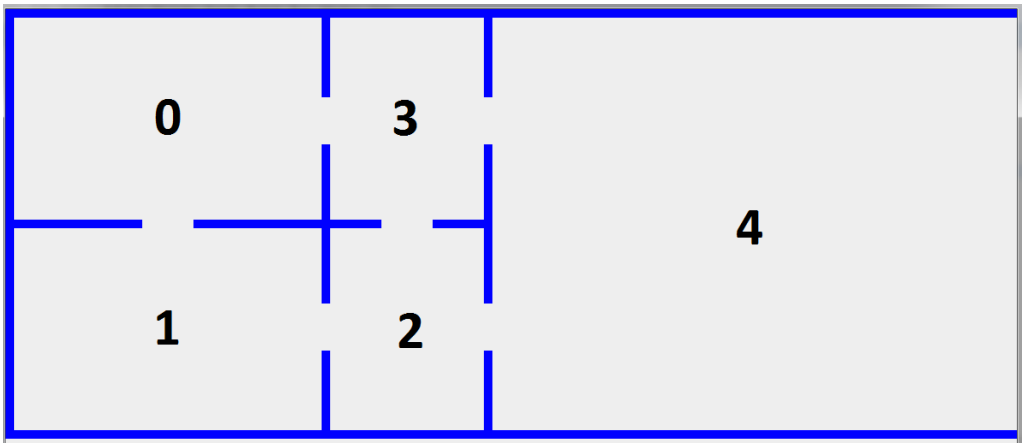


Figure 6.25 The floor map

In the agent-based model, the width of room 0 and room 1 is set to be 360 unit distance, while the height of room 0 and room 1 is set to be 236 unit distance. The size of the agent is set to be with diameter 15; as a result, the capacity of room 0 and room 1 is around 376. The width

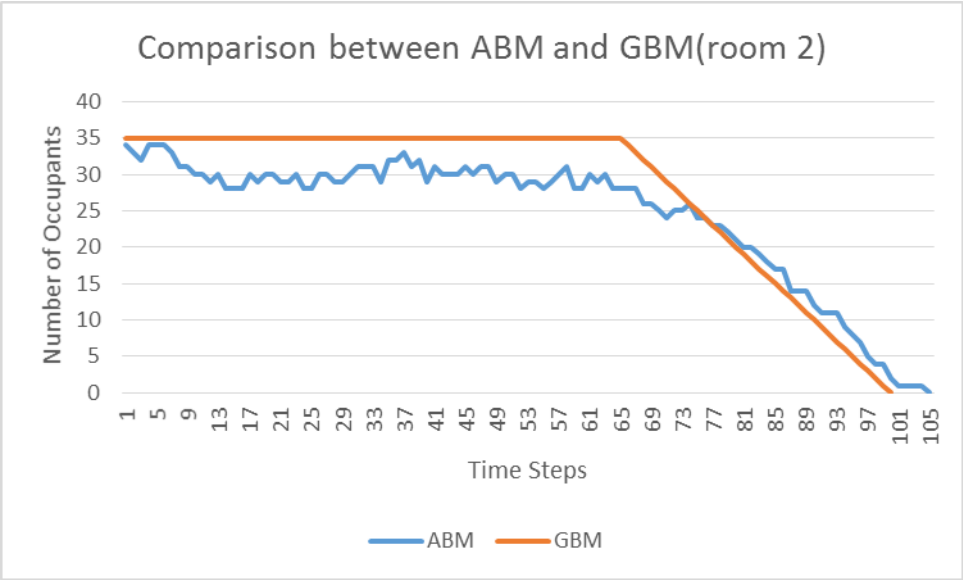
of room 2 and room 3 is set to be 180, while the height of room 2 and room 3 is set to be 236; as a result, the capacity of room 2 and room 3 is around 188. The width of the doors linking the rooms is set to be 55 unit distance. The speed of the agent is set to be 9 distance per second. The capacity of room 4 is around 2000.

In the graph-based model, the capacity of the room 0 and room 1 is set to be 376 and the capacity of the room 2 and room 3 is set to be 188. The occupants's speed is set to be 3 and the link distance between doors is set to be 60. Because the width of room 2 is set to be 180 and agent's speed is 9 in the agent-based model, therefore, the time required for an occupant to travel from room 1 to room 4 in both models is 20 time steps. By observing the agent-based simulation, we set the flow capacity of each of the link to be 1 .The capacity of room 4 is set to be 2000.

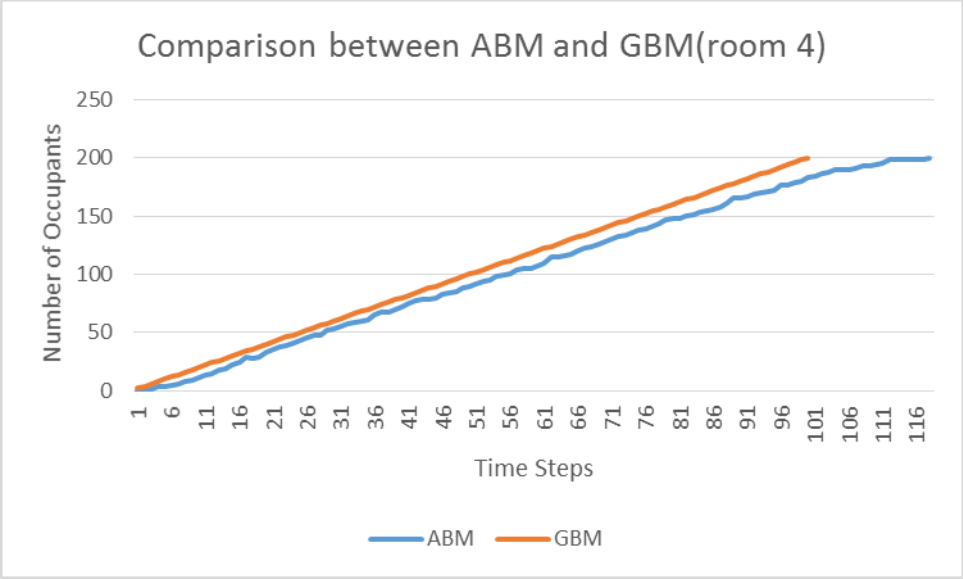
The simulations are carried out as follows: we assume there are agents in room 0, 1, 2, 3 and there is an emergency in the building; consequntly, the agents move to evacuate the building. Room 4 is set to be the exit room so that agents all try to move to room 4. In the agent-based model, this is done by assigning the destinations of each of the agent to the locations in room 4. In the graph-based model, this is implemented using the transition table. In the transition table, the probability for the occcupant to travel to a vertex(room) that is nearer to the exit room is set to be 1, this is because in the agent-based model, the agent always tries to move to the final destination and will not try to stay in current vertex, therefore, in graph model, we don't set the agent to choose to stay in current vertex or move to a vertex that is far from the exit room.

We conduct several experiments using the agent-based model and the graph model to compare the result of two simulations. In the this set of experiments, we vary the total number of agents and the number of agents in each room to see whether the results of two type of simulations match. The experiment result is shown as following:

In the first experiment, we set the total number of agents to be 200, number of agents in room 0 and 1 is set to 65, number of agents in room 2 and 3 is set to 35. The result in shown in figure 6.26:



(a) Number of occupants in room 2

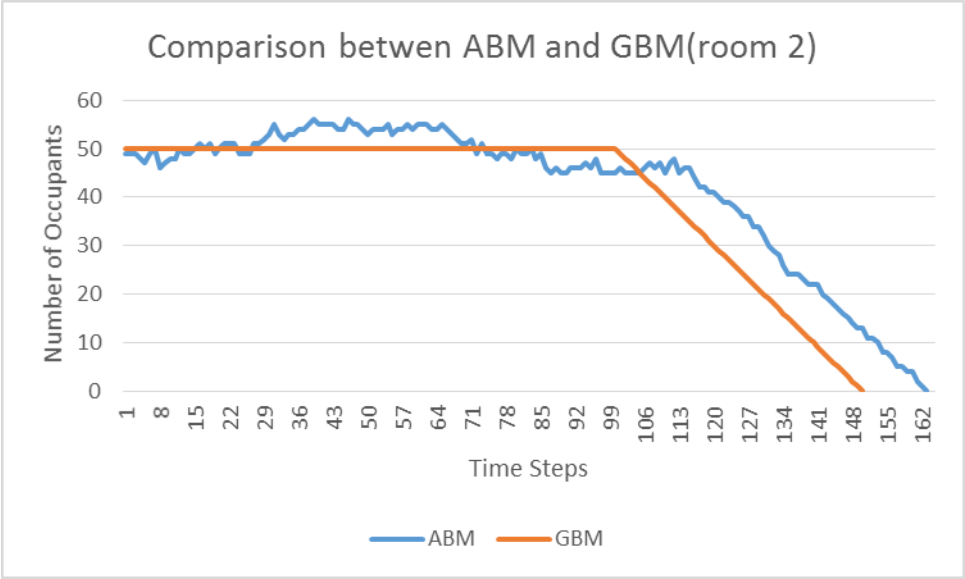


(b) Number of occupants in room 4

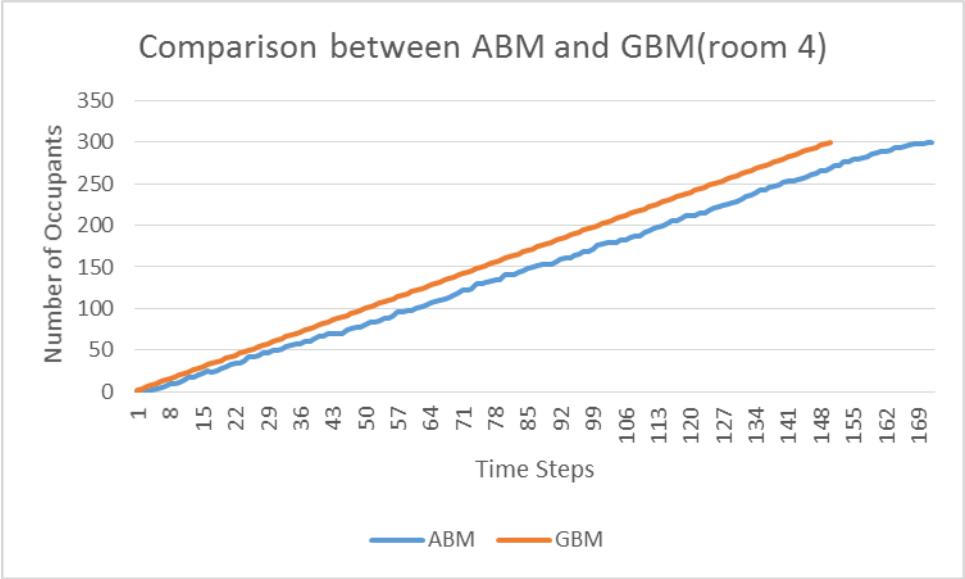
Figure 6.26 Simulation result using 200 agents

The curve for the result of the graph-based model is smooth and linear, this is because that in the definition of the transition table, the probability for occupant to travel to a vertex that is nearer to the exit vertex is set to be 100%, as a result, the evolving process of the model becomes deterministic and the agents simply travel forward towards room 4. In both of the results from agent-based model and graph-based model, the curve shows a stabilized stage where the number of occupants in room 2 does not increase or decrease significantly. This is because that the number of occupants entering the room and the number of occupants exiting the room is equal. The number of occupants then decreases fast when the occupants in room 1 all enter room 2. At this stage, the number of occupants entering room 2 becomes 0 and thus the number of occupants in room 2 tend to decrease. In the experiment, it shows that the results using agent-based model and graph-based model match closely except that the curve of agent-based model contains spikes. There are spikes in agent-based model because the movement of each individual agent contains randomness and thus the time for each individual agent to leave or enter a room is unpredictable. Overall, the comparison between agent-based model and graph-based model shows that the graph model correctly describes the dynamic of occupant movement in a building.

In the second experiment, we set the total number of agents to be 300, number of agents in room 0 and 1 is set to 100, number of agents in room 2 and 3 is set to 50. The result in shown in figure 6.27:



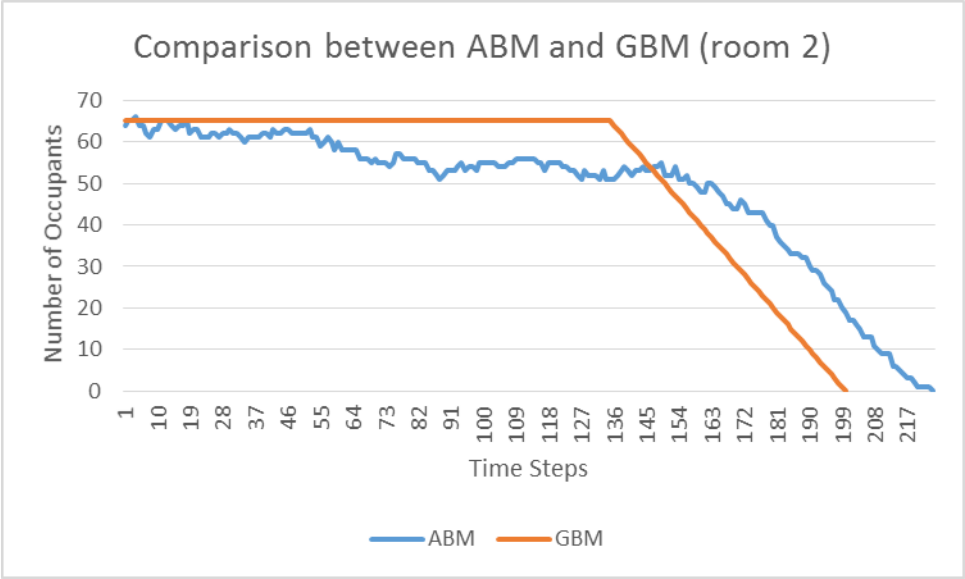
(a) Number of occupants in room 2



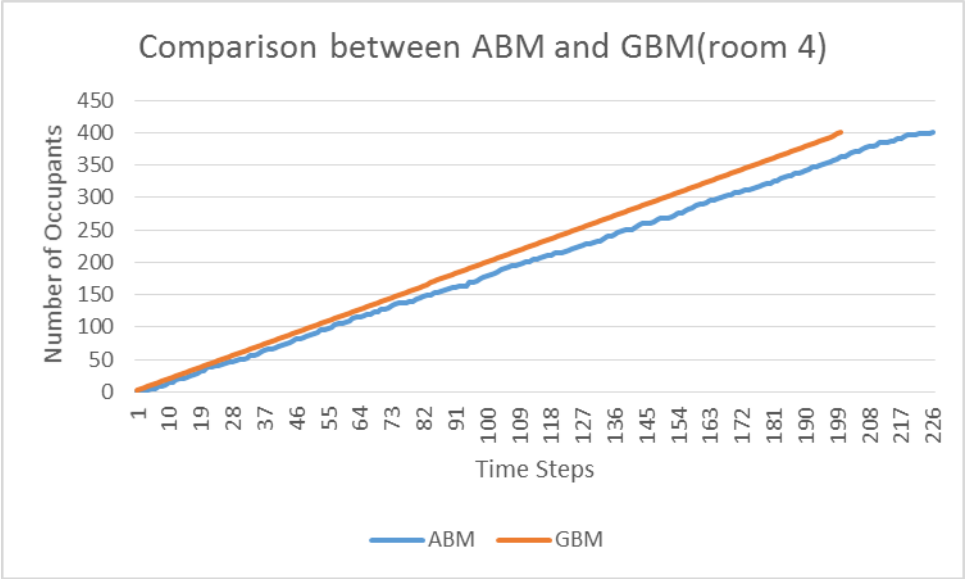
(b) Number of occupants in room 4

Figure 6.27 Simulation result using 300 agents

In the third experiment, we set the total number of agents to be 400, number of agents in room 0 and 1 is set to 135, number of agents in room 2 and 3 is set to 65. The result in shown in figure 6.28:



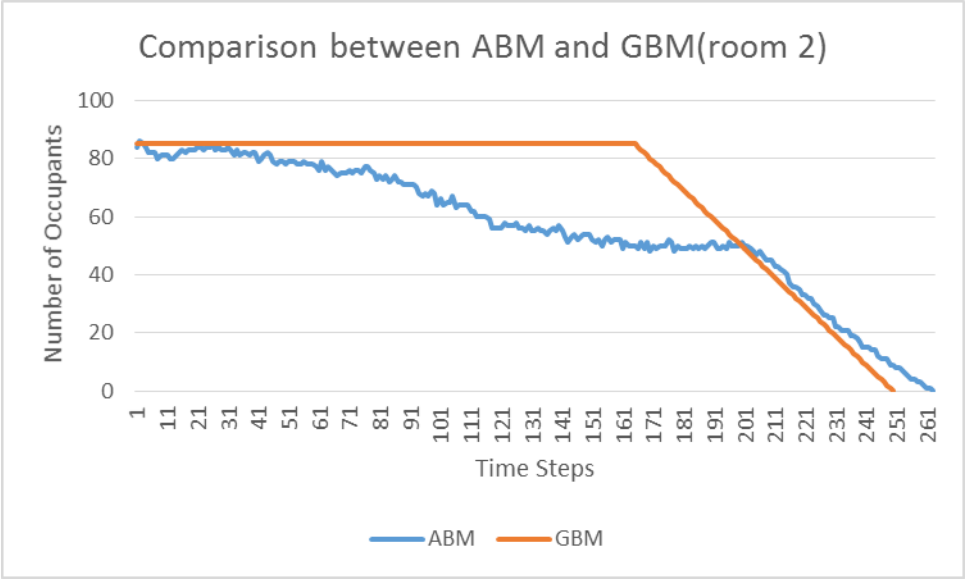
(a) Number of occupants in room 2



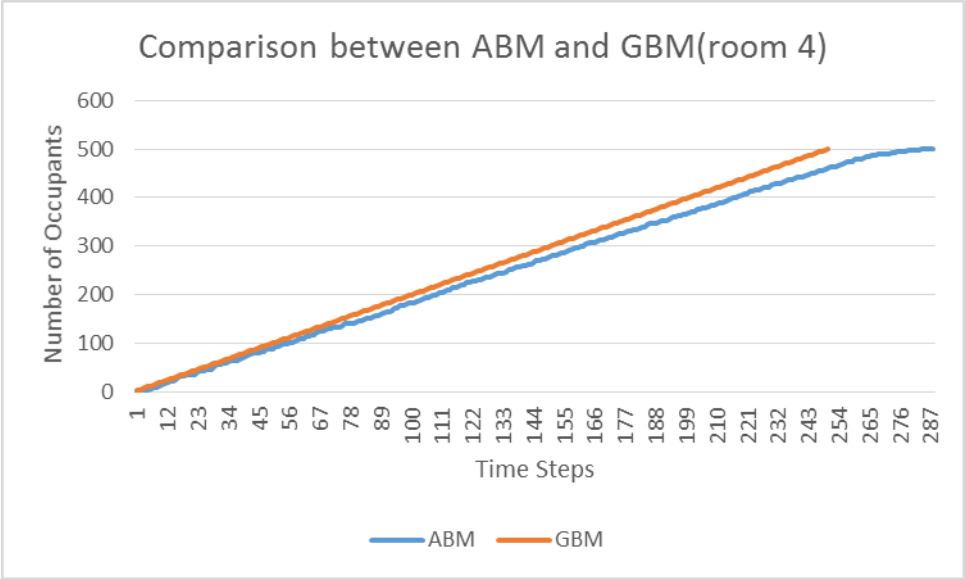
(b) Number of occupants in room 4

Figure 6.28 Simulation result using 400 agents

In the fourth experiment, we set the total number of agents to be 500, number of agents in room 0 and 1 is set to 165, number of agents in room 2 and 3 is set to 85. The result in shown in figure 6.29:



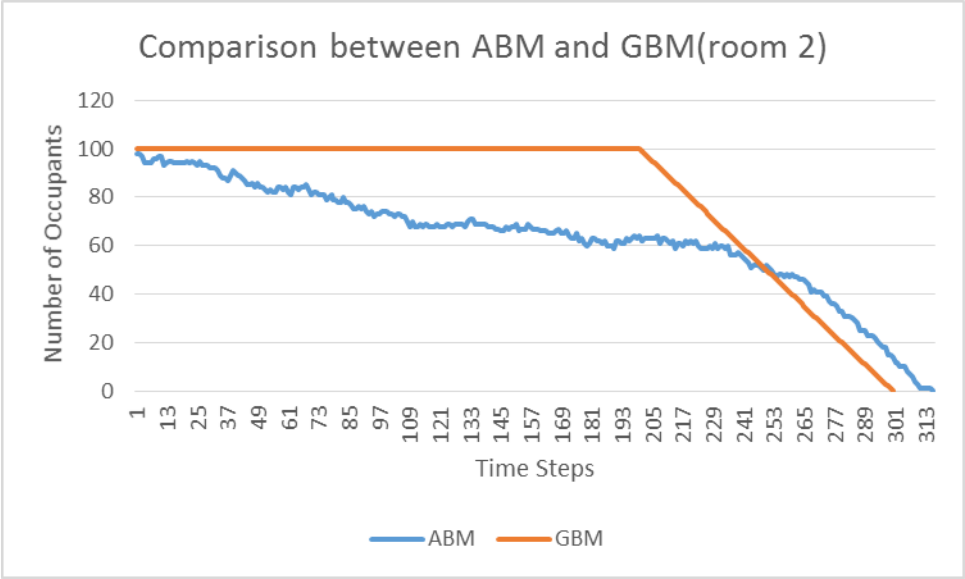
(a) Number of occupants in room 2



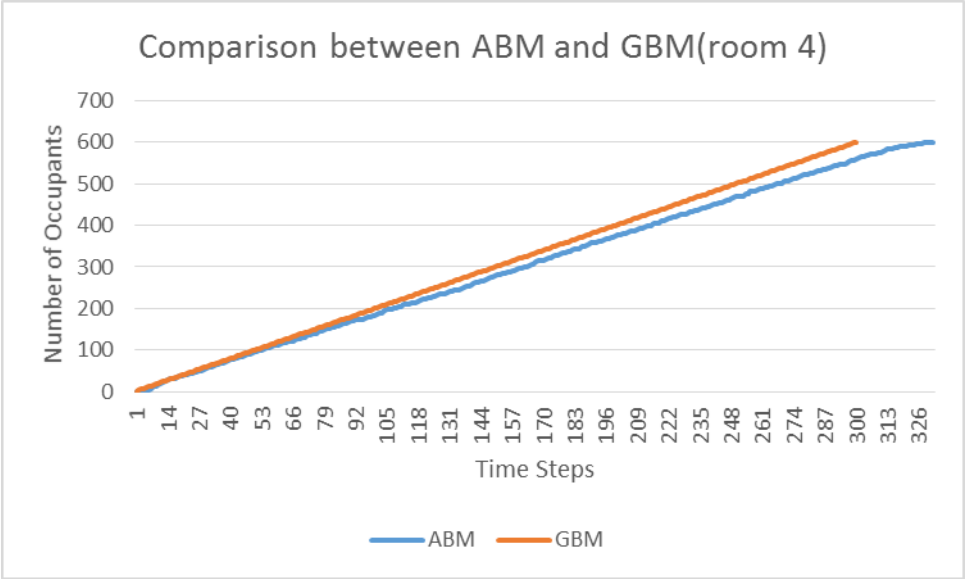
(b) Number of occupants in room 4

Figure 6.29 Simulation result using 500 agents

In the fifth experiment, we set the total number of agents to be 600, number of agents in room 0 and 1 is set to 200, number of agents in room 2 and 3 is set to 100. The result in shown in figure 6.30:



(a) Number of occupants in room 2



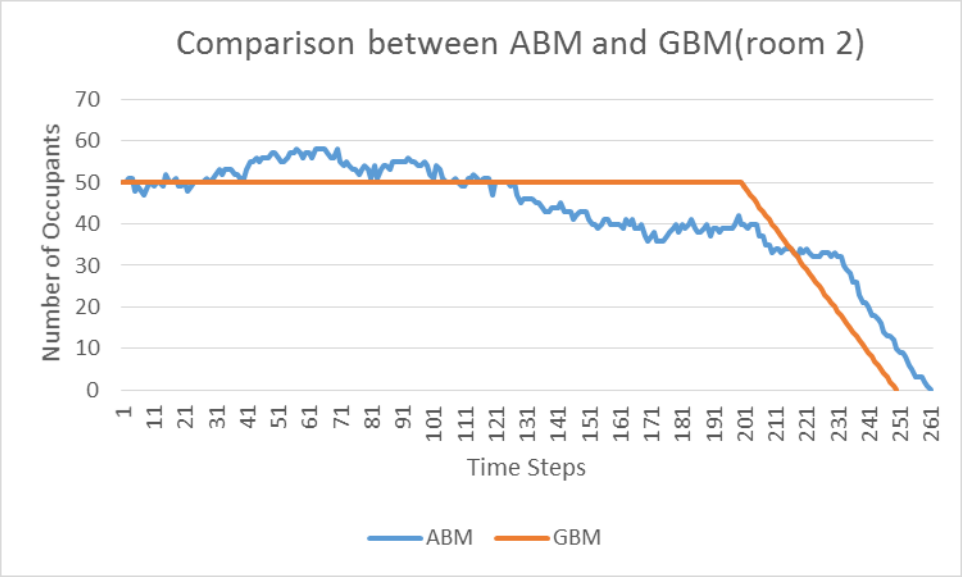
(b) Number of occupants in room 4

Figure 6.30 Simulation result using 600 agents

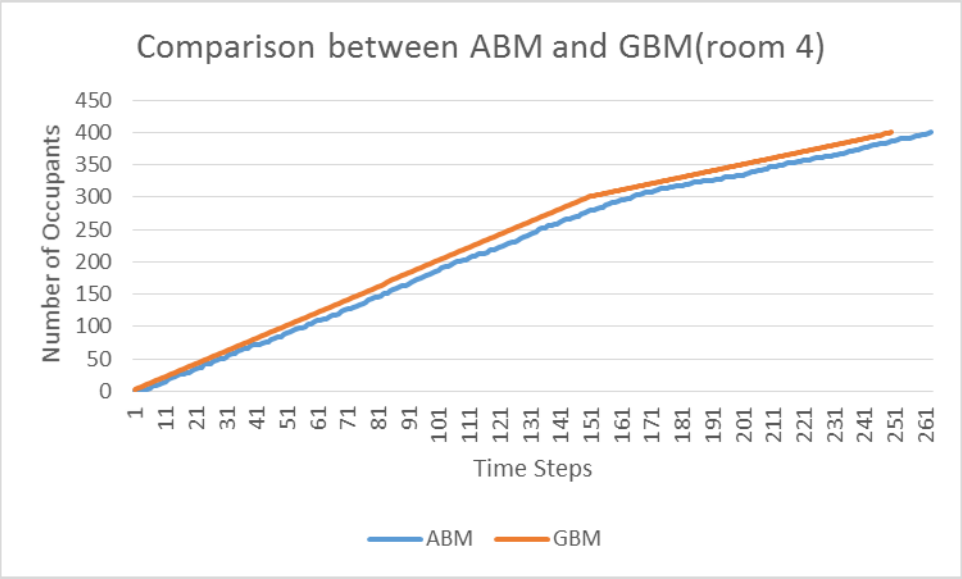
As the number of occupants increases, the degree of matching between curve of agent-based model and the graph-based model also decreases. This is because as the number of occupants in room 2 increases, in the agent-based model, the agents tend to block each other in

the room. Since the agents tend to move in a line while traveling, the agents entering room 2 from room 1 will be blocked by the tail of the move queue of agents. When this happens, the number of occupants entering room 2 becomes 0 and thus the number of occupants in room 2 tend to decrease, as a result, there is no stabilized phase in the curve of results of agent-based models. Consequently, the result of agent-based model and the result of graph-based model does not match in the 500 agents and 600 agents case.

In the sixth experiment, we set the total number of agents to be 400, number of agents in room 0 is set to be 100, number of agents in room 1 is set to 200, number of agents in room 2 and 3 is set to 50. The result in shown in figure 6.31:



(a) Number of occupants in room 2

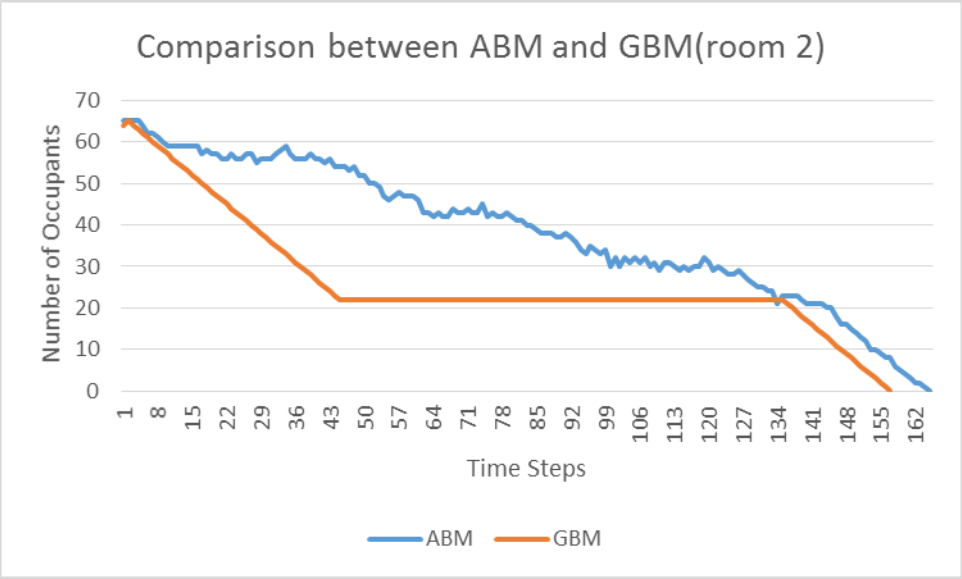


(b) Number of occupants in room 4

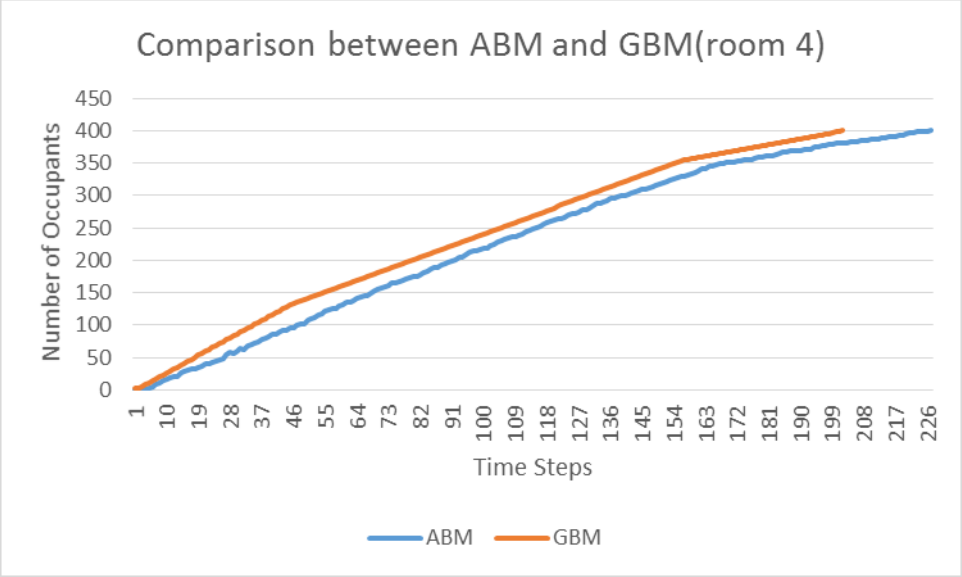
Figure 6.31 Simulation result using 400 agents with a different initial distribution

This experiment shows under a different initial distribution of occupants, the result of agent-based model and the graph model match in some degree.

In the seventh experiment, we change the door width of the door linking room 2 and room 4 to a different value, 95, in agent-based model and change the flow capacity of link 2 to 4 to 2 in graph-based model, we set the total number of agents to be 400, number of agents in room 0 and 1 is set to be 135, number of agents in room 2 and 3 is set to 65, the result is shown in figure 6.32:



(a) Number of occupants in room 2



(b) Number of occupants in room 4

Figure 6.32 Simulation result using 400 agents and varying flow capacity

The results of the agent-based model and the results of the graph-based model do not match in terms of number of occupants in room 2. The result of the graph-based model shows three stages. At the first stage, the number of occupants exiting the room is larger than the

number of occupants entering the room, as a result, the number of occupants in the room tend to decrease. At the second stage, when all of the occupants originally resides in room 2 have finished moving to room 4, the occupants in room 2 are all from room 1. Because the flow capacity of link 1 to 2 is 1, which means there is one agent entering room 2 from room 1 per time step, and the occupants in room 2 are all from room 1, so the frequency for these occupants to arrive at link 2 to 4 is also 1 agent per time step. Therefore, the number of occupants entering the room is equal to the number of occupants exiting the room. At the third stage, when all of the occupants in room 1 have finished entering room 2, the number of occupants entering room 2 becomes 0, as a result, the number of occupants in room 2 tends to decrease. In agent-based model, since the movement of agents are defined using way point graph, agents tend to be congested at the way points because all of the agents tend to move towards the same waypoints, this delayed the movement of agents, and thus the number of occupants in room 2 does not decrease as fast as the graph-based model. Furthermore, although the link distance and speed of occupants in the graph-based model is set to be matching the room width and speed of agents in the agent-based model, because there is congestion blocking the movement of agents in the agent-based model, the actual moving time for an agent to travel from room 1 to room 4 is larger than the value obtained simply by dividing the width of the room by the speed of the agents, thus the number of occupants in room 2 does not decrease as fast as in graph-based model.

From the experiments, we observed several drawbacks of both the agent-based model and the graph-based model. One drawback is that the graph-based model cannot model the effect of congestion of occupants. The congestion affect the occupants movement in two ways: first, when occupants congest at the entering door of the room, the occupants from another room cannot enter the room, even if the capacity of the room is not full; in current graph-based model, this

cannot be handled naturally; second, since there are always congestions in the room, the time required for agent to travel from one door to another in a room is larger than the time obtained by dividing the distance between the two doors by the speed of the agent. This problem can be solved by changing the speed of the agent based on the congestion level of the room. Another drawback is that the agent-based model uses waypoint graph for navigation, since all of the agent tend to move to several way points, it becomes easier for the agents to be congested while moving.

7 DISCUSSIONS AND CONCLUSIONS

7.1 Discussion and conclusion

Occupancy estimation is an important research topic. Occupancy of a building provides valuable information to a variety applications. With occupancy information, it is possible to schedule energy consumption more efficiently, deliver conditional air according to occupants demand, realize what is happening in the building or discover and recognize the activities of the occupants. Occupancy estimation normally involves a procedure of measuring using a set of sensors and combine the data obtained from the sensor with a computer model to produce the best estimate of occupants' positions. The sensors can have different resolutions, a high resolution sensor such as a video camera can produce accurate measurement of occupants' locations but is expensive and intrusive; a low resolution sensor such as gas sensor is cheap and is not subject to privacy issues but only provides ambiguous information about the occupants' locations. Sensors are always subject to ambiguity, imprecision and uncertainty, which makes occupancy estimation method merely based on sensor inappropriate. There is a need to develop computer models to fill the gap between the unreliable sensor data and the real occupancy

distribution. The method that incorporate sensor data into the computer model for estimation and simulation is called data assimilation. Traditionally, data assimilation is widely applied in field of geoscience, oceanic science and weather forecast. There are few literatures about data assimilation applied in agent based simulations. In this dissertation, we develop an agent-based model to simulate the movement dynamic of occupants in a building and apply this model in a data assimilation framework.

Agent-based model models the occupants' movements at a low abstraction level. Most of the details related to the movement of occupants are included in the model. This significantly increases the ability of the model to describe the phenomena, however, because the agent-based models includes all the details, it also adds to the computational complexity of the model. The agent-based model consists two modules of behaviors, one is navigation module, which uses a waypoint graph to navigate the agent to its destination; the other is avoidance module, which uses a set of rules to define the way the agents avoid each other. The avoidance behavior has higher priority than the navigation behavior which means agents will first try to avoid each other before they move towards their destinations. The agent-based model precisely describes the movement behaviors of the occupants.

A data assimilation framework normally consists of three components: a model, a series of observations and a scheme to meld the model and observations together to obtain expected outcome. In this research, the model for the data assimilation framework is the agent-based model we construct. The observations are assumed to be generated from a set of deployed binary proximity sensors. The melding scheme is particle filter algorithm. Particle filters are a set of statistical methods aim at using a number of samples, namely particles, to approximate the posterior distribution of the target system. Because the posterior is represented using samples, it

does not require the target distribution to be of any particular analytic form. This is particularly suitable for occupancy estimation using the agent-based model since the agent-based model cannot be written in analytic form. In applying the particle filter algorithm, the proposal distribution is chosen to be the transition prior, which is the agent based model proposed, as a result, the weight is calculate based on the likelihood of the particles. Using particle filter algorithm and the agent-based model, the positions of occupants can be obtained by recursively incorporating the observations into the model.

The particle filter algorithm has several drawbacks. One of the major drawback is brought by the standard re-sampling procedure where due to the nature of re-sampling, the diversity of the particles reduces over time. Eventually, the particles may end up with being concentrating on one or several points in the state space. This problem get more serious when the dimension of the state space is high. The high dimensionality of state space introduces combinatorial explosion. As a result, the number of particles required to cover the state space so that the algorithm can converge significantly increases. To deal with this problem, we develop another re-sampling method called component set re-sampling. Instead of re-sampling the particles as a whole, we divide the states of the particles into sub states called components and then re-sample from the set formed by each component of the states. By doing this, new state can be generated from the components of the old state and therefore the diversity of the particles is increased. Consequently, the ability of using limited number of particles to represent larger sample space is increased. Experiments show that the component set re-sampling significantly increases the accuracy of the particle filter algorithm even with limited number of particles.

The data assimilation framework using agent-based model and particle filter algorithm can estimate the occupants' locations in a building when the number of occupant is limited. It

does not perform well when there are massive number of occupants in the building. This is because the agent-based model itself is quite complex and adds to the computational complexity of the estimation procedure. To solve this problem, a model that models the occupancy dynamic at a higher abstraction level with less details is needed. In this research, we develop another model to support the data assimilation in the situation that massive occupants are getting involved. This model models the building structure using a graph and models the occupancy dynamics as flows over the graph. Each vertex of the graph represents a room or a segment of the corridors; each link of the graph represents the transition boundary such as doors between two vertices. The movements of each occupant is defined using a transition table. Transition table defines the probability each occupant move from his current location to its neighbor locations. The transition table is a property of the vertex itself rather than a property of the occupant. This model can effectively simulates the occupancy dynamic even the number of occupants is large. It can serve as the transition model in the application of occupancy estimation where large number of occupants get involved.

Several future research directions exist after the work proposed in this dissertation. These include applying the graph model in the data assimilation for estimating occupancy when massive occupants get involved and validating the approach using data collected from real sensors. Just as same as the agent-based model, the graph model can also be used as the transition prior in the particle filter algorithm for estimation purposes. As long as appropriate sensor data is utilized, the data assimilation using graph model can estimate locations of massive number of occupants. This approach finds its application in public places such as malls, sub ways or stadium and is an interesting research direction. Another direction would be validating the approaches using real sensor data. Currently, we use simulated environment and identical

twin experiments to test the methods proposed, there are still challenge on migrating the approach tested in simulated environment to the real environment.

REFERENCES

[1] Caroline M. Clevenger , John Haymaker, THE IMPACT OF THE BUILDING OCCUPANT ON ABSTRACT ENERGY MODELING SIMULATIONS, Report, Stanford

[2] Zheng Yang, Nan Li, Burcin Becerik-Gerber, and Michael Orosz. A Non-Intrusive Occupancy Monitoring System for Demand Driven HVAC Operations, Construction Research Congress 2012

[3] Rashidi, P.; Cook, D.J.; Holder, L.B.; Schmitter-Edgecombe, Maureen, "Discovering Activities to Recognize and Track in a Smart Environment," Knowledge and Data Engineering, IEEE Transactions on , vol.23, no.4, pp.527,539, April 2011doi: 10.1109/TKDE.2010.148

[4] Tomastik, R.; Lin, Y.; Banaszuk, A., "Video-based estimation of building occupancy during emergency egress," American Control Conference, 2008 , vol., no., pp.894,901, 11-13 June 2008

[5] B. VonNieda, D. Maniccia, A. Tweed, An analysis of the energy and cost savings potential of occupancy sensors for commercial lighting systems , Illuminating Engineering Society of North America 2000 Annual Conference: Proceedings, IESNA, New York (2000), pp. 433–459

[6] Claudio Martani, David Lee, Prudence Robinson, Rex Britter, Carlo Ratti, ENERNET: Studying the dynamic relationship between building occupancy and energy consumption, Energy and Buildings, Volume 47, April 2012, Pages 584-591

[7] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. 2010. Occupancy-driven energy management for smart building automation. In Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building

[8] Leon R. Glicksman ,Steven Taub, Energy Efficiency of Occupant Controlled Heating, Ventilating and Air Conditioning Systems for Office Buildings, report, Massachusetts Institute of Technology

[9] Erickson, V.L.; Carreira-Perpinan, M.A.; Cerpa, A.E., "OBSERVE: Occupancy-based system for efficient reduction of HVAC energy," Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on , vol., no., pp.258,269, 12-14 April 2011

[10] Leon R. Glicksman, Steven Taub, Thermal and behavioral modeling of occupant-controlled heating, ventilating and air conditioning systems, Energy and Buildings, Volume 25, Issue 3, 1997, Pages 243-249, ISSN 0378-7788

[11] Susan Mckeever, Juan Ye, Lorcan Coyle, Chris Bleakley, and Simon Dobson. 2010. Activity recognition using temporal evidence theory. J. Ambient Intell. Smart Environ. 2, 3 (August 2010), 253-269.

[12] Wongpatikaseree, K.; Ikeda, M.; Buranarach, M.; Supnithi, T.; Lim, A.O.; Yasuo Tan, "Activity Recognition Using Context-Aware Infrastructure Ontology in Smart Home Domain," Knowledge, Information and Creativity Support Systems (KICSS), 2012 Seventh International Conference on , vol., no., pp.50,57, 8-10 Nov. 2012

[13] Mohamed Tarik Moutacalli, Abdenour Bouzouane, Bruno Bouchard, Unsupervised Activity Recognition using Temporal Data Mining, SMART 2012, The First International Conference on Smart Systems, Devices and Technologies , page 15-20

[14] Castanedo, Federico and Aghajan, Hamid and Kleihorst, Richard, Modeling and Discovering Occupancy Patterns in Sensor Networks Using Latent Dirichlet Allocation,Book chapter, Foundations on Natural and Artificial Computation 2011, 481-490, Springer Berlin Heidelberg.

[15] Shuai Zhang; McClean, S.I.; Scotney, B.W., "Probabilistic Learning From Incomplete Data for Recognition of Activities of Daily Living in Smart Homes," Information Technology in Biomedicine, IEEE Transactions on , vol.16, no.3, pp.454,462, May 2012

[16] I.A. Essa, "Ubiquitous Sensing for Smart and Aware Environments", IEEE Personal Communications, 2000, IEEE

[17] Tomastik, Robert and Narayanan, Satish and Banaszuk, Andrzej and Meyn, Sean , Model-Based Real-Time Estimation of Building Occupancy During Emergency Egress, Pedestrian and Evacuation Dynamics 2008 215-224 Springer Berlin Heidelberg, 2010

[18] J. Page, D. Robinson, N. Morel, J.-L. Scartezzini, A generalised stochastic model for the simulation of occupant presence, Energy and Buildings, Volume 40, Issue 2, 2008, Pages 83-98, ISSN 0378-7788, http://dx.doi.org/10.1016/j.enbuild.2007.01.018.

[19] J. MacGregor Smith, State-dependent queueing models in emergency evacuation networks, Transportation Research Part B: Methodological, Volume 25, Issue 6, December 1991, Pages 373-389, ISSN 0191-2615,

[20] Chenda Liao; Barooah, P., "An integrated approach to occupancy modeling and estimation in commercial buildings," American Control Conference (ACC), 2010 , vol., no., pp.3130,3135, June 30 2010-July 2 201

[21] Xiaoshan Pan, Charles S. Han, Ken Dauber, and Kincho H. Law. 2007. A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. AI Soc. 22, 2 (October 2007), 113-132

[22] Talagrand, O., 1997: Assimilation of observations, an introduction.J. Met.Soc. Japan Special Issue75, 1B, 191-209.

[23] Eugenia Kalnay, Atmospheric Modeling, Data Assimilation and Predictability, Cambridge University Press, 2003

[24] Greg Welch and Gary Bishop. 1995. An Introduction to the Kalman Filter. Technical Report. University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.

[25] A. Doucet, N. De Freitas, and N. Gordon, Sequential Monte Carlo methods in practice. New York: Springer, 2001.

[26] Philippe Smets. 1991. Varieties of ignorance and the need for well-founded theories. Inf. Sci. 57-58 (September 1991), 135-144.

[27] Gilbert, Nigel. Agent-based models. No. 153. Sage, 2008

[28] Schelhorn T, O'Sullivan D, Haklay M, Thurstain-Goodwin M, , ``STREETS: an agent-based pedestrian model", paper presented at the conference Computers in Urban Planning and Urban Management, 1999,Venice

[29] Turner, Alasdair, and Alan Penn. "Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviour in the built environment." Environ Plann B 29.4 (2002): 473-490.

[30] Christakos, C.K. A Simple Pedestrian Simulator Using Node-Edge Graphs for Floorplan Models 38th, Summer computer simulation conference .4; 105-109

[31] W.R. Gilks, C. Berzuini, Following a moving target—Monte Carlo inference for dynamic Bayesian models, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63 (2001) 127-146.

[32] D. Fox, KLD-Sampling: Adaptive Particle Filters and Mobile Robot Localization, in: In Advances in Neural Information Processing Systems. 2001.

[33] S. Särkkä, A. Vehtari, J. Lampinen, Rao-Blackwellized particle filter for multiple target tracking, Information Fusion, 8 (2007) 2-15.

[34] Vishal Garg, N.K. Bansal, Smart occupancy sensors to reduce energy consumption, Energy and Buildings, Volume 32, Issue 1, June 2000, Pages 81-87

[35] Jiakang Lu, Tamim Sookoor, Vijay Srinivasan, Ge Gao, Brian Holben, John Stankovic, Eric Field, and Kamin Whitehouse. 2010. The smart thermostat: using occupancy sensors to save energy in homes. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10). ACM, New York, NY, USA

[36] Antimo Barbato, Luca Borsani, Antonio Capone, and Stefano Melzi. 2009. Home energy saving through a user profiling system based on wireless sensors. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys '09)

[37] Zheng Yang, Nan Li, Burcin Becerik-Gerber, and Michael Orosz. 2012. A multi-sensor based occupancy estimation model for supporting demand driven HVAC operations. In Proceedings of the 2012 Symposium on Simulation for Architecture and Urban Design (SimAUD '12)

[38] Tachwali, Y.; Refai, H.; Fagan, J.E., "Minimizing HVAC Energy Consumption Using a Wireless Sensor Network," Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE , vol., no., pp.439,444, 5-8 Nov. 2007

[39] Bing Dong, Burton Andrews, Khee Poh Lam, Michael Höynck, Rui Zhang, Yun-Shang Chiou, Diego Benitez, An information technology enabled sustainability test-bed (ITEST) for occupancy detection through an environmental sensing network, Energy and Buildings, Volume 42, Issue 7, July 2010, Pages 1038-1046

[40] M. Pidd, F.N. de Silva, R.W. Eglese, A simulation model for emergency evacuation, European Journal of Operational Research, Volume 90, Issue 3, 10 May 1996, Pages 413-419, ISSN 0377-2217

[41] Barnes, M.; Leather, H.; Arvind, D. K., "Emergency Evacuation using Wireless Sensor Networks," Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on , vol., no., pp.851,857, 15-18

[42] Peter A. Thompson, Eric W. Marchant, A computer model for the evacuation of large building populations, Fire Safety Journal, Volume 24, Issue 2, 1995, Pages 131-148

[43] Jianyong Shi, Aizhu Ren, Chi Chen, Agent-based evacuation model of large public buildings under fire conditions, Automation in Construction, Volume 18, Issue 3, May 2009, Pages 338-347

[44] Tzu-Sheng Shen, ESM: a building evacuation simulation model, Building and Environment, Volume 40, Issue 5, May 2005, Pages 671-680

[45] Rashidi, P.; Cook, D.J.; Holder, L.B.; Schmitter-Edgecombe, Maureen, "Discovering Activities to Recognize and Track in a Smart Environment," Knowledge and Data Engineering, IEEE Transactions on , vol.23, no.4, pp.527,539, April 2011

[46] Eunju Kim; Helal, S.; Cook, D., "Human Activity Recognition and Pattern Discovery," Pervasive Computing, IEEE , vol.9, no.1, pp.48,53, Jan.-March 2010

[47] E. Munguia-Tapia, S.S. Intille, and K. Larson, “Activity Recognition in the Home Using Simple and Ubiquitous Sensors,” Proc.2nd Int’l Conf. Pervasive Computing (Pervasive 04), Springer,2004, pp. 158–175.

[48] Turaga, P, Chellappa, R, Subrahmanian ,V. S,. Machine recognition of human activities: A survey[J]. Circuits and Systems for Video Technology, IEEE Transactions on, 2008, 18(11): 1473-1488.

[49] Jiang B. SimPed: simulating pedestrian flows in a virtual urban environment[J]. Journal of geographic information and decision analysis, 1999, 3(1): 21-30.

[50] Beltaief, O.; El Hadouaj, S.; Ghedira, K., "Multi-agent simulation model of pedestrians crowd based on psychological theories," Logistics (LOGISTIQUA), 2011 4th International Conference on , vol., no., pp.150,156

[51] S. Sarmady, F. Haron and A. Z.H.Talib., "Agent-based simulation of crowd at the tawaf area," 2008 Second Asia International Conference on Modelling x0026; Simulation, 2007

[52] Bikramjit Banerjee, Ahmed Abukmail, and Landon Kraemer. 2008. Advancing the Layered Approach to Agent-Based Crowd Simulation. In Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation (PADS '08)

[53] Marcelo C. Toyama, Ana L. C. Bazzan, and Roberto da Silva. 2006. An agent-based simulation of pedestrian dynamics: from lane formation to auditorium evacuation. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS '06)

[54] Qiu Qin; Junhu Wei, "An agent-based approach for crowd dynamics simulation," Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on , vol.2, no., pp.78,82

[55] J.P. Yuan, Z. Fang, Y.C. Wang, S.M. Lo, P. Wang, Integrated network approach of evacuation simulation for large complex buildings, Fire Safety Journal, Volume 44, Issue 2, February 2009, Pages 266-275

[56] Ren-Yong Guo, Hai-Jun Huang, S.C. Wong, Collection, spillback, and dissipation in pedestrian evacuation: A network-based method, Transportation Research Part B: Methodological, Volume 45, Issue 3, March 2011, Pages 490-506

[57] Smith J M G. Application of state-dependent queues to pedestrian/vehicular network design[J]. Operations Research, 1994, 42(3): 414-427.

[58] Rolf H. Reichle, Data assimilation methods in the Earth sciences, Advances in Water Resources, Volume 31, Issue 11, November 2008, Pages 1411-1418

[59] C.J. Hutton, Z. Kapelan, L. Vamvakeridou-Lyroudia, D. Savić, Real-time Data Assimilation in Urban Rainfall-runoff Models, Procedia Engineering, Volume 70, 2014, Pages 843-852

[60] H. Xue, F. Gu, X. Hu, Data Assimilation Using Sequential Monte Carlo Methods in Wildfire Spread Simulation, The ACM Transactions on Modeling and Computer Simulation (TOMACS), Vol. 22, No. 4, Article No. 23, 2012

[61] Balasubramanya T. Nadiga, W. Riley Casper, Philip W. Jones, Ensemble-based global ocean data assimilation, Ocean Modelling, Volume 72, December 2013, Pages 210-230

[62] I. Okeya, Z. Kapelan, C. Hutton, D. Naga, Online Modelling of Water Distribution System Using Data Assimilation, Procedia Engineering, Volume 70, 2014, Pages 1261-1270

[63] Geir Evensen , The Ensemble Kalman Filter: theoretical formulation and practical implementation, Ocean Dynamics, Vol 53, No. 4,2003, 343-367, Springer-Verlag

[64] Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. 2000. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI '00)

[65] Kwok, N.-M.; Gu Fang; Weizhen Zhou, "Evolutionary particle filter: re-sampling from the genetic algorithm perspective," Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on , vol., no., pp.2935,294

[66] Bugallo, M.F.; Ting Lu; Djuric, P.M., "Target Tracking by Multiple Particle Filtering," Aerospace Conference, 2007 IEEE , pp.1,7

[67] Pitt M K, Shephard N. Filtering via simulation: Auxiliary particle filters. Journal of the American statistical association, 1999, 94(446): 590-599.

[68] Van Der Merwe R, Doucet A, De Freitas N, The unscented particle filter, NIPS. 2000: 584-590.

[69] Pulford, G.W., "Taxonomy of multiple target tracking methods," Radar, Sonar and Navigation, IEE Proceedings - , vol.152, no.5, pp.291,304, October 2005

[70] Blackman, S.S., "Multiple hypothesis tracking for multiple target tracking," Aerospace and Electronic Systems Magazine, IEEE , vol.19, no.1, pp.5,18, Jan. 2004

[71] Hue, C.; Le Cadre, J.-P.; Perez, P., "Sequential Monte Carlo methods for multiple target tracking and data fusion," Signal Processing, IEEE Transactions on , vol.50, no.2, pp.309,325, Feb 2002

[72] Fortmann, T.E.; Bar-Shalom, Y.; Scheffe, M., "Multi-target tracking using joint probabilistic data association," Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on , vol.19, pp.807,812, Dec. 1980

[73] K. Zia, T. Balch, and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 27, pp. 1805-1819, 2005.

[74] M. Coates, "Distributed Particle Filters for Sensor Networks," in IN PROC. OF 3ND WORKSHOP ON INFORMATION PROCESSING IN SENSOR NETWORKS (IPSN), 2004, pp. 99--107.

[75] K. Zia, B. T, and D. F, "Efficient Particle Filter-Based Tracking of Multiple Interacting Targets Using an MRF-based Motion Model," presented at the IROS, Las Vegas, 2003.

[76] J. Singh, R. Kumar, U. Madhow, S. Suri, and R. Cagley, "Multiple-Target Tracking With Binary Proximity Sensors," ACM Trans. Sen. Netw., vol. 8, pp. 1-26, 2011.

[77] N. Shrivastava, R. Mudumbai, U. Madhow, and S. Suri, "Target tracking with binary proximity sensors: fundamental limits, minimal descriptions, and algorithms," presented at the Proceedings of the 4th international conference on Embedded networked sensor systems, Boulder, Colorado, USA, 2006.

[78] L. Vu, Q. Do, K. Nahrstedt, Jyotish: A novel framework for constructing predictive model of people movement from joint Wifi/Bluetooth trace, in: Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on.(2011), 54-62

[79] S. Hassan, J. Pavón, L. Antunes, N. Gilbert, Injecting Data into Agent-Based Simulation, in: K. Takadama, C. Cioffi-Revilla, G. Deffuant (Eds.) Simulating Interacting Agents and Social Phenomena. (Springer Japan, 2010), 177-191.

[80] O. Baqueiro, Y. Wang, P. McBurney, F. Coenen, Integrating Data Mining and Agent Based Modeling and Simulation, in: P. Perner (Ed.) Advances in Data Mining. Applications and Theoretical Aspects. (Springer Berlin Heidelberg, 2009), 220-231.

[81] S. Hassan, L. Antunes, J. Pavon, N. Gilbert, Stepping on earth: A roadmap for data-driven agent-based modelling, in: The Centre for Research in Social Simulation. Paper 5. (2008).

[82] M. Altaweel, L. Alessa, A. Kliskey, C. Bone, A framework to structure agent-based modeling data for social-ecological systems, in: Struct. Dynam. eJ. Anthro. Rel. Sci. 2010.

[83] L. Yang, N. Gilbert, Getting away from numbers: Using qualitative observation for agent-based modelling, in: F. Amblard (Ed.) ESSA'07: Fourth Conference of the European Social Simulation Association. (Toulouse, France, 2007), 205-214

[84] D.G. Brown, R. Riolo, D.T. Robinson, M. North, W. R, Spatial process and data models: Toward integration of agent-based models and GIS, Journal of Geographical Systems, (2005) 25-47.

[85] Macal C M, North M J. Tutorial on agent-based modelling and simulation[J]. Journal of Simulation, 2010, 4(3): 151-162.

[86] R. Aylett , M. Cavazza, Intelligent Virtual Environments: a State-of-the-Art Report, in: In: Eurographics 2001, STAR Reports volume. (2001), 87-109