UNSUPERVISED MULTI-RESIDENT TRACKING IN SMART ENVIRONMENTS

By

TINGHUI WANG

A dissertation submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

DECEMBER 2019

© Copyright by TINGHUI WANG, 2019
All Rights Reserved
To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of TINGHUI WANG find it satisfactory and recommend that it be accepted.

________________________________________
Diane J. Cook, Ph.D., Chair

________________________________________
Sandip Roy, Ph.D.

________________________________________
Thomas R. Fischer, Ph.D.
ACKNOWLEDGMENTS

First and foremost, I would like to thank my wife, Junhui Zhao, for her support during my graduate study, and my daughter, Iris Wang, for bringing happiness and laughter into the family everyday. I must also thank my mother, Peiqin She, for her support in my graduate study, and my late father, Zhongfa Wang, for igniting and nurturing my curiosity in engineering and teaching me programming in my early age.

I would like to extend my deepest gratitude to my advisor, Dr. Diane J. Cook, for her invaluable advice, guidance and encouragement during my graduate study, her patience during the time when I was struggling to narrow down the research problem and keeping me focused on my research during the process. I would also like to extend my sincere thanks to my committee, Dr. Thomas Fischer and Dr. Sandip Roy, for the insightful comments and patient explanations to my questions.

Thanks should also go to Brian Thomas, Aaron Crandall, and Bryan Minor for their assistance in data collection, and Sue Nelson for her assistance in providing the ground truth labels.

I would also like to acknowledge the help and support that I received from Clint Cole, John Yates and Rick Hoover. I am also grateful for the teaching opportunity provided by Dr. Partha Pande, Dr. Ananth Kalyanaraman, and Dr. Shira Broschat.
Aging is a global challenge that our society will face in the next few decades. Smart environment and ambient assisted living (AAL) offer promising technologies to help people stay active, socially connected, and independent into older age. Ambient binary sensors, such as PIR motion sensors, magnetic door sensors, and contact-based item sensors, offer a low cost, easy to deploy and unobtrusive solution to constructing a smart environment. However, the limited ability of coping with multiple residents in a smart environment hinders widespread adoption of the AAL and smart environment technologies.

In this dissertation, we investigate the multi-resident tracking problem in smart environments equipped with ambient binary sensors. First, we establish the theoretical foundation of the hypothesis that human daily indoor mobility is predictable. Based on information theory, we derive an upper bound of predictability using sensor events recorded in over 100 smart homes.

In addition, we formulate the multi-resident tracking (MRT) problem in the framework of finite set statistics (FISST) and propose a sensor-vectorized MRT solution, sMRT. In sMRT, the resident movement in the smart environment is mapped to a point target ma-
neuvering in a latent measurement space. The sensors are mapped into the vectors of the measurement by mining the spatio-temporal relation exhibited in the recorded sequence. Furthermore, we propose sMRT-ML by introducing an unsupervised learning procedure to derive the model parameters previously treated as hyper-parameters in sMRT. We also introduce a track consistency optimization procedure to reduce target mismatch errors during tracking.

Our proposed sMRT and sMRT-ML algorithms are evaluated using sensor events collected in two multi-resident smart homes. The ground truth labels are provided by an external annotator using a visualization tool, ActViz, that we specifically designed for this research. Experimental results shows promising performance of sMRT and sMRT-ML in comparison with baseline algorithms which rely on the smart environment floor plans and sensor locations to associate sensor events with residents.
# TABLE OF CONTENTS

| ACKNOWLEDGMENTS | iii |
| ABSTRACT | iv |
| LIST OF TABLES | viii |
| LIST OF FIGURES | ix |

**CHAPTER**

1. Introduction ........................................ 1
   1.1 Dissertation Contributions .......................... 3

2. Related Work ........................................... 7
   2.1 Indoor Wireless Location Tracking .................. 8
   2.2 Indoor Location Tracking with Wearable Inertial Devices .... 17
   2.3 Resident Tracking and Identification with Biometric Information .... 19
   2.4 Vision-based Resident Tracking ....................... 23
   2.5 Resident Tracking with Anonymous Ambient Sensors .............. 24
   2.6 Summary ........................................... 29

3. Background .............................................. 32
   3.1 CASAS Smart Home Framework ....................... 33
   3.2 CASAS Testbeds ....................................... 40
   3.3 CASAS Data Structure ................................ 43
   3.4 Dataset Visualization and Annotation .................. 48
   3.5 Summary ........................................... 54

4. The Predictability of Indoor Human Mobility .................. 56
   4.1 Entropy Rate and Predictability ....................... 57
4.2 Experimental Results and Discussion .................................... 72
4.3 Summary ............................................................................. 82

5. Unsupervised Multi-Resident Tracking via Sensor Vectorization........... 85

5.1 sMRT: sensor-vectorized Multi-Resident Tracking .......................... 86
5.2 sMRT-ML: sMRT with Unsupervised Training ............................... 112
5.3 Track Consistency ................................................................ 117
5.4 Summary ............................................................................. 119

6. Experimental Results and Discussion ........................................ 121

6.1 Baseline Methods: NN-SG and GNN-SG ................................. 122
6.2 Performance Metrics ............................................................. 126
6.3 Evaluation of sMRT ............................................................. 131
6.4 Evaluation of sMRT-ML ........................................................ 141
6.5 Track Consistency ............................................................... 145
6.6 Discussion and Limitations .................................................... 149
6.7 Computational Complexity .................................................... 152

7. Conclusions and Future Work ............................................... 154

REFERENCES ........................................................................... 158
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparison of existing radio-based positioning systems.</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Sensor messages recorded in the TM004 dataset.</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>Sensor sequence extracted from the sensor messages recorded in the TM004 dataset.</td>
<td>46</td>
</tr>
<tr>
<td>3.3</td>
<td>Sensor observations, recorded each time a sensor is activated.</td>
<td>47</td>
</tr>
<tr>
<td>6.1</td>
<td>Multi-label accuracy and Hamming loss of sMRT, NN-sg, and GNN-sg measured using the TM004 dataset.</td>
<td>133</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance of sMRT, NN-sg and GNN-sg using the TM004 dataset, measured based on binary classification accuracy on a per-resident basis.</td>
<td>134</td>
</tr>
<tr>
<td>6.3</td>
<td>Performance of sMRT, NN-sg and GNN-sg using the Kyoto dataset, measured based on binary classification accuracy on a per-resident basis.</td>
<td>135</td>
</tr>
<tr>
<td>6.4</td>
<td>Average error of sMRT, NN-sg and GNN-sg in estimation of the number of active residents in the smart homes.</td>
<td>139</td>
</tr>
<tr>
<td>6.5</td>
<td>MRTA performance of NN-SG, GNN-SG and sMRT.</td>
<td>140</td>
</tr>
<tr>
<td>6.6</td>
<td>Multi-label accuracy and Hamming loss of sMRT-ML measured using the TM004 dataset.</td>
<td>142</td>
</tr>
<tr>
<td>6.7</td>
<td>Performance of sMRT-ML using the TM004 dataset, measured based on binary classification accuracy. Accuracy is calculated on a per-resident basis.</td>
<td>143</td>
</tr>
<tr>
<td>6.8</td>
<td>MRTA performance of sMRT-ML evaluated using the TM004 dataset.</td>
<td>143</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Indoor wireless location tracking with trilateration.</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Indoor wireless location tracking with fingerprinting methods.</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Indoor proximity-based wireless location tracking with RFID.</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Smart home in a box.</td>
<td>34</td>
</tr>
<tr>
<td>3.2 PIR motion sensors with one-wire interface</td>
<td>35</td>
</tr>
<tr>
<td>3.3 Zigbee wireless PIR motion sensors</td>
<td>36</td>
</tr>
<tr>
<td>3.4 Zigbee wireless PIR motion area sensors</td>
<td>36</td>
</tr>
<tr>
<td>3.5 One-wire door sensor with an attached reed switch.</td>
<td>38</td>
</tr>
<tr>
<td>3.6 Zigbee wireless magnetic contact relay, used to sense door activity.</td>
<td>38</td>
</tr>
<tr>
<td>3.7 Contact-based item sensors with a one-wire interface.</td>
<td>39</td>
</tr>
<tr>
<td>3.8 Floor plan and sensor locations in testbed Kyoto</td>
<td>41</td>
</tr>
<tr>
<td>3.9 Floor plan and sensor locations in testbed TM004</td>
<td>42</td>
</tr>
<tr>
<td>3.10 Association between residents and sensor events in TM004.</td>
<td>49</td>
</tr>
<tr>
<td>3.11 ActViz v1.0, a visualization tool for annotation and activity analysis in multi-resident smart environment</td>
<td>51</td>
</tr>
<tr>
<td>3.12 ActViz v2.0, a visualization tool for annotation and activity analysis in multi-resident smart environment</td>
<td>52</td>
</tr>
<tr>
<td>4.1 Entropy rate estimated based on synthetic data generated by a second-order Markov chain.</td>
<td>62</td>
</tr>
<tr>
<td>4.2 Predictability estimated based on synthetic data generated by a second-order Markov chain.</td>
<td>63</td>
</tr>
<tr>
<td>4.3 Entropy rates estimated based on sensor messages recorded in the smart home (TM004).</td>
<td>73</td>
</tr>
</tbody>
</table>
4.4 Upper bounds of predictability estimated based on sensor messages recorded in the smart home (TM004). ......................................................... 74
4.5 Scatter plot of the estimated entropy rate of resident mobility in smart homes ........................................................................................................... 75
4.6 Scatter plot of the estimated predictability of resident mobility in smart homes ................................................................................................. 76
4.7 Example of resident trajectories observed in a single-resident smart home. ........................................................................................................... 77
4.8 Example of resident trajectories observed in a multi-resident smart home. ......................................................................................................... 78
4.9 Comparison of resident mobility predictability with/without multiple residents and with/without pets................................................................. 79
5.1 The generative model of sensor vectorization................................................................. 92
5.2 A example vector mapping of all the sensors in Kyoto testbed into a 3-dimensional measurement space. .......................................................... 94
5.3 The sMRT tracking phase. ......................................................................................... 96
5.4 The propagation pipeline of single-target Bayes filter, multi-target Bayes filter and PHD filter. ................................................................. 99
5.5 The relationship between PHD of resident states and the probability density of each resident state. ................................................................. 102
5.6 The propagation of multi-target PHD in a GM-PHD filter implemented in sMRT to solve the multi-resident tracking problem. ..................... 104
5.7 Parameter learning for dynamic and measurement model by maximizing the likelihood of sensor observations. .................................................. 113
6.1 Sensor adjacency in smart home TM004. ................................................................. 123
6.2 Transition matrix of the sensor graph in TM004. ....................................................... 124
6.3 Accuracy score as a function of the number of active residents for sMRT, NN-sg and GNN-sg using the TM004 dataset. ................................................. 137
6.4 Hamming loss as a function of the number of active residents for sMRT, NN-sg and GNN-sg using the TM004 dataset. ................................................. 138
6.5 MRTA performance versus the minimum length of sensor events............. 144
6.6 MRTA performance versus the number of residents....................... 146

6.7 The effect of track consistency optimization strategy on tracking performance measured in accuracy, Hamming loss and MRTA metrics. ......... 148
Dedication

dedicated to

my loving and supportive wife, Junhui Zhao,

my family, Iris, Grayson, Mason, Milo and Dolly,

and my parents, Peiqin She and Zhongfa Wang
CHAPTER 1. INTRODUCTION

Given recent medical advances in our society, people today are living longer and generally healthier lives. According to the United Nations world population prospect, by 2050, one in six people in the world (16%) and more than one in four people in more developed regions (27%) will be over 65 years old [1]. The aging of the population represents the great achievements of our medical and technological advances. However, at the same time, it poses dramatic challenges to society. Ambient assisted living (AAL), which introduces information and communication technologies to assist with a person’s daily living and working environment, is a promising solution to help people stay active, socially connected, and independent into older ages [2]. By monitoring the daily activities of the residents via sensor networks, AAL environments may acquire the intelligence to recognize the residents’ activities, monitor their well-being, and provide assistance and intervention when needed.

In the past, most research in AAL and smart homes focused on mono-occupant settings, where the smart home or the environment is inhabitant by single individuals. Based on the data collected by the sensors deployed in the AAL environments and smart homes, data-driven or ontology-based methods have been proposed to recognize activities of daily living (ADLs), understand resident intent, forecast their future activities, monitor and assess physical and mental health status, and enable building automation to minimize energy consumption. However, the ability to handle multi-resident scenarios hinders wide-
spread real-life adoption of AAL technology [3].

There are two significant challenges in a multi-resident smart home and AAL environment: resident tracking and resident identification [4]. The objective of resident tracking is to associate data collected by sensors deployed in the environment with the corresponding residents in order to monitor and provide fine-grained location-based services. Resident identification then tries to distinguish residents from one another based on data association identified by the resident tracking process. In past decades, researchers proposed alternative tracking algorithms and sensor technologies to address these multi-resident challenges. Sensors used for multi-resident tracking include a video-based camera system [5], a smart floor [6], passive infrared (PIR) motion sensors [7–11], RFID-based and Wifi-based systems [12], and other ultrasonic systems [13].

However, when deploying a smart home solution in real-life settings, systems with PIR motion sensors rise to the top among the sensor technologies, due to their low cost, convenience in installation and deployment, and unobtrusive nature. Because PIR motion sensors offer an unobtrusive monitoring solution, the data collected by these sensors lack the ability to identify the residents who activate the sensor. In a multi-resident scenario or a single-resident environment with pets and visitors, an algorithm that can successfully re-organize the aggregated sensor events based on the corresponding residents or moving targets (e.g., pets) is an indispensable piece of a smart home solution that is deployable in real-life settings. The data association problem in the multi-resident environment with anonymous binary sensors is a popular research topic that interests many researchers.

Some resident tracking solutions proposed in the literature assume that the number
of residents in the space is constant. However, in reality, the number of residents may change when neighbors, family members, friends or care givers visit. Moreover, if there are pets in the household, the activity of the pets may trigger the ambient sensors, which results in a multi-occupant scenario even in a single-human-resident smart home. Other research addresses the resident tracking problem by taking advantage of additional information, such as annotated labels, physical models of the sensors, environment floor plans and the sensor locations in the environment. However, such information may be impractical or too costly to obtain in real-life deployment. Thus, developing a resident tracking solution that can solve the data association problem solely from the sensor data without expert annotation or additional information would advance the adoption of smart home technology in real life.

In this dissertation, we investigate solutions to the resident tracking problem in multi-resident scenarios, taking into consideration difficulties in real-life deployments. Instead of assuming a constant number of residents in the smart home, or availability of the floor plan, sensor location, or data annotation, our goal is to solve the problem in an unsupervised way, where the residents’ mobility in the smart home can be statistically modeled and learned with unlabelled sensor data collected over a reasonable period of time.

1.1 Dissertation Contributions

With the practical difficulties in the deployment of AAL environment in mind, the work presented in this dissertation proposes a statistical model to solve the multi-resident
tracking problem in AAL environments equipped with anonymous binary sensors. The major contributions of this dissertation include:

1. A mathematical rigorous modeling of the multi-resident tracking problem in smart homes and AAL environments equipped with anonymous binary sensors. Resident states are mapped to a Hilbert latent space and modeled using the multi-target probability hypothesis density (PHD) represented in the form of a Gaussian mixture. Sensor events triggered by smart home residents are approximated with a multi-target PHD Filter.

2. The introduction of an unsupervised sensor vectorization procedure that maps each sensor in the smart home to a vector in some measurement space. With such a mapping, prior information about sensor locations, sensor adjacencies, and labeled sensor event-to-resident associations are no longer required. Sensor vectorization also serves as a starting point for parameter training of the multi-resident dynamic model as well.

3. A multi-resident tracking algorithm based on sensor vectorization (sMRT) that solves the multi-resident tracking problem based on raw sensor data. With the help of sensor vectorization and the probabilistic modeling of the multi-resident tracking problem, the result is competitive with other state-of-art methods.

4. An unsupervised parameter training procedure that enhances the performance of sMRT. The proposed training procedure leverages the generative nature of the statistical multi-resident tracking model, and fits the parameters of the multi-resident
model by maximizing the likelihood of an observed sequence.

5. A set of multi-resident tracking metrics that offer valuable insights into the performance of alternative multi-resident tracking solutions. Rather than treating the data association problem as a traditional multi-class classification, the proposed metrics examine errors attributed to target tracking misses, target mismatches, and false positives.

6. Evaluation of multi-resident tracking algorithms using smart home datasets captured in real-life environments. The sensor event-to-resident association is produced by experienced annotators to serve as the ground truth for evaluation. We also designed a annotation application to help with the annotation of data associations.

7. Analysis of resident mobility in smart homes. This analysis provides a theoretical limit for the accuracy of an algorithm that predicts resident movement in a smart home. The analysis also compares the theoretical limits for multi-resident settings and single-resident settings. The study is based on over 100 smart home datasets collected by the Center for Advanced Studies in Adaptive Systems (CASAS) at Washington State University.

In the remainder of this dissertation, we first offer a comprehensive description of the current state of multi-resident tracking in smart homes, including both sensor-driven approaches and data-driven approaches (Chapter 2). Then, we introduce the smart home technology on which this dissertation is focused, explain the challenges of multi-resident tracking with data collected in real-life deployments and define the terminology used
throughout the dissertation (Chapter 3). To establish the predictability of resident mobility in smart homes, we present our study of the predictability of human mobility in smart home based on entropy rate analysis for over 100 smart homes (Chapter 4). In Chapter 5, we proposed the sMRT algorithm as a method to solve the multi-resident tracking problem in an unsupervised way. We establish the mathematical representation of the multi-resident tracking problem within the framework of Finite Set Statistics (FISST). The sMRT algorithm hypothesizes that smart home resident mobility can be equivalent to a point target maneuvering in a latent space created by the sensor vectorization process. Instead of hypothesizing a constant velocity model in the latent space, sMRT-ML offers a training process to tune the parameters of the dynamic model and measurement model by maximizing the likelihood of the observed sequence. We further conduct a study into different track management methods to improve the performance of multi-resident tracking by grouping identified track segments together according to a set of rule. The multi-resident tracking performance is measured using annotated real-life sensor events recorded from two smart homes. The evaluation results and discussion are presented in Chapter 6. The dissertation is completed with a summary and directions for future research (Chapter 7).
CHAPTER 2. RELATED WORK

The ability to track each resident in the smart home or AAL environment is an essential task for real-life deployment. Such an ability also enables additional personal, location-specific, or context-aware applications. In the past, researchers have shown great interest in the multi-resident tracking problem. Solutions for resident tracking and localization in smart homes are closely related to choices of sensor technology equipped in the smart home and AAL environment. Popular sensor technologies for indoor location tracking involve wireless RF (radio frequency) sensors, wearable sensors, distributed sensors capturing biometric information for each resident at critical locations, vision-based surveillance system, low-cost ambient sensors, and other specialized sensors for indoor tracking. Depending on the nature of the information collected by various sensor technologies, the algorithms and methods used for localization and tracking vary.

In this chapter, we discuss sensor technologies that were adopted in previous research on tracking smart home residents. We examine setup and device requirements of these solutions, the problem statements and the tracking algorithm. We conclude the chapter by discussing the limitations and constraints the solutions will face during deployment in real-life scenarios.
2.1 Indoor Wireless Location Tracking

Indoor wireless location tracking is a common approach to obtaining the location of each resident, where residents are required at all times to carry a wireless radio device, commonly referred to as a target, agent or mobile node. Devices equipped with GPS (Global Positioning System), such as cell phones, can acquire the location of the user based on a world-wide network of dedicated satellites. With proper filtering and noise cancellation, according to Federal Aviation Association (FAA), horizontal tracking accuracy can reach 3 meters (i.e. 6 meters with a 95% confidence interval) for civilian applications [14]. As a result, GPS devices are widely used in outdoor environments and represent one of the most accurate sources of position information worldwide when it is available. However, in indoor applications, the operation of GPS-based tracking devices may not be feasible due to signal obstruction and accuracy requirements for different indoor applications. Thus, alternative methods that rely on local networks have been developed and adopted for indoor location tracking.

Many indoor wireless location tracking solutions leverage existing wireless infrastructures commonly available in the smart environment, such as WiFi (IEEE 802.11), Zigbee (IEEE 802.15.4), Bluetooth or Bluetooth Low Energy (BLE). To offer better resolution and tracking accuracy, specialized RF technologies, such as ultrasound and ultra-wide band (UWB) are also considered in many research work. Within the smart environment, a few anchor nodes (or landmarks), such as Wi-Fi access points (AP), Bluetooth Low Energy (BLE) beacons, RFID tags or other RF transceivers, are deployed. Residents are
required to carry a wireless agent device at all times. In the context of indoor location tracking, the wireless agent device is commonly referred to as target, mobile node or simply agent. Nowadays, there are multiple commercialized solutions for indoor wireless location tracking, such as Estimote, Pointr, and Netvox, to name just a few. The tracking methods can be generally classified into three categories: multilateration (or multiangulation), fingerprinting or proximity.

**Multilateration** is a geometric method to locate the agent node in the smart home, as shown in Figure 2.1. With multiple anchor nodes (usually more than three) deployed
in the environment, the distance between the anchor nodes and the agent are estimated based on RF signal parameters, such as received signal strength (RSS), Time Of Arrival (TOA), Time Difference Of Arrival (TDOA), Angle Of Arrival (AOA), and Phase Difference Of Arrival (PDOA) [15–17]. If three anchor nodes are used, then the method is called trilateration. As a geometric approach, the tracking error of multilateration approaches is dependent on the stability and accuracy of the correlation between distance and RF signal measurements. Obstructions and environmental changes in the path between the transmitter and receiver, commonly referred to as Non-Line-Of-Sight (NLOS), can cause errors in RF signal measurements and distance estimates. Movements of multiple residents may cause further disturbance to the tracking reliability. If the computation is based on angles (e.g. AOA) instead of distance estimates, the method is called multiangulation.

In order to improve tracking accuracy, Bayesian signal processing methods are often adopted to compensate for the errors in signal measurements and environmental uncertainty. These methods include Extended Kalman Filtering (EKF) [18, 19], Unscented Kalman Filtering (UKF) [20], and Particle Filters (PF) [21–23]. Under the framework of Bayes filtering, provided with a dynamic model that predicts data evolvement and the relation between the true state of the data and measurements, the true state of the data can be estimated with a minimum mean square error (MMSE) estimator or maximum a posteriori (MAP) criteria. Both EKF and UKF assume that the posterior state distributions are Gaussian. However, EKF is based on linearization of the non-linear function in the dynamic model, while UKF employs a deterministic sampling approach to parameterize the mean and co-variance matrix of the Gaussian distributions. Alternatively, PF tackles
both nonlinear and non-Gaussian systems by numerically approximating the distribution of target state via recursive sampling.

The early RADAR system developed by Microsoft Research [24] utilizes Wi-fi Wireless Local Area Network (WLAN) for indoor tracking. The Wi-Fi signal propagation model considers signal attenuation factors from both the walls and the floor. The RADAR system claims an accuracy of 2-3 meters in a space the size of a typical office. Li et al. [25] identified that the performance degradation of WiFi-based passive localization can be attributed to multi-path and Non-Line Of Sight (NLOS) propagation, both leading to significant errors in the distance measure between mobile node and anchor nodes. To solve the problem, Channel Impulse Response (CIR) information at the physical layer is extracted using Software Defined Radio (SDR) techniques to mitigate the multipath fading problem. In addition, instead of using the traditional log-distance path loss model, a Non-Linear Regression (NLR) method is used to relate the filtered power information to propagation distance.

Sundar et al. [26] proposed a novel weighted adaptive location estimation (WALE) algorithm by taking into account the quality and properties of the circle overlaps determined by the signal strength between the mobile node and anchor nodes. Using a maximum likelihood estimation over a weighted grid of the region, the WALE system reported an accuracy ranging from 1.2-2.20 meters in majorities cases in a large office space.

In another study, Zhao et al. [23] compared the tracking performance of Wi-Fi and BLE by measuring the signal RSS and computed the target location using a PF. The study finds that the average root-mean square (RMS) error of indoor tracking using BLE with
varying numbers of samples is approximately 3.8 meters compared to 5.3 meters using Wi-Fi. The superior accuracy observed for BLE is due in part to the channel hopping mechanism, low transmission power, and much higher sampling rates compared with Wi-Fi networks.

Other specialized RF systems, such as ultrasound and impulse radio ultra-wideband (UWB) technologies, can improve tracking performance by offering more accurate distance estimates [27]. Ultrasound systems, such as ActiveBat [28], DOLPHIN [29], and Cricket [30], track the target location by measuring the difference between the emitted ultrasonic signal and the received signal. Active Bat and DOLPHIN utilize an array of ultrasound receivers mounted on the ceiling to obtain information from active transmitters attached to devices carried by agents. In contrast, Cricket inverts the configuration with passive ultrasound beacons deployed in the environment and passive receivers carried by the users. Compared with Active Bat and DOLPHIN, Cricket offers greater scalability, ease of deployment, and ability to identify targets, at the cost of higher energy consumption and difficulties with continuous tracking. The Cricket system achieved an accuracy of 2 centimeters in tracking target locations [30]. Kim et al. [13] further lower the system cost and complexity by leveraging the reflection of ultrasound waves, and report a tracking accuracy of 35 centimeters. However, due to ultrasound line-of-sight limitations, these tracking systems struggle in real-life deployments where the line-of-sight between anchor nodes and target may be blocked by furniture or other items in the smart environment.

UWB-based systems also measure TOA to compute the distance between target and anchor nodes. Compared with ultrasound systems, UWB offers better multi-path
resolvability, but does not eliminate non-line-of-sight (NLOS) problems and multi-path propagation [31]. When obstacles exist in the line of site between anchor node and agent, multi-path propagation of RF signal occurs. In these cases, UWB waves need to travel around the obstacle via multiple propagation channels (MPC), producing several meters in tracking error. To resolve this issue, multiple machine learning algorithms, such as decision tree [32], multi-layer perceptron (MLP) [33], or support vector machine (SVM) regression [34], can be used to detect and compensate for NLOS-related error. With the support of machine learning algorithms to distinguish MPC, the root mean squared error of target tracking has gotten as low as 100 centimeters [32].

**Fingerprinting** methods, also known as *mapping* or *scene analysis*, offer an alternative class of approaches to tracking [24,35–37]. In contrast with geometric measurement strategies, fingerprinting methods focus on building a database with scenario features (fingerprints) at the target location, as shown in Figure 2.2. Fingerprinting methods are generally composed of an offline stage and an online stage. During the offline stage, the scenario is surveyed at a known location, and features (i.e. fingerprints) based on RF signal parameters (e.g. RSS) are computed and stored in the database. During the online stage when the mobile node (target) navigates through the environment, features captured by measuring the RF signal parameters of the mobile node are compared against the records in the database. Based on the similarity between the measurements and the stored fingerprints, the approximate target location is determined using proximity matching algorithm, such as k-nearest neighbor (kNN) [38], together with interpolation. Other supervised classification methods, such as Bayes networks [37], support vector machines [39], and neural
Figure 2.2: Indoor wireless location tracking with fingerprinting methods.
networks [40], can also be used to match the online measurements with the stored fingerprints in the database and predict the location of the target.

Proximity methods are the simplest way to provide location information in an indoor environment. The methods only track residents at a set of pre-determined locations. Anchor nodes, such as RFID tags and BLE Beacons, are deployed at the locations of interest in the environment. When a mobile agent is “close” to a pre-determined location, the identity of the resident (mobile agent ID) is captured by the anchor node, or the location (anchor node ID) is detected by the mobile agent. Figure 2.3 illustrates a proximity-based

Figure 2.3: Indoor proximity-based wireless location tracking with RFID.
wireless location tracking system using Radio frequency identification (RFID) technology. The RFID tags, which are passive devices that returns an identifier when approached by an active RFID reader device, are deployed to locations of interest or attached to specific items. The residents in the smart home are equipped with battery-powered RFID readers. The location of the resident is registered when the resident carrying the RFID reader moves into the connectivity range of the RFID tag. Similar arrangements are made with BLE technologies. The locations of the resident carrying a BLE device can be identified via an identifier that is broadcast by a nearby BLE beacon.

Because having the resident carrying a battery-powered RF device could be impractical in real-life deployments, some proximity-based tracking systems adopt an inverse design. Instead of using passive devices, such as RFID tags, as the anchor nodes, the active RFID readers are deployed to pre-determined locations and connected to household electricity. The residents, on the other hand, are tagged with passive RF devices with small form factors, such as RFID tags. This tracking solution offers the least impact to the normal routines of the residents in the environments, at the cost of difficulties and complexity at installation time.

All the RF-based indoor location tracking solutions discussed here require a map of the environment as well as the locations of anchor devices. Each resident needs to carry a mobile device at all times for the system to function. Once there are changes to the environment, a calibration process is required to re-capture or re-train the algorithm parameters. Such inconveniences to the residents, the requirement of calibration at deployment time, the difficulty of system maintenance and the limitations in multi-resident tracking hinder
Technology | Measurements | Accuracy | Features
--- | --- | --- | ---
GPS | TDOA | 6m | Earth scale coverage
Ultrasound | TDOA | 2-35 cm | High accuracy, short range
UWB | TOA/TDOA/AOA | 0.1-1m | High accuracy, short range
Wi-Fi | RSS | 1-5m | Low cost
ZigBee | RSS/PDOA | 1-10 m | Low power, low accuracy
Bluetooth | RSS/PDOA | 1-3 m | Low power, one tag per location
RFID | Proximity | Connectivity range | Low power, one tag per location

Table 2.1: Comparison of existing radio-based positioning systems.

...the wide adoption of RF-based tracking systems in real-world smart environments.

### 2.2 Indoor Location Tracking with Wearable Inertial Devices

Many mobile devices currently incorporate low-power Inertial Measurement Units (IMU) based on Micro-Electro-Mechanical Systems (MEMS) technology. IMUs are microscopic devices that can measure the acceleration, angular rate, and orientation of the target using a combination of accelerometers, gyroscopes, and sometimes magnetometers. Thus, given the initial position of the target, this target can be tracked using self-measurements generated from the IMUs embedded in mobile devices. Early work conducted by Foxlin [41] utilizes shoe-mounted inertial sensors in a device called NavShoe, to improve the tracking...
accuracy of pedestrians walking in outdoor environments. Similar to the NavShoe, Bird and Arden [42] also use data collected by shoe-mounted inertial sensors to track resident movement in indoor environments. With a Kalman filter to process the accumulated accelerometer data and magnetic headings obtained by a gyroscope and a magnetometer, the system is able to track a resident’s indoor location with an average error of 2 meters. The ActionSLAM system proposed by Hardegger et al. [43] improves earlier wearable sensor tracking by introducing heading drift compensation, stance detection adaption, and ellipse landmarks. This approach achieved a positioning error of 0.52 meters in an indoor environment.

Compared with wireless-based location tracking, IMU-based location tracking methods do not require additional pre-deployment effort. However, the target-tracking accuracy of IMU-based systems depends on the frequency and magnitude of acceleration maneuvers, gyro drift rates, and accelerometer precision and accuracy. IMU measurement errors usually accumulate with the time and distance the target travels [42]. Therefore, hybrid tracking solutions that integrate IMU measurements with other ambient tracking solutions are popular for improving tracking performance.

Considering that the IMUs and BLE/Wi-Fi radios are ubiquitously available in mobile phones and wearable devices, Correa et al. [44] fused the self-measurements obtained with a 9-axis IMU with the RSS anchor node measurements at the mobile node using an EKF. The system shows an tracking accuracy below 1 meter during a 20-minute experiment. Zhang et al. [15] proposed a WiBall system that integrates the time-reversal radio with angular velocity and gravity direction collected from an on-board IMU. The WiBall
shows promising results with 1 meter error in 80% of the evaluated cases.

Pham et al. [45] fuses inertial sensor measurements with ambient PIR motion sensor events to localize residents in indoor environments. The PIR motion sensor provides binary information about resident motion in its field of view and the IMU sensors collect motion data from body activity, walking velocity, and heading estimation. Location tracking is handled by a particle filter-based data fusion algorithm. The author claims a root mean square error of 0.53 meters based on their experiments that were conducted in a small apartment.

2.3 Resident Tracking and Identification with Biometric Information

Resident tracking and identification with biometric information offers another approach to achieve room-level resident tracking in smart environments. Biometric sensors are deployed at the boundary between spaces of interest (usually at the entrance of a room), so that when the location of a target changes, he or she will inevitably enter the field of view of the sensor and the biometric information captured by the sensor are used to determine the identity of the resident. Both biometric-based resident tracking / identification systems and proximity-based wireless location tracking systems offer coarse-grained multi-resident location tracking. However, biometric-based resident tracking / identification systems are capable of identifying the nearby resident without carrying a mobile device and are less intrusive.

Early work used smart-floor based sensing systems [6, 46–48] to track and identify
residents based on their weights. The smart floor embedded a layer of load sensors underneath the traditional wooden flooring surface. Though the floor offers a flat surface for residents to walk on, it also causes clutter in the sensor readings. Due to the different body weight of the residents, the identification of the resident can be resolved directly from the pressure readings. To further improve the location precision, Bayes filtering methods and probabilistic data association techniques are used to predict their future movements as well as reduce the measurement noise. However, the installation cost of such systems prevents the wide adoption of smart floors in real-life settings.

Instead of weight, height is another weak biometric measure that can easily obtained with low-cost sensors. Srinivasan et al. [49], utilizes ultrasonic range finders mounted above each doorway, pointed downward to sense the height of people as they walk by. Due to the inherent sensing accuracy, the height of residents needs to be 7 centimeters apart for the ultrasonic sensors to differentiate them reliably. Given the small number of residents usually resides in the same home, the authors leverage the history of height measurements at each doorway to surpass the inherent accuracy of the sensors. In the experiment, the solution achieved 85% identification accuracy. Furthermore, it reduced the height difference required for 99% identification accuracy from 7 centimeters to 3.25 centimeters. The Doorjamb system, proposed by Hnat et al. [50], further improves the height-based tracking and identification system by determining the resident moving direction when crossing a doorway. A single-height measure during doorway crossing cannot measure walking direction. In the Doorjamb system, multiple ultrasonic range finders are combined to produce a continuous stream of height values that can determine the resident height
as well as the moving direction. Experimenting with 3000 manually recorded door way crossings, the Doorjamb system achieved a room-level tracking accuracy of 90% on average, a 7% false detection rate, and 4% misses. Griffiths et al. [51] enrich the Doorjamb system by introducing infrared sensors to facilitate crossing detection and direction detection. The authors performed a six-day in situ study in a ten-room house with two volunteers. The results indicate that the proposed doorway tracking can achieve 99.5% room accuracy on average for controlled settings and 96% room accuracy for in situ settings.

However, because resident identification in the above approaches is based on weak biometric information, such as resident weight and height, these approaches have a higher chance of success within environments where there are only a handful of residents with different biometric features due to age, gender or family role. However, the approaches are likely to fail in environments with a large number of people or residents who exhibit similar biometric features. These approaches are also subject to consistency issues or error based on body position and change of outfits (e.g., shoes).

Instead of height and weight, certain indoor tracking approaches leverage the existing ubiquitous lighting infrastructure. As the wavelength of visible light is measured in nanometers, any macroscopic object can completely block the light and the shadow cast on the floor is essentially a two-dimensional projection of a 3D object. Thus, the shadow of the residents captured by low-cost photodiodes embedded in the floor can be used to track and identify residents in the smart environment. Li et. al [52] first prototyped the LiSense system and reconstructed a 3D human skeleton from the captured shadow data to infer the activity of the resident.
Kalyanaraman et al. [53], built a proof-of-concept system called FormaTrack using monostatic pulse radar. The radar measures total reflected energy at a given distance and produces a single-dimension measurement vector. When the sensor is mounted at the top of a door frame pointed downwards and a resident walks through the doorway, different levels of reflected energy from different resident body parts are captured by the sensor. The sensor measurements are combined with a Multi-Hypothesis Tracker (MHT) to offer room-level tracking in a smart environment with an accuracy of 90%.

The human walking gait is a complex mechanism. Gait contains a wealth of information for resident tracking and identification. In 2001, Geisheimer et al. [54] utilized fully coherent continuous-wave radar operating near 10.5GHz to record the radar signature corresponding to human walking gait. With the help of a short-term Fourier Transform (STFT) and a Chirplet Transform, the walking gait signature can be used to identify a person under all illumination circumstances [55]. Kalgaonkar et al. [56] proposed a low-cost acoustic Doppler sensor for tracking and identification of walking residents in the space. With the features computed from a Frequency Modulation (FM) demodulated signal, the system achieved an identification accuracy of 92% and an accuracy 96% for determining the walking direction.

Similarly, Hsu et al. [57] designed a RF-based WiGait sensor for monitoring gait velocity and stride length. The WiGait is an extension to the 3D tracking system WiTrack [58] and infers a person’s gait by analyzing the same RF signals reflected off his or her body. The accuracy of gate velocity in the presence of activity-based motion achieved by WiGait is above 96% in all 18 trials, and the accuracy of stride length estimation is over 85%. WiFi
signals are also attempted in the WiFiU system proposed by Wang et al. [59] to capture the human walking gait signature in smart environments. However, in order for the system to function, the users must walk on a predefined path in a predefined walking direction in a confined space, such as a corridor or a narrow entrance. The presence of multiple users walking at the same time will impact the tracking and identification accuracy of the system.

2.4 Vision-based Resident Tracking

Vision-based multi-resident tracking through surveillance systems and closed circuit television cameras (CCTV) offer high-fidelity information for numerous environmental intelligence applications including resident tracking. In the past, tracking multiple objects within a video has been extensively studied. However, mapping the target movement into real-world coordinates remains a challenging problem.

Yu et al. [60] fuse the video from a multi-camera system with a sensory floor via their Condensation algorithm to track multiple residents in the smart home. In their system, the cameras are calibrated such that real-world coordinates can be mapped to the image directly based on a linear transformation. The authors implement a Condensation algorithm to merge the tracking results from the camera localization and sensory floor localization. They conduct an experiment in a lab environment with 4 CCD cameras and sensor floor. The result shows that 78.8% tracking errors are less than 30 cm.

Muñoz-Salinas et al. [61] propose a particle filter-based approach for indoor tracking
of multiple people using multiple heterogeneous cameras. The solution combines both monocular and binocular depth cameras. The proposed method relies on the use of a geometric 3D model representing the silhouettes of people. All the cameras are calibrated to map the image into the 3D room model. A particle filter is used to derive the location of the user on the floor plan of the room. The authors claim that an tracking error of less than 0.1m is achieved in all scenarios.

However, vision-based tracking solutions commonly face difficult challenges for real-life deployments. Oftentimes, residents do not want cameras in their home due to privacy implications, both perceived and real. Another factor is the cost of camera surveillance systems, which may limit the use of this approach in private residences. Even when these considerations are not pertinent, video-based tracking approaches can face technological limitations. They are not uniformly accurate across multiple lighting conditions and differing levels of occlusions. Furthermore, the processing time that is required to track residents with video can be prohibitive.

2.5 Resident Tracking with Anonymous Ambient Sensors

Ambient sensors, which include Passive Infra-Red (PIR) motion sensors, magnetic door sensors, temperature/light sensors, and contact-based item sensors, offer a low-cost and less-intrusive solution for smart environment applications. Past research has shown that, with the integration of artificial intelligence and data analytics algorithms, the data collected by these ubiquitous and easy-to-deploy sensors can power a wide range of smart
environment applications such as recognition of activities of daily living (ADLs) [62–66], building automation [67–71], health monitoring [72–78] and home security [79–81]. However, these ambient sensors lack the ability to identify residents. In multi-resident scenarios, the sensor event stream is composed of the data collected by all the ambient sensors deployed in the smart environment, organized in chronological order. The difficulty in resolving the data association between the sensor event and the corresponding resident hinders the wide adoption of smart environment technology in real-life applications.

In smart environments with passive ambient sensors, the resident tracking problem is usually formulated as a multi-class classification problem between sensor events and the residents. The tracking approaches proposed in the literature vary depending on the assumptions of information availability. Some work assumes that the floor plan and location of sensors deployed in the smart homes are readily available. Other work assumes that the activity and motion model of each resident or all residents in the smart home can be constructed using annotated data or through controlled experiments. The multi-resident tracking problem can be greatly simplified if the number of residents in the smart home is known a priori. In this section, we provide a discussion of each of these research endeavors.

Early work of Wilson and Atkeson [11] focused on activity recognition in multi-resident smart homes, and proposed the simultaneous tracking and activity recognition (STAR) problem. Their proposed solution to the STAR problem is a data-driven statistical model, trained with supervised learning methods, to infer the association between sensor event and residents, as well as the activity each resident is performing. Within the
formulation, the number of residents in the smart home is specified a priori and remains constant. Each sensor event in the training data is labeled with the activities performed by each resident at the moment. In their work, a Hidden Markov Model (HMM) is constructed to model the statistical dependency within the hidden states, a combination of activities of each resident and their locations, and between the hidden states and the observable state, the sensor event captured by the deployed sensors in the smart home. The parameters of the HMM model are estimated by maximizing the likelihood of the annotated sensor events in the training data. During the tracking phase, the activity recognition and resident tracking is equivalent to the HMM inference problem and is solved using a Rao-Backwellised particle filter.

Given the foundation laid by Wilson and Atkeson, Hsu et al. [10] proposed a solution to STAR problem based on a Conditional Random Field (CRF). In their solution, three CRFs are constructed based on the annotated training data, modeling the conditional probability of (a) resident activities dependent on sensor observations, (b) resident location dependent on sensor observation and resident activity, and (c) resident activity dependent on sensor observation and resident location. The parameter of each CRF can be trained using annotated sensor events. During the activity recognition and tracking phase, the inference processes of CRF (b) and CRF (c) are invoked iteratively, with CRF (a) providing the initialization of the iterative inference. Tested with sensor events collected in scripted dual-resident environment, the proposed method achieved 81% activity recognition accuracy if a sensor event-to-resident association is provided. Activity recognition accuracy is decreased to 64% with an 72% data association accuracy using the iterative
training approach.

Assuming the number of residents in the smart home is a constant, Crandall and Cook [82,83] consider the data association between sensor events and smart home residents as a multi-class classification problem. Thus, based on data with ground truth labels, a naive Bayes classifier and a Markov model classifier are both trained to predict the associated resident with a series of sensor events as the input. Their work concludes that subtle differences exist and can be learned using supervised learning algorithms to identify associated residents.

In real-life settings, the number of residents who are actively performing daily activities in the smart home is not a constant. A family friend, relative or caregiver may visit, leading to an increase of the number of residents in the smart home than previously assumed. On the other hand, one resident may take a nap somewhere in the smart home and remain undetected for a period of time, and thus the number of active residents in the smart home decreases. In order to cope with the varying number of active residents, other research focuses on constructing a model of resident dynamics in the smart home. A sensor graph [7], also referred to as a Bayes updating graph [9], or accessibility graph [84], is a standard graph model that captures the resident movement information in the smart home. In the graph, the nodes are mapped to the sensors deployed in the smart home. The sensors that are physically adjacent to each other in the smart home are connected, and a weight can be assigned to each edge representing the likelihood of the resident moving from one sensor location to the other. The sensor adjacency information can be collected from the floor plan and the location of the sensors in the smart home by conducting a
controlled experiment in the smart home. The weights, however, can be estimated using annotated data or maximizing the likelihood of a recorded sensor event stream [84].

Crandall and Cook [9] proposed a rule-based multi-resident tracker. The rule-based multi-resident tracker assumes a strict one-to-one mapping between sensor event and residents in the smart environment. A resident can only trigger two adjacent sensors consecutively. If a sensor event is not associated with any existing resident, a new resident is assumed to enter the smart environment or become active. If a resident does not trigger any sensor event for a certain duration, the resident is considered to have left the environment or become inactive. Based on those rules, the proposed rule-based tracker correctly classified 44% of sensor events of a multi-resident smart environment recorded in real-life settings.

Muller and Hein [7] combined the sensor graph with a multi-hypothesis tracking (MHT) algorithm [85] to infer the location of each resident in the smart home. The sensor graph is constructed using expert knowledge about the floor plan and sensor layout of the environment, with weighted computed using annotated training data. Instead of evaluating on a per-sensor-event basis, the tracking accuracy is measured based on reasonable time frames. The SG-MHT algorithm accurately tracks residents 90% of the time. However, due to the difference in performance metrics, the result is not directly comparable to other solutions.

In addition to sensor adjacency, a detailed model of the field of view (FoV) of each sensor with respect to the floor plan of the smart home can provide valuable information to solve the data association problem. Amri et al. [86] employ square boxes to model the
coverage of motion sensors on the floor plan, and formulate the data association problem within a set-membership estimation framework. Song and Wang [8] introduce a unit disk graph to represent the FoV of each motion sensor, and propose a multi-color particle filter to associate sensor events with the residents.

Additionally, De et al. [87] and Wang et al. [88] proposed the idea of mining possible trajectories of smart home residents directly from the recorded sensor events. Each trajectory is a short sequence of sensor events that may be triggered by a resident consecutively. During the tracking phase, various data association hypotheses are created by fitting the mined trajectories to the incoming sensor events. The best hypothesis is chosen so that the average velocity variance is minimized. However, in order to calculate the velocity variance, the distance between any adjacent sensors are required. The algorithm performs better if the number of residents is known during the trajectory mining process.

2.6 Summary

In this chapter, we examined various approaches towards indoor location tracking in multi-resident scenarios. Indoor geometric-based tracking systems generally require a map or floor plan of the environment and the location of anchor nodes. Tracking accuracy relies on geometric measurements derived from RF signal parameters. Because the geometric measurements are computed from signal parameter measurements according to a RF physical channel model, calibration procedures are required to derive the model parameters at the time of initial deployment or when there is a change in the environment
(e.g., a rearrangement of furniture). Additionally, the different mobile nodes may have different RF signal characteristics. As a result, multiple sets of parameters need to be determined when multiple mobile nodes are used in the system.

Comparatively, the fingerprinting methods do not require any prior knowledge about environment and anchor node locations. Furthermore, they do not rely on any physical model of the RF communication channel. However, reliance of these methods on the training process is a clear drawback. Both feature extraction and classifier training can be costly processes. Moreover, the database and classifier require regular updates to compensate for the change in the scenario. In multi-resident settings, noise introduced by other residents moving in the same environment introduces yet another source of tracking errors.

In addition, RF-based indoor tracking solutions require attaching tracking devices, including mobile phones or smart watches, to smart home residents. In these cases, the residents are responsible for correctly wearing the devices at all times and they cannot share their devices with other residents. Those additional constraints on the residents are usually inconvenient in practice and not reliable in real-life deployment. Moreover, such user-specific sensor devices are tailored toward monitoring one person’s movements rather than all of the activities that occur within the space, which represents valuable information for recognition and analysis of daily activities.

Vision-based methods offer rich information that can be used to recognize the resident in the video as well as infer the activity that is being performed. However, in addition to facing challenges with lighting and obstruction, cameras are often considered too intrusive to be used in homes due to privacy concerns.
Ambient sensors, which include motion sensors, door sensors, temperature/light sensors, and contact-based item sensors, offer a low-cost and unobtrusive solution for smart home applications. As the data collected by these sensors are not associated with any specific resident, the data association problem presents a major challenge when multiple people are present in the smart home at the same time. The unsupervised multi-resident tracking algorithms proposed in this dissertation are designed to target these scenarios. In the next chapter, we provide a detailed description of the smart home technologies that are used to evaluate the proposed multi-target tracking solutions.
CHAPTER 3. BACKGROUND

The unsupervised multi-resident tracking algorithms introduced in this dissertation are designed to operate in a smart environment that is equipped with passive ambient sensors. These passive ambient sensors offer a low-cost, less-intrusive solution to monitoring resident activities in smart homes. They have proved to be sufficient for many applications such as activity recognition, activity forecasting, building automation, and health monitoring. The Center of Advanced Studies in Adaptive Systems (CASAS) group at Washington State University (WSU) has deployed smart home testbeds. In this smart homes, sensor data are continuously collected while residents perform normal routines. The types of sensors that are deployed in these testbeds include PIR motion sensors, magnetic door sensors, item presence sensors based on contact pads, power meters, water flow meters, ambient light sensors, and ambient temperature sensors. The water flow meters, power meters, light sensors, and temperature sensors offer numerical measurements of the environment. In contrast, PIR motion sensors, magnetic door sensors, and item presence sensors generate binary readings (e.g., ON/OFF, OPEN/CLOSED, ABSENT/PRESENT) that are generally triggered by resident activities. The data association of sensor events generated by PIR motion sensors, door sensors, and item sensors are the focus of the methods discussed in this dissertation.

In this chapter, we start with an introduction to the smart home sensor technology and service architecture by the WSU CASAS group. To evaluate the performance of multi-
resident tracking algorithms, we analyze sensor events collected from two smart home testbeds, Kyoto and TM004. The ground truth data association labels for the recorded sensor events are produced by experienced annotators with a visualization application, ActViz, developed for illustrating the past trajectories for each resident. The chapter ends with a discussion on the challenges of multi-resident tracking in smart environments with ambient sensors and formulates the goal of this research.

3.1 CASAS Smart Home Framework

The CASAS group at WSU focuses on smart home technologies with less-intrusive, low-cost ambient sensors that are readily available in the commercial market. The installation and deployment of the sensors to create a fully-functional smart environment is a difficult task, which prevents adoption in real-life settings. As a result, many early smart home projects, such as MavHome, are tested on simulated or lab-based data, and the availability of data collected in the real-life environment are limited [89]. To facilitate the scalability of smart home technology, CASAS designed a smart home kit, named Smart Home in a Box (SHiB). The SHiB (as shown in Figure 3.1), contains light-weight, battery powered ambient sensors, together with a server with an ITX form factor. The SHiB offers a portable and easy-to-install solution that is ready to monitor user activities and perform key capabilities right out of the box. Over the past decade, CASAS group has deployed the SHiB to over 140 smart environment sites.

The PIR motion sensors are a widely-used sensor technology to detect whether a
Figure 3.1: Smart home in a box.
Figure 3.2: PIR motion sensors with one-wire interface. Depending on the additional blocking of the lens, the motion sensor can be configured to concentrate on a small region or monitor the “presence” of residents in a wide area.

A resident has moved in or out of the sensor range. The PIR motion sensors are usually made of a pyroelectric sensor which can detect the infra-red radiation levels. The pyroelectric sensor inside a PIR motion sensor is actually divided into halves. When a resident is moved into the focus of view of a PIR motion sensor, one half will detect more or less IR radiation than the other, leading the output voltage to swing high or low, indicating that human motion has been detected. Once motion is detected, the low-power microcontroller on the board will send a sensor message to the CASAS smart home gateway via the communication link. The sensor message eventually gets relayed to the smart home database residing in the cloud.
Figure 3.3: Zigbee wireless PIR motion sensors. The lens is partially blocked to detect motion in a small region (approximately 4’ × 4’).

Figure 3.4: Zigbee wireless PIR motion area sensors. These wall-mounted motion area sensors contain an unblocked lens and act as an “occupancy” sensor for a large region.
In the early designs of CASAS smart homes, the Dallas one-wire bus motion sensors were used, as shown in Figure 3.2. More recently, the motion sensors were provided by Control4, communicating with each other using a Zigbee protocol. The Control4 motion sensors are powered by a 9V battery with a battery life of more than one year on average.

In the CASAS smart home testbeds, two kinds of PIR motion sensors are deployed. The first is a downward-facing motion sensor, usually installed on the ceiling. With a partially blocked lens, the downward-facing motion sensors are sensitive to resident activities within a 4’ × 4’ space underneath it, as shown in Figure 3.3, providing an accurate measure of a resident presence at a specific location.

Because the downward-facing motion sensors may not cover the entire smart home, they may not be sufficient to detect all resident activities. To fill gaps between these sensors, area motion sensors, shown in Figure 3.4, are also installed. The area motion sensors are fitted with a lens so that it can monitor the resident activity within a wide area, and pick up the movement of residents when they are out of the FoV of the downward-facing motion sensors.

The opening and closing of doors as the resident enter or leave a room is captured with simple magnet-driven switches attached to contact relays. Figure 3.5 shows the hardware for the magnetic door switch that was deployed in earlier CASAS smart homes, and the updated Zigbee-based wireless magnetic door sensor manufactured by Control4 is shown in Figure 3.6. When the magnet moves away from the reed switch as the door opens or moves back to the reed switch as the door closes, the state of the reed switch changes, causing a voltage swing at the output. The microcontroller then detects the voltage and
Figure 3.5: One-wire door sensor with an attached reed switch.

Figure 3.6: Zigbee wireless magnetic contact relay, used to sense door activity.

sends out an “OPEN” message if the magnet moves away, or a “CLOSE” message when the
door is closed. The magnetic door sensors can also be mounted to kitchen cabinets doors
and refrigerators to provide essential information for resident activities such as “cooking”
or “meal preparation”.

Aside from the motion sensors and magnetic door sensors, contact-based item sensors
are sometimes deployed in CASAS smart homes as well to capture the interaction between
residents and items of interest. Figure 3.7 shows a one-wire contact-based item sensor deployed in an earlier testbed, named Kyoto. A few of these contact-based item sensors are installed in the kitchen cabinets for cups and medicine bottles. Those sensors are wired in series with the cabinet door sensors to capture information about activities such as “taking medicine”.

The generated sensor messages are transmitted via a one-wire network (in earlier CASAS smart homes) or a Zigbee wireless mesh network (in more recent homes) to the smart home gateway server. The manager and messenger services on the gateway add time stamps to the sensor messages, assign universally-unique identifiers (UUID) to each message, and maintain site-wide sensor states. The sensor messages appended with the timestamp and UUID are then relayed to the publish/subscribe service via Advanced Message Queuing Protocol (AMQP) protocol. A CASAS database worker in the cloud will listen to the messages published to the publish/subscribe service and store them in a relational database.
3.2 CASAS Testbeds

To illustrate and evaluate the multi-resident tracking methods, we utilize two multi-resident smart home datasets, Kyoto and TM004. The Kyoto testbed is one of the primary research facilities for the CASAS project. The testbed is a two-story town house with three bedrooms and a large downstairs living room. The floor plan and the sensor layout of the Kyoto testbed is shown in Figure 3.8. The testbed is designed to house two full-time undergraduate students. Each resident typically has their own bedroom with a bed, desk, and closet. There is a shared bathroom upstairs and shared living room and kitchen downstairs. Occasionally, their friends will come to visit and even stay for a few days.

The Kyoto testbed is designed to be a sensor-rich space that can capture as many resident movements and activities as possible. A total of 91 sensors are installed in the testbed. PIR motion sensor grids are installed in the bedrooms, bathroom, kitchen, dining room and living room. There are multiple motion sensors deployed in the hallways. Furthermore, magnetic door sensors are positioned on the front and back exterior doors as well as on cabinets, closets and refrigerator doors. A few items of interest are equipped with contact-based item sensors where information about resident interaction with these items can provide insights into the activity they are performing. Deployed initially in 2007, sensors based on one-wire communications are also installed.

The TM004 testbed was deployed in the fall of 2016. It is the most recent smart home dataset with ground truth annotation for multi-resident tracking. The TM004 testbed is a two-bedroom single family house. The floor plan and sensor layout are shown in Figure
Figure 3.8: Floor plan and sensor location in Kyoto. The Kyoto smart home contains 65 motion sensors, 15 door sensors, and 11 item sensors. Motion sensors whose IDs start with “MA” are fitted with a lens that is responsive to resident motion in a wider area. Motion sensors whose IDs start with “M” are only sensitive to a small calibrated area. Sensors which whose IDs start with “I” are contact-based item sensors, and sensors whose IDs start with “D” are magnetic door sensors.
Figure 3.9: Floor plan and sensor location in TM004. There are 25 motion sensors deployed in the TM004 smart home. Motion sensors whose IDs start with “MA” are fitted with a lens that is responsive to resident motion in a wider area. Motion sensors whose IDs start with “M” are only sensitive to a small calibrated area.
3.9. The house has a master bedroom with a private bathroom, a second bedroom with bathroom, two living rooms, a dining room, and a kitchen. Residents can enter the house from the garage on the bottom left, from the back yard through the door on the right and via the main entrance located at the bottom middle. The TM004 testbed is monitored by 25 ambient sensors. There are two older adult residents staying in the house. Occasionally, their child will come and visit them and stay in their house for a few days.

3.3 CASAS Data Structure

As mentioned in Section 3.1, when resident activity is captured by a sensor, a sensor message is generated and sent via a local network to the smart home gateway, where the message is time stamped. The message is then forwarded to the publish/subscribe service and stored in a relational database hosted in the cloud. Each sensor message recorded in the database can be represented as a three-tuple, consisted of the message time tag, the sensor identifier, and the message content.

The states of the ambient sensors are usually binary, as they are either “active” or “inactive” at any given moment. The content of each sensor message differs depending on the type of the sensor. In CASAS smart homes, local PIR motion sensors send an “ON” message when resident motion is detected within the sensor’s field of view (“active” state), and an “OFF” message when the motion is no longer detected (“inactive” state). The magnetic door sensor sends an “OPEN” message when the door is opened (“active” state) and a “CLOSE” message when the door is closed (“inactive” state). Contact-based item
<table>
<thead>
<tr>
<th>Time Tag</th>
<th>Sensor ID</th>
<th>Message</th>
<th>Resident</th>
</tr>
</thead>
<tbody>
<tr>
<td>12/25/2016 15:24:06</td>
<td>LivingRoomAChair</td>
<td>OFF</td>
<td>R3</td>
</tr>
<tr>
<td>12/25/2016 15:24:08</td>
<td>LivingRoomAArea</td>
<td>OFF</td>
<td>R1</td>
</tr>
<tr>
<td>12/25/2016 15:24:08</td>
<td>KitchenADiningChair</td>
<td>ON</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:08</td>
<td>KitchenAArea</td>
<td>ON</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:09</td>
<td>DiningRoomAArea</td>
<td>ON</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:10</td>
<td>KitchenADiningChair</td>
<td>OFF</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:10</td>
<td>KitchenAArea</td>
<td>OFF</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:12</td>
<td>KitchenADiningChair</td>
<td>ON</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:12</td>
<td>MainEntryway</td>
<td>ON</td>
<td>R1</td>
</tr>
<tr>
<td>12/25/2016 15:24:13</td>
<td>KitchenAArea</td>
<td>ON</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:13</td>
<td>DiningRoomAArea</td>
<td>OFF</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:13</td>
<td>LivingRoomAArea</td>
<td>ON</td>
<td>R1,R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:13</td>
<td>KitchenADiningChair</td>
<td>OFF</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:14</td>
<td>MainEntryway</td>
<td>OFF</td>
<td>R1</td>
</tr>
<tr>
<td>12/25/2016 15:24:14</td>
<td>KitchenAArea</td>
<td>OFF</td>
<td>R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:14</td>
<td>LivingRoomAArea</td>
<td>OFF</td>
<td>R1,R2</td>
</tr>
<tr>
<td>12/25/2016 15:24:22</td>
<td>LivingRoomAArea</td>
<td>ON</td>
<td>R1,R2</td>
</tr>
</tbody>
</table>

Table 3.1: An example of sensor messages recorded in the TM004 dataset. Each sensor message is a three-tuple consisting of the timestamp, sensor ID, and message content. The resident label is provided by annotators. These serve as the ground truth for performance evaluation of multi-resident tracking algorithms.
sensors produce an “ABSENT” message when the item is removed from the sensor ("active" state) and a “PRESENT” message when the item is put back into place ("inactive" state).

Table 3.1 shows a series of PIR motion sensor messages recorded in the TM004 testbed. Each sensor activation is followed by a deactivation. In this dissertation, we use sensor event to refer to the subset of sensor messages that contain an “active state” message (“ON” message in this case). The goal of the multi-resident tracking is to associate each sensor event with the residents who activated the sensor. The ground truth labels of sensor event-to-resident association, as shown in the “Resident” column of Table 3.1, are produced by independent annotators with the help of a visualization tool (introduced in Section 3.4).

For the purpose of multi-resident tracking, we first extract a “sensor sequence” by focusing on the “active state” messages, as shown in Table 3.2. In single-resident settings, mutual information (MI), representing the likelihood that two sensors generating consecutive events, can be estimated using the sensor sequence [65]. Similarly, in the multi-resident environment, sensor pairs with a stronger MI relationship will occur close to each other in the recorded sensor event stream. Hence, the sensor sequence can still provide valuable information about the spatio-temporal relationship between sensors in a multi-resident smart home.

When a sensor is activated, we can take a snapshot of all sensor states in the smart home. In the snapshot, each active sensor represents an observation of a resident activity. We use term sensor observations to describe the set of active sensors in the snapshot. Table 3.3 shows a series of sensor observations extracted from the sensor messages in Table 3.1.
Table 3.2: Sensor sequence extracted from the sensor messages recorded in the TM004 dataset, as shown in Table 3.1.

<table>
<thead>
<tr>
<th>Time Tag</th>
<th>Sensor ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>12/25/2016 15:24:08</td>
<td>KitchenADiningChair</td>
</tr>
<tr>
<td>12/25/2016 15:24:08</td>
<td>KitchenAArea</td>
</tr>
<tr>
<td>12/25/2016 15:24:09</td>
<td>DiningRoomAArea</td>
</tr>
<tr>
<td>12/25/2016 15:24:12</td>
<td>KitchenADiningChair</td>
</tr>
<tr>
<td>12/25/2016 15:24:12</td>
<td>MainEntryway</td>
</tr>
<tr>
<td>12/25/2016 15:24:13</td>
<td>KitchenAArea</td>
</tr>
<tr>
<td>12/25/2016 15:24:13</td>
<td>LivingRoomAArea</td>
</tr>
<tr>
<td>12/25/2016 15:24:22</td>
<td>LivingRoomAArea</td>
</tr>
<tr>
<td>Time Tag</td>
<td>Sensor Observations</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>12/25/2016 15:24:08</td>
<td>KitchenADiningChair, MainDoor</td>
</tr>
<tr>
<td>12/25/2016 15:24:08</td>
<td>KitchenADiningChair, MainDoor, KitchenAArea</td>
</tr>
<tr>
<td>12/25/2016 15:24:09</td>
<td>DiningRoomAArea, KitchenADiningChair, MainDoor, KitchenAArea</td>
</tr>
<tr>
<td>12/25/2016 15:24:12</td>
<td>DiningRoomAArea, KitchenADiningChair, MainDoor</td>
</tr>
<tr>
<td>12/25/2016 15:24:12</td>
<td>DiningRoomAArea, KitchenADiningChair, MainDoor, MainEntryway</td>
</tr>
<tr>
<td>12/25/2016 15:24:13</td>
<td>DiningRoomAArea, KitchenADiningChair, MainDoor, KitchenAArea, MainEntryway</td>
</tr>
<tr>
<td>12/25/2016 15:24:13</td>
<td>LivingRoomAArea, KitchenADiningChair, MainDoor, KitchenAArea, MainEntryway</td>
</tr>
<tr>
<td>12/25/2016 15:24:22</td>
<td>LivingRoomAArea, MainDoor</td>
</tr>
</tbody>
</table>

Table 3.3: Sensor observations, recorded each time a sensor is activated.
The relationship between sensor messages, sensor events, and sensor observations is illustrated in Figure 3.10. In the graph, each vertical grid line represents the time a sensor in the smart home is activated. The circle represents the sensor observations, and the shaded box represents the time period that a sensor is in the active state. The red, blue and yellow arrows show the trajectories of residents R1, R2 and R3, respectively. These designations are based on manual annotation of sensor events with corresponding residents. As shown in Figure 3.10, a resident (R2) may be associated with multiple sensor observations (“DiningRoomAArea”, “KitchenAArea”, and “KitchADiningChair”) at the same time, while a sensor observation (“LivingRoomAArea”) may be associated with multiple residents (R1 and R2). Moreover, some sensor observations, such as “MainDoor” in the graph, are not associated with any resident. In the context of multi-resident tracking, we use the term false alarms or clutter process to refer to such sensor observations.

### 3.4 Dataset Visualization and Annotation

Acquiring ground truth labels for resident activity and data associations between residents and sensor events is a challenging task in practice. Traditional methods to acquire ground truth annotations involve self-reporting, video surveillance, in-house monitoring, or offline annotation [90]. Self-reporting annotations require the residents to manually record their locations and activities at all time [91]. This method is commonly used to acquire ground truth labels for activity recognition applications in smart environments, although requiring self-reporting of each resident location is generally impractical in real-life settings.
Figure 3.10: Association between residents and sensor events in TM004. The figure shows the relationship among sensor messages, sensor events, and sensor observations. The figure is generated using sensor messages recorded in the TM004 dataset from the same period as Table 3.1 – 3.3. The arrows in the graph show the movement of all active residents with respect to sensor observations.
Moreover, when smart home technology is used for older adults with dementia, the resident cannot reasonably be expected to remember which activities they performed, let alone regularly and accurately report their locations and activities at all times [92]. Theoretically, having a staff on site monitoring the residents [93, 94] or recording their activity through a video surveillance system [95, 96], would provide the most accurate ground truth labels. However, considering the impact the above annotation method would have on the residents’ daily routine and intrusion of their privacy, such methods are not practical in reality, especially when the experiment lasts for half a year or longer. Additionally, an on-site annotator would introduce yet another resident to track who does not follow a normal activity routine.

Offline annotation can be produced by experienced annotators by examining the raw sensor events with the help of visualization tools. According to a study by Szewczyk et al. [92], the availability of visualization tools can improve annotation ease, efficiency, and consistency compared to offline annotation based on just the raw sensor data or home diary. Available visualization tools for smart homes include 3D-based simulators, such as CASASSim [92], openSHS [97], or 2D web-based or desktop based visualization tools, such as PyViz [98] and CASASViz [99]. Most of these visualization tools aim to represent the state of all the sensors in the smart environment and are usually designed for single-resident scenarios. However, when analyzing and annotating sensor events collected in multi-resident scenarios, features such as showing the past trajectories of identified residents and visually indicating the relative length of time each resident has spent in each location are very helpful to annotators and researchers.
Figure 3.11: ActViz, a visualization tool for data association and activity annotation in multi-resident smart environments. Version 1.00 is developed using Visual Studio as a Universal Windows Platform (UWP) Application.
Figure 3.12: ActViz, a visualization tool for data association and activity annotation in multi-resident smart environments. Version 2.00 is developed using Electron and TypeScript. The application is available on all platforms, including Mac OS, Linux and Windows.
To fill the gap for multi-resident smart home visualization, we developed ActViz, as shown in Figure 3.11 and 3.12. The initial ActViz design (Figure 3.11) was developed as a Universal Windows Platform (UWP) application. It is used by our lab annotators to generate ground truth label for data association between sensor events and residents in both the Kyoto and TM004 datasets. In the application, the user can filter out activities based on the sensor ID, the sensor type, the sensor message, resident ID and the activity label. The past trajectories of each resident is shown as an overlay on top of the smart home floor plan, connected with color-coded solid lines. The opacity of these lines indicate the movement times with respective to the current event. If a sensor is in active state, the sensor’s box is filled with a solid color.

However, the representation of a resident moving from the location of one sensor to another with a solid line does not indicate the direction of the resident movement, causing confusion to the user if a resident is moving between sensor events within relatively short time intervals. Moreover, as an UWP application, it is difficult to port the visualization tool to other operating system (OS) platforms. Thus, we renovated the ActViz into version 2.0 (Figure 3.12), implemented with Electron, Typescript and ReactJS libraries. ActViz 2.0 can run on all OS platforms and potentially be deployed as a web application as well. The resident trajectories in the smart home are represented using solid filled arrows with a narrowed tail to indicate the direction of movement.
3.5 Summary

In this chapter, we introduced the CASAS smart home sensor technologies. We also introduced the two testbeds we use in our experiments to evaluate the performance of the proposed multi-resident tracking solutions. When a resident activity is detected by a sensor, the sensor sends the state change message to the smart home gateway server. With time tag appended at the gateway, the message is then forwarded to the cloud middleware for storage or further processing.

Each sensor message stored in the database is a three tuple composed of the time tag, sensor identifier, and sensor state. For the purpose of multi-resident tracking, we define a sensor sequence by focusing on the “active” messages. We further define a sensor observation as a snapshot of all sensor states when a sensor state switches to “active”. In multi-resident scenarios, a sensor observation may correspond to more than one resident, while one resident may correspond to multiple sensors that are “active” at the same time. Additionally, some sensor activations are caused by communication errors or false positive detections of the sensor. Such cases are referred to as a clutter process in the context of multi-resident tracking. In order to evaluate the performance of the multi-resident tracking algorithms, the sensor events recorded from both testbeds are labeled with the corresponding resident by an external annotator. To help with the ground truth annotation, we also developed a visualization tool, named ActViz.

Tracking of multiple residents in a smart environment relies on the construction of a resident movement model. Thus, in the following chapter, we first establish the theoretical
foundation of the predictability of resident movement in the smart environment.
A data-driven unsupervised multi-resident tracking approach relies on the accurate prediction resident mobility in the smart environment. But, how predictable is our routine behavior as sensed by a smart home? Equivalently, how much randomness exists in our daily routines? Previous smart home research, such as activity recognition, activity forecasting, anomaly detection, and building automation, are built on the assumption of regularity in our daily routines. Predictive behavioral models such as Active Le-Zi [100, 101] and trajectory mining [102] directly depend on the repetitive nature in our daily indoor behavior. However, due to the limited availability and longevity of smart home datasets, quantifying the predictability of resident mobility in indoor environments remains an unanswered question.

In this chapter, we investigate the predictability of indoor human mobility from an information theory perspective. A sensor sequence, extracted from sensor messages (as detailed in Chapter 3.3), constructs an observation of an underlying stationary stochastic process, i.e., the human indoor mobility model. Thus, the predictability of such a human indoor mobility model is bounded, according to Fano’s inequality, by the entropy rate. Given an observation of adequate length, under the ergodic hypothesis, the entropy rate of the underlying model can be estimated. Hence, the upper bound of predictability can be computed. The SHiB technology has been deployed to over 140 houses all over the world.
These sites recorded sensor events over a period ranging from a few weeks to multiple years. Provided with such rich datasets, we investigated the theoretical upper bound of the predictability of indoor human mobility under different occupancy scenarios.

In Section 4.1, we introduce the theoretical foundation of entropy rate estimation and its relation to predictability of the underlying statistical model. To validate the entropy estimation from an observed sequence, we employ alternative compression-based methods and a theoretical estimator. We demonstrate the theory with a simulation of a randomly-generated HMM model. We present experimental results with discussion in Section 4.2.

4.1 Entropy Rate and Predictability

The goal of this research is to investigate the theoretical bound for the predictability of indoor human mobility in real-world settings. Here we present the relationship between predictability and entropy rate estimated from real smart home empirical data.

As introduced in Section 3.1, the indoor locations of smart home residents are identified by sensor events. Resident trajectory in a home, $T$, is represented as a series of sensor events, $\{X_i\}_{i \in \mathbb{Z}}$, where $X_i$ is the $i^{th}$ sensor event in the trajectory. We define indoor mobility prediction as prediction of the next sensor event activated by the smart home resident(s) based on their past trajectories. In the case of multiple residents, we consider prediction of the next sensor event that is caused by any of the residents at the site, without attempting to identify which of the residents was responsible for the event. Multiple resident prediction can thus be viewed as prediction of a single, more “complex” resident.
in the building. The upper bound of the indoor mobility predictability can be calculated by estimating the entropy rate of the underlying stationary stochastic process [103].

### 4.1.1 Entropy Rate and Predictability

We assume that an ambient sensor-detected resident trajectory inside a building can be modeled by a stationary stochastic process \( \mathcal{X} = \{X_1, X_2, \ldots, X_n\} \), where \( X_i \) is a random variable representing the \( i \)th sensor event. The entropy rate, \( H(\mathcal{X}) \), measures the average conditional entropy \( H(X_{n+1}|X_n, X_{n-1}, \ldots, X_1) \) when \( n \) approaches infinity, as shown in Equation (4.1). This rate can also be interpreted as the growth of the information contained in the trajectory compared to the length of the trajectory.

\[
H(\mathcal{X}) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} H(X_{i+1}|X_i, \ldots, X_1) \quad (4.1)
\]

The most powerful prediction model for indoor mobility, characterized by the conditional probability distribution \( f(X_{n+1}|T_n) \), would predict the next sensor event based on the complete past history of resident trajectories. Here, \( X_{n+1} \) represents the next sensor event, and \( T_n \) represents the past history of resident trajectory up to time step \( n \). The maximum predictability \( \Pi^{max} \), defined as the average prediction accuracy of the next sensor event, can be formulated according to Equation (4.2). In Equation (4.2), \( P(T_i) \) represents the probability of observing a resident trajectory \( T_i \).

\[
\Pi^{max} = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \sup_{T_i} \sum_{X_{i+1}} f(X_{i+1}|T_i)p(T_i) \quad (4.2)
\]
When trajectory $T_i$ is observed, let $g(T_i) = \hat{X}_{i+1}$ be any estimator of $X_{i+1}$, and the probability of error be represented as $P_e(T_i) = Pr(\hat{X}_{i+1} \neq X_{i+1})$. Fano’s inequality [104] states that

$$H_b(P_e(T_i)) + P_e(T_i) \log_2(N) \geq H(X_{i+1}|T_i).$$  \hspace{1cm} (4.3)

In Equation (4.3), $H_b(p) = -p \log_2 p - (1-p) \log_2(1-p)$ is the binary entropy function. Thus, given conditional entropy $H(X_{i+1}|T_i)$, Fano’s inequality guarantees a lower bound of the probability of error $P_e(T_i)$, and thus an upper bound of the predictability $\Pi(T_i) = 1 - P_e(T_i)$. We can represent the left side of Equation (4.3) as a function of $\Pi(T_i)$ as shown in Equation (4.4).

$$H_F(\Pi(T_i)) = H_b(1 - \Pi(T_i)) + (1 - \Pi(T_i)) \log_2 N$$ \hspace{1cm} (4.4)

Based on the concavity of $H_F(\Pi(T_i))$, we can associate the maximum predictability $\Pi^{max}$ with the entropy rate of the underlying stationary stochastic process by applying Jensen’s inequality, as shown in Equation (4.8).

$$H(X) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} H(X_{i+1}|X_i, \ldots, X_1)$$ \hspace{1cm} (4.5)

$$= \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{T_i} H(X_{i+1}|T_i)P(T_i)$$ \hspace{1cm} (4.6)

$$\leq \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{T_i} S_F(\Pi(T_i))P(T_i)$$ \hspace{1cm} (4.7)

$$\leq H_F \left( \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{T_i} \Pi(T_i)P(T_i) \right)$$ \hspace{1cm} (4.8)

$$= H_F(\Pi^{max})$$ \hspace{1cm} (4.9)
According to Equation (4.9), given the entropy rate of the underlying stochastic process, the upper bound of predictability of the sequence can be calculated numerically.

4.1.2 Estimate of Entropy Rate

Having established the association between entropy rate and predictability, the key challenge is to reliably estimate the entropy rate from empirical data. The entropy rate, defined in Equation (4.1), is the conditional entropy of the future random variable as the length of the random process approaches infinity. Due to the limited sample size of our empirical data, we implement multiple entropy rate estimators to ensure the consistency of entropy rate estimation.

Previous work by Song et al. [103] and Smith et al. [105] used an entropy rate estimator based on Limpel-Ziv data compression proposed by Kontoyiannis et al. [106]. Let $X = \{X_i\}$ be a stationary ergodic process with entropy rate $H(X) > 0$.

\[
\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \frac{\Lambda_i^k}{\log_2 n} = \frac{1}{H(X)} \quad (4.10)
\]

In Equation (4.10), $\Lambda_i^k$ represents the length of the shortest substring starting at position $i$ that does not appear as a continuous substring of the previous $k$ symbols.

However, the entropy rate can also be estimated using the code length after data compression. Let $R(T_n)$ denote the size in bits of a sequence $T_n = \{X_n, X_{n-1}, \ldots, X_1\}$ after compression. The code length per unit, $r(n) = \frac{1}{n} R(T_n)$, is always larger than the entropy rate $H(X)$ [107]. To evaluate performance of existing compression methods with
respect to the theoretical bounds, we consider multiple state-of-the-art data compressors. Specifically, we evaluate 7-zip deflate (an implementation of Lempel-Ziv 77 with Huffman coding), 7-zip LZMA (an optimized version of LZ77) and 7-zip PPMD (a lossless data compression using prediction by partial matching).

Though all of the above approaches will converge to the true entropy rate, the estimate accuracies differ and are affected by the data sample size. To demonstrate the differences between the estimators and contrast the estimated values with the true entropy rate, we generated a state sequence of 10,000,000 time steps based on a second-order Markov chain with 20 states, where the probability of the current state is dependent on the previous two states. The state transition probabilities of the second-order Markov chain, characterized by $P(X_{i+1}|X_i, X_{i-1})$, are generated randomly. Thus, the true entropy rate of the synthetic state sequence can be calculated based on the definition of entropy rate as shown in Equation (4.11).

\[
H(X) = H(X_{i+1}|X_i, X_{i-1}) = \sum_{x_i, x_{i-1}} P(x_i, x_{i-1})H(X_{i+1}|x_i, x_{i-1})
\] (4.11)

In Equation (4.11), $P(x_i, x_{i-1})$ represents the stationary probability distribution of state sequence $\{x_i, x_{i-1}\}$, and $H(X_{i+1}|x_i, x_{i-1})$ represents the entropy of random variable $X_{i+1}$ given the value of the previous two states.

In Figure 4.1, we plot the entropy rates based on sequence size using multiple established methods including the LZ77 (deflate) ($H_{LZ77}$), LZMA ($H_{LZMA}$) and PPMD
Figure 4.1: Entropy rate estimated based on synthetic data generated by a second-order Markov chain.

(H^{PPMD}) algorithms, as well as the estimator $H^{est}$ defined in Equation (4.10). The corresponding predictability of each estimator is shown in Figure 4.2. To put the results in perspective, we also include the random entropy $H^{rand}$, the temporal uncorrelated entropy $H^{unc}$ and the entropy rate, $H^{mc}$, of a first order Markov chain fitted to the synthetic sequence. The random entropy $H^{rand} = \log_2 N$ represents the maximum amount of possible information content in any sequence of $N$ states. The uncorrelated entropy $H^{unc}$ assumes that there is no temporal correlation between consecutive random variables in the random
process. In other words, the sensor event at each time step is independently drawn from a probability distribution $P(X_i)$. Thus, $H_{\text{unc}}$ can be calculated according to Equation (4.12), where $P(X_i)$ can be estimated by counting the occurrences of each observable symbol.

$$H_{\text{unc}} = - \sum_{X_i} P(X_i) \log_2 P(X_i)$$ (4.12)

The Markov chain-based entropy rate $H_{\text{mc}}$ assumes that the state sequence can be modeled by a Markov chain, characterized by the conditional probability of state transition.
\( P(X_{i+1}|X_i) \). Based on the empirical data, the transition probability can be estimated by maximizing the likelihood of the observed sequence. Thus, the entropy rate of the constructed Markov chain can be calculated according to Equation (4.13) [104]. Here, \( P(X_i) \) is the stationary state distribution of the Markov chain, calculated by solving Equation (4.14).

\[
H^{MC} = - \sum_{X_{i+1},X_i} P(X_i)P(X_{i+1}|X_i) \log_2 P(X_{i+1}|X_i) \tag{4.13}
\]

\[
P(X_i) = \sum_{X_j} P(X_i|X_j) \forall j \tag{4.14}
\]

In Figure 4.1, the true entropy rate (the red dashed line) is calculated according to Equation (4.11). As shown, all estimators \( (H^{LZ77}, H^{LZMA}, H^{PPMD}, \text{ and } H^{est}) \) converge to the true entropy rate as the sequence length increases. Among the estimators based on the compressed code length \( (H^{LZ77}, H^{LZMA}, \text{ and } H^{PPMD}) \), PPMD yields the tightest upper bounds and fastest convergence. Due to the dictionary size, the entropy rates reported by those estimators are higher than the actual process entropy rate. Comparatively, \( H^{est} \) provides the most accurate estimate among all of the tested entropy rate estimators. However, the estimators based on compressed code length are guaranteed to be an upper bound to the actual entropy rate, but \( H^{est} \) may undershoot when there is not enough data, leading to an over-estimated predictability.

The entropy rate of any stochastic process with an alphabet size of 20 should lie between the random entropy \( H^{rand} \) (4.33 in this example), indicating that the process is completely random, and 0, indicating that the process is fully deterministic. The second-order
Markov model has an actual entropy rate of 1.35, and thus a corresponding predictability of 83.51%. The entropy rates estimated by $H^{est}$ and $H^{PPMD}$ are 1.52 and 1.60, resulting in a predictability error of 3.5% and 4.01%, respectively. Since the synthetic data in this example is generated by a second-order Markov chain, a first-order Markov chain is not sufficient to capture patterns in the data. As a result, the entropy rate $H^{MC}$ calculated by fitting a first-order Markov chain to the synthetic data is 3.58, much higher than the actual entropy rate of the second-order Markov chain or the estimates reported by $H^{est}$ and $H^{PPMD}$.

Additionally, the predictability estimated by $H^{MC}$ is more than 30% lower than estimates derived from $H^{est}$ or $H^{PPMD}$. Compared to $H^{MC}$, the temporally-uncorrelated entropy $H^{unc}$ is less powerful for capturing the regularities in the synthetic data. Quantitatively, the predictability corresponding to the uncorrelated entropy is close to a fully-random sequence, which is more than 40% lower than the first-order Markov chain.

<table>
<thead>
<tr>
<th>Site</th>
<th>Sensors</th>
<th>Residents</th>
<th>Messages</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>27</td>
<td>2</td>
<td>1,178,506</td>
<td>54</td>
</tr>
<tr>
<td>m2</td>
<td>42</td>
<td>2</td>
<td>6,826,679</td>
<td>236</td>
</tr>
<tr>
<td>m3</td>
<td>25</td>
<td>2-3</td>
<td>1,378,574</td>
<td>31</td>
</tr>
<tr>
<td>m4</td>
<td>70</td>
<td>2</td>
<td>18,944,701</td>
<td>614</td>
</tr>
<tr>
<td>m5</td>
<td>36</td>
<td>2-3</td>
<td>973,349</td>
<td>52</td>
</tr>
<tr>
<td>m6</td>
<td>19</td>
<td>2</td>
<td>1,155,121</td>
<td>31</td>
</tr>
<tr>
<td>m7</td>
<td>22</td>
<td>2</td>
<td>1,550,683</td>
<td>132</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of the smart home datasets used in the empirical evaluation.
<table>
<thead>
<tr>
<th>Site</th>
<th>Sensors</th>
<th>Residents</th>
<th>Messages</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>m8</td>
<td>17</td>
<td>2-3</td>
<td>980,093</td>
<td>10</td>
</tr>
<tr>
<td>m9</td>
<td>39</td>
<td>2-3</td>
<td>409,115</td>
<td>23</td>
</tr>
<tr>
<td>m10</td>
<td>47</td>
<td>2-3</td>
<td>176,412</td>
<td>8</td>
</tr>
<tr>
<td>m11</td>
<td>22</td>
<td>2-3</td>
<td>2,144</td>
<td>2</td>
</tr>
<tr>
<td>m12</td>
<td>37</td>
<td>2-3</td>
<td>129,213</td>
<td>6</td>
</tr>
<tr>
<td>m13</td>
<td>35</td>
<td>2-3</td>
<td>60,377</td>
<td>13</td>
</tr>
<tr>
<td>m14</td>
<td>60</td>
<td>2-3</td>
<td>426,996</td>
<td>13</td>
</tr>
<tr>
<td>m15</td>
<td>40</td>
<td>2-3</td>
<td>236,637</td>
<td>8</td>
</tr>
<tr>
<td>m16</td>
<td>51</td>
<td>2-3</td>
<td>293,211</td>
<td>10</td>
</tr>
<tr>
<td>m17</td>
<td>24</td>
<td>2</td>
<td>153,603</td>
<td>10</td>
</tr>
<tr>
<td>m18</td>
<td>27</td>
<td>2</td>
<td>228,634</td>
<td>10</td>
</tr>
<tr>
<td>m19</td>
<td>22</td>
<td>2</td>
<td>376,708</td>
<td>10</td>
</tr>
<tr>
<td>m20</td>
<td>30</td>
<td>2</td>
<td>840,649</td>
<td>16</td>
</tr>
<tr>
<td>m21</td>
<td>30</td>
<td>2</td>
<td>377,391</td>
<td>13</td>
</tr>
<tr>
<td>m22</td>
<td>35</td>
<td>2</td>
<td>1,432,718</td>
<td>80</td>
</tr>
<tr>
<td>m23</td>
<td>36</td>
<td>2</td>
<td>70,066</td>
<td>3</td>
</tr>
<tr>
<td>m24</td>
<td>27</td>
<td>2</td>
<td>249,987</td>
<td>10</td>
</tr>
<tr>
<td>m25</td>
<td>19</td>
<td>2</td>
<td>307,187</td>
<td>10</td>
</tr>
<tr>
<td>m26</td>
<td>25</td>
<td>2</td>
<td>284,137</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of the smart home datasets used in the empirical evaluation.
<table>
<thead>
<tr>
<th>Site</th>
<th>Sensors</th>
<th>Residents</th>
<th>Messages</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>m27</td>
<td>36</td>
<td>2</td>
<td>220,960</td>
<td>14</td>
</tr>
<tr>
<td>m28</td>
<td>24</td>
<td>2</td>
<td>324,905</td>
<td>12</td>
</tr>
<tr>
<td>m29</td>
<td>34</td>
<td>3</td>
<td>472,826</td>
<td>15</td>
</tr>
<tr>
<td>m30</td>
<td>30</td>
<td>2</td>
<td>114,684</td>
<td>4</td>
</tr>
<tr>
<td>m31</td>
<td>30</td>
<td>2</td>
<td>249,387</td>
<td>14</td>
</tr>
<tr>
<td>b1</td>
<td>27</td>
<td>2</td>
<td>328,260</td>
<td>9</td>
</tr>
<tr>
<td>b2</td>
<td>33</td>
<td>2</td>
<td>437,733</td>
<td>23</td>
</tr>
<tr>
<td>b3</td>
<td>25</td>
<td>3</td>
<td>784,428</td>
<td>21</td>
</tr>
<tr>
<td>b4</td>
<td>20</td>
<td>2</td>
<td>120,469</td>
<td>4</td>
</tr>
<tr>
<td>b5</td>
<td>27</td>
<td>4</td>
<td>594,493</td>
<td>13</td>
</tr>
<tr>
<td>b6</td>
<td>19</td>
<td>2</td>
<td>423,214</td>
<td>14</td>
</tr>
<tr>
<td>b7</td>
<td>21</td>
<td>2</td>
<td>242,491</td>
<td>10</td>
</tr>
<tr>
<td>b8</td>
<td>33</td>
<td>2</td>
<td>311,322</td>
<td>15</td>
</tr>
<tr>
<td>b9</td>
<td>21</td>
<td>2</td>
<td>288,301</td>
<td>13</td>
</tr>
<tr>
<td>b10</td>
<td>30</td>
<td>3</td>
<td>1,431,903</td>
<td>24</td>
</tr>
<tr>
<td>b11</td>
<td>30</td>
<td>3</td>
<td>599,925</td>
<td>19</td>
</tr>
<tr>
<td>b12</td>
<td>30</td>
<td>2</td>
<td>249,289</td>
<td>11</td>
</tr>
<tr>
<td>b13</td>
<td>32</td>
<td>2</td>
<td>528,142</td>
<td>15</td>
</tr>
<tr>
<td>b14</td>
<td>35</td>
<td>2</td>
<td>63,146</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of the smart home datasets used in the empirical evaluation.
<table>
<thead>
<tr>
<th>Site</th>
<th>Sensors</th>
<th>Residents</th>
<th>Messages</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>20</td>
<td>1</td>
<td>1,509,615</td>
<td>82</td>
</tr>
<tr>
<td>p2</td>
<td>31</td>
<td>1</td>
<td>207,690</td>
<td>12</td>
</tr>
<tr>
<td>p3</td>
<td>29</td>
<td>1</td>
<td>224,022</td>
<td>10</td>
</tr>
<tr>
<td>p4</td>
<td>18</td>
<td>1</td>
<td>759,750</td>
<td>51</td>
</tr>
<tr>
<td>p5</td>
<td>20</td>
<td>1</td>
<td>3,884,561</td>
<td>220</td>
</tr>
<tr>
<td>p6</td>
<td>26</td>
<td>1</td>
<td>611,724</td>
<td>18</td>
</tr>
<tr>
<td>p7</td>
<td>30</td>
<td>1</td>
<td>296,860</td>
<td>14</td>
</tr>
<tr>
<td>p8</td>
<td>23</td>
<td>1</td>
<td>286,749</td>
<td>18</td>
</tr>
<tr>
<td>p9</td>
<td>18</td>
<td>1</td>
<td>489,311</td>
<td>21</td>
</tr>
<tr>
<td>p10</td>
<td>35</td>
<td>1</td>
<td>91,890</td>
<td>7</td>
</tr>
<tr>
<td>s1</td>
<td>32</td>
<td>1</td>
<td>327,271</td>
<td>26</td>
</tr>
<tr>
<td>s2</td>
<td>32</td>
<td>1</td>
<td>285,207</td>
<td>26</td>
</tr>
<tr>
<td>s3</td>
<td>32</td>
<td>1</td>
<td>256,301</td>
<td>26</td>
</tr>
<tr>
<td>s4</td>
<td>19</td>
<td>1</td>
<td>639,517</td>
<td>54</td>
</tr>
<tr>
<td>s5</td>
<td>20</td>
<td>1</td>
<td>56,729</td>
<td>7</td>
</tr>
<tr>
<td>s6</td>
<td>34</td>
<td>1</td>
<td>2,583,713</td>
<td>92</td>
</tr>
<tr>
<td>s7</td>
<td>30</td>
<td>1</td>
<td>2,399,519</td>
<td>146</td>
</tr>
<tr>
<td>s8</td>
<td>21</td>
<td>1</td>
<td>2,683,317</td>
<td>299</td>
</tr>
<tr>
<td>s9</td>
<td>33</td>
<td>1</td>
<td>2,460,156</td>
<td>118</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of the smart home datasets used in the empirical evaluation.
<table>
<thead>
<tr>
<th>Site</th>
<th>Sensors</th>
<th>Residents</th>
<th>Messages</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>s10</td>
<td>30</td>
<td>1</td>
<td>3,154,735</td>
<td>276</td>
</tr>
<tr>
<td>s11</td>
<td>39</td>
<td>1</td>
<td>2,028,153</td>
<td>169</td>
</tr>
<tr>
<td>s12</td>
<td>33</td>
<td>1</td>
<td>3,395,594</td>
<td>269</td>
</tr>
<tr>
<td>s13</td>
<td>27</td>
<td>1</td>
<td>5,393,455</td>
<td>239</td>
</tr>
<tr>
<td>s14</td>
<td>33</td>
<td>1</td>
<td>3,423,249</td>
<td>309</td>
</tr>
<tr>
<td>s15</td>
<td>20</td>
<td>1</td>
<td>298,254</td>
<td>21</td>
</tr>
<tr>
<td>s16</td>
<td>29</td>
<td>1</td>
<td>1,117,701</td>
<td>73</td>
</tr>
<tr>
<td>s17</td>
<td>26</td>
<td>1</td>
<td>3,778,629</td>
<td>273</td>
</tr>
<tr>
<td>s18</td>
<td>20</td>
<td>1</td>
<td>818,002</td>
<td>51</td>
</tr>
<tr>
<td>s19</td>
<td>19</td>
<td>1</td>
<td>205,246</td>
<td>24</td>
</tr>
<tr>
<td>s20</td>
<td>16</td>
<td>1</td>
<td>385,107</td>
<td>44</td>
</tr>
<tr>
<td>s21</td>
<td>25</td>
<td>1</td>
<td>2,991,898</td>
<td>153</td>
</tr>
<tr>
<td>s22</td>
<td>27</td>
<td>1</td>
<td>1,528,537</td>
<td>223</td>
</tr>
<tr>
<td>s23</td>
<td>20</td>
<td>1</td>
<td>906,345</td>
<td>100</td>
</tr>
<tr>
<td>s24</td>
<td>28</td>
<td>1</td>
<td>438,681</td>
<td>31</td>
</tr>
<tr>
<td>s25</td>
<td>16</td>
<td>1</td>
<td>845,371</td>
<td>85</td>
</tr>
<tr>
<td>s26</td>
<td>18</td>
<td>1</td>
<td>12,299</td>
<td>22</td>
</tr>
<tr>
<td>s27</td>
<td>32</td>
<td>1</td>
<td>2,460,714</td>
<td>220</td>
</tr>
<tr>
<td>s28</td>
<td>17</td>
<td>1</td>
<td>1,654,459</td>
<td>178</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of the smart home datasets used in the empirical evaluation.
<table>
<thead>
<tr>
<th>Site</th>
<th>Sensors</th>
<th>Residents</th>
<th>Messages</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>s29</td>
<td>17</td>
<td>1</td>
<td>1,161,989</td>
<td>181</td>
</tr>
<tr>
<td>s30</td>
<td>20</td>
<td>1</td>
<td>4,323,961</td>
<td>175</td>
</tr>
<tr>
<td>s31</td>
<td>20</td>
<td>1</td>
<td>1,715,885</td>
<td>98</td>
</tr>
<tr>
<td>s32</td>
<td>26</td>
<td>1</td>
<td>371,175</td>
<td>27</td>
</tr>
<tr>
<td>s33</td>
<td>14</td>
<td>1</td>
<td>1,379,134</td>
<td>211</td>
</tr>
<tr>
<td>s34</td>
<td>20</td>
<td>1</td>
<td>1,155,121</td>
<td>211</td>
</tr>
<tr>
<td>s35</td>
<td>15</td>
<td>1</td>
<td>1,681,679</td>
<td>207</td>
</tr>
<tr>
<td>s36</td>
<td>18</td>
<td>1</td>
<td>217,944</td>
<td>8</td>
</tr>
<tr>
<td>s37</td>
<td>25</td>
<td>1</td>
<td>116,878</td>
<td>14</td>
</tr>
<tr>
<td>s38</td>
<td>24</td>
<td>1</td>
<td>280,500</td>
<td>12</td>
</tr>
<tr>
<td>s39</td>
<td>32</td>
<td>1</td>
<td>267,881</td>
<td>12</td>
</tr>
<tr>
<td>s40</td>
<td>20</td>
<td>1</td>
<td>86,996</td>
<td>10</td>
</tr>
<tr>
<td>s41</td>
<td>22</td>
<td>1</td>
<td>364,617</td>
<td>9</td>
</tr>
<tr>
<td>s42</td>
<td>28</td>
<td>1</td>
<td>287,873</td>
<td>14</td>
</tr>
<tr>
<td>s43</td>
<td>23</td>
<td>1</td>
<td>565,153</td>
<td>20</td>
</tr>
<tr>
<td>s44</td>
<td>17</td>
<td>1</td>
<td>888,745</td>
<td>60</td>
</tr>
<tr>
<td>s45</td>
<td>17</td>
<td>1</td>
<td>2,745,749</td>
<td>28</td>
</tr>
<tr>
<td>s46</td>
<td>19</td>
<td>1</td>
<td>5,321,015</td>
<td>61</td>
</tr>
<tr>
<td>s47</td>
<td>19</td>
<td>1</td>
<td>9,768,315</td>
<td>90</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of the smart home datasets used in the empirical evaluation.
<table>
<thead>
<tr>
<th>Site</th>
<th>Sensors</th>
<th>Residents</th>
<th>Messages</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>s48</td>
<td>17</td>
<td>1</td>
<td>773,010</td>
<td>67</td>
</tr>
<tr>
<td>s49</td>
<td>19</td>
<td>1</td>
<td>439,836</td>
<td>43</td>
</tr>
<tr>
<td>s50</td>
<td>22</td>
<td>1</td>
<td>395,083</td>
<td>20</td>
</tr>
<tr>
<td>s51</td>
<td>30</td>
<td>1</td>
<td>596,031</td>
<td>20</td>
</tr>
<tr>
<td>s52</td>
<td>17</td>
<td>1</td>
<td>316,837</td>
<td>20</td>
</tr>
<tr>
<td>s53</td>
<td>21</td>
<td>1</td>
<td>185,086</td>
<td>19</td>
</tr>
<tr>
<td>s54</td>
<td>20</td>
<td>1</td>
<td>517,534</td>
<td>15</td>
</tr>
<tr>
<td>s55</td>
<td>26</td>
<td>1</td>
<td>285,168</td>
<td>10</td>
</tr>
<tr>
<td>s56</td>
<td>31</td>
<td>1</td>
<td>311,815</td>
<td>12</td>
</tr>
<tr>
<td>s57</td>
<td>29</td>
<td>1</td>
<td>254,952</td>
<td>10</td>
</tr>
<tr>
<td>s58</td>
<td>30</td>
<td>1</td>
<td>700,171</td>
<td>22</td>
</tr>
<tr>
<td>s59</td>
<td>23</td>
<td>1</td>
<td>611,557</td>
<td>19</td>
</tr>
<tr>
<td>s60</td>
<td>36</td>
<td>1</td>
<td>468,679</td>
<td>10</td>
</tr>
<tr>
<td>s61</td>
<td>36</td>
<td>1</td>
<td>134,012</td>
<td>11</td>
</tr>
<tr>
<td>s62</td>
<td>19</td>
<td>1</td>
<td>117,986</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of the smart home datasets used in the empirical evaluation.
4.2 Experimental Results and Discussion

In the previous section, we presented a formal method to determine the theoretical limits of indoor human mobility predictability in real-life settings. Here, we validate our theoretical formulations based on sensor message data collected from 117 smart homes, characterized in Table 4.1. We collected these data from participants using the CASAS Smart Home in a Box technology (SHiB, [108,109]). Residents in the smart homes performed their normal daily routines while ambient sensors collected data. Data collection in these homes ranged from 2 weeks to over 11 years. Among these homes, 45 smart homes have ≥ 2 residents and 14 of these include pets. The remainder of the 72 smart homes are single-resident, 10 of which include pets. Many of these datasets, as well as the code, are available online.

Since the estimated entropy rate accuracy is affected by the data sample size, we start by establishing the relationship between entropy rate estimators and the length of the recorded message sequence. Figure 4.3 plots the estimated entropy rates (top) and the corresponding predictability (bottom) against the number of sensor messages. In this example, the sensor messages were collected in a home with multiple residents, home m3, over 6 months. This home was chosen because it houses multiple residents (making the data complex). Furthermore, ground truth resident labels are provided for a subset of this dataset.

\[2^{2}\text{CASAS datasets and code are available at http://casas.wsu.edu/}.$$
Figure 4.3: Entropy rates estimated based on sensor messages recorded in the smart home (TM004).

home’s data, associating each sensor message with a corresponding resident ID. To provide a time reference in the graphs, the plots include red vertical lines indicate the first, second, third, and fourth weeks, as well as the second, third, and sixth months, on a log scale. Entropy rates in Figure 4.3 are estimated using the code length of the compressed data based on the LZ77 (deflate) ($H^{LZ77}$), LZMA ($H^{LZMA}$), and PPMD ($H^{PPMD}$) algorithms, as well as the entropy rate estimator $H^{est}$ defined in Equation (4.10). The results are compared against random entropy $H^{rand}$, uncorrelated entropy $H^{unc}$, and the entropy
rate $H^{mc}$ calculated by fitting a first-order Markov chain to the sensor messages. The predictabilities plotted in Figure 4.3 are calculated according to Equation (4.9). Since there are 25 sensors in home m3, the size of alphabet $N$ in Equation (4.9) is 25.

Among the entropy rate estimators ($H_{LZ77}^{LZ77}$, $H_{LZMA}^{LZMA}$, $H_{PPMD}^{PPMD}$ and $H^{est}$), $H^{est}$ provides the lowest estimates. In the figure, $H^{est}$ oscillates when there is not enough data, and stabilizes after consuming two months of data. Based on such observations, we acknowledge that the entropy rate reported by $H^{est}$ is the best approximation to the true
Figure 4.5: Scatter plot of the estimated entropy rate of resident mobility in smart homes. Single resident smart homes are represented in red circles and multi-resident smart homes are represented in blue. A star marker indicates that the household includes pets.

The entropy rate of the underlying stochastic process that generates the observed human trajectories. As shown in Figure 4.3, the upper bound of indoor mobility predictions in this home converges to 79.3% according to the entropy rate estimated by $H^{est}$. If a first-order Markov model is used to predict the next sensor message triggered by the residents living...
Figure 4.6: Scatter plot of the estimated predictability of resident mobility in smart homes.

Single resident smart homes are represented in red circles and multi-resident smart homes are represented in blue. A star marker indicates that the household includes pets.

in the smart home, the prediction accuracy, according to the entropy rate $H^{mc}$, is 67.5%, indicating that an improvement of 11.8% can potentially be achieved.

By repeating the above experiment for each smart home, we calculated the predictability upper bound of each home. Figure 4.6 summarizes the upper bounds of res-
Figure 4.7: Example of resident trajectories observed in a single-resident smart home.

Ident mobility predictability using entropy estimator $H^{est}$. Homes with single residents are shown in red and multi-resident homes are colored blue. Homes with pets as well as human residents are indicated with a star symbol. The indoor mobility of residents in some of these homes exhibits high orders of regularity, where the prediction accuracy of resident movement exceeds 95%. As expected, these occur in single-resident homes with no pets. On the contrary, in many multi-resident smart homes, the predictability is much lower and, in some cases, drops below 65%.

To further explore the effect of multiple residents and pets on the predictability
of indoor mobility, we pictorially compare predictability between a single resident smart home and a multi-resident smart home. Figure 4.7 shows two resident trajectories that are subsets of the collected sensor data. One trajectory is recorded in a single-resident smart home (home s4) and the other is recorded in a multi-resident smart home (home m3). In the
Figure 4.9: Comparison of resident mobility predictability with/without multiple residents and with/without pets.

The single-resident trajectory recorded in home s1, the resident generally triggers sensors along the path as the individual moves from one part of the home to another. In this case, the
next sensor message is expected to be strongly correlated to the previous sensor message. In the case of multi-resident home m3, the merged (dual-resident) trajectory is indicated by red arrows, and the actual path of each resident, identified as R1 and R2, are shown with dotted green arrows and dotted blue arrows, respectively. In multi-resident smart homes, given that sensor messages are not mapped to specific residents, the combined movement is more complex and therefore more difficult to predict. As a result, we expect that the predictability of resident mobility is lower in multi-resident settings when compared to the single-resident scenario. According to the statistics shown in Figure 4.9, the predictability limit of resident mobility in single-resident homes averages 81.86%, while the upper bound of mobility predictability in multi-resident settings is 74.40%, approximately 7.46% lower on average.

In addition to the uncertainty caused by multiple residents, pets in the household can also trigger ambient sensors and cause the creation of sensor messages. Because of their smaller mass, some pets do not consistently trigger passive infrared motion sensors, causing what appears to be a “teleporting” effect. This causes an increase in data noise and an overall decrease in predictability. In the experiments, we studied the predictability limits of smart homes with pets in comparison with the smart homes without pets. Based on these results, we found that the decrease in the predictability of indoor mobility caused by pets is 3% in both single-resident and multi-resident scenarios. Specifically, the predictability limit for single resident homes with pets averages 79.00% and multi-resident homes with pets averages 72.10%.

Because of the inherent difficulty in associating sensor messages with specific indi-
individuals in multi-resident scenarios, a simple Markov chain is commonly used to construct a human mobility model based on recorded sensor messages for these situations [84]. A Markov chain conditions the probability distribution of the next sensor message only on the previous sensor message. To model multi-resident movement, we fit a first-order Markov chain to the recorded data and calculated both entropy rates and associated predictability. In Figure 4.9, we average these values across all single-resident homes without pets, multi-resident homes without pets, single-resident homes with pets, and multi-resident homes with pets. We also compute and plot the theoretical upper bound on predictability for these house categories. In single-resident cases where there are no pets in the household, we found that the predictability of the first-order Markov chain is 6% below the theoretical upper bound of the estimated predictability, while in multi-resident homes or smart homes with pets, the predictability is closer to 8% below the theoretical limit. We note that the multi-resident homes exhibit the greatest variability in number of residents as well as overall movement patterns. At any particular point in time, the number of residents in these spaces can vary from 0 to 4+. As the number of residents increases, so may the randomness observed in the sequence. Based on the above results, there is certainly room for improvement in constructing more representative mobility models than are found in first-order Markov chains, especially in multi-resident settings.

We note that the empirical results discussed here are subject to sensor error. For PIR motion sensors, error can originate from multiple sources. In some homes, there may be gaps in coverage. As a result, residents may move to locations that are not reflected in the sensor message sequence. Occasionally, motion sensors can generate false positive messages
when hit with heat from an outside source such as a laser printer or baseboard heater. We estimated these errors occurred fewer than 0.05% of the days that were monitored. More commonly, sensor messages may be lost due to communication errors. In the CASAS smart homes, these are alerted as "radio errors". Such errors occurred on less than 0.45% of the monitored days.

Some of the errors due to ambient sensors could be corrected by fusing multiple sensor sources [110], such as fusing ambient sensor data with that of wearable and object sensors. Each data source is faced with challenges including sensor noise, participant non-compliance, and gaps in coverage. However, fusing data from multiple sources can harness the strengths of the individual sensor modalities to compensate for the weaknesses of others.

4.3 Summary

In this chapter, we presented the study of the limits of predictability for indoor human mobility using sensor events recorded in real-life smart homes. We examined multiple methods for modeling this predictability and provided evidence to support the models based on sensor messages collected from 117 smart homes. In single-resident smart homes, we found that the upper bound of the prediction accuracy for the next sensor message averages 83%. With the presence of multiple people in the smart home, the predictability lowers on average by 11%. If pets are present in smart homes, the predictability decreases by approximately 3% for both single-resident and multi-resident settings.
The study presented in this chapter provides a quantitative measurement to back up the hypothesis of regularity in human daily routines. For applications that rely on the prediction of resident movement trajectories, developers can make more educated design decisions to achieve better user experience based on the statistical limits of mobility prediction performance. In scenarios where there are multiple residents or pets, the mobility of human are largely predictable, though 11% lower than single-resident cases on average.

Moreover, we can use the difference between the theoretical limits of predictability and the prediction accuracy of a particular mobility model as a quantitative absolute measurement of the performance of the mobility model. In the experiments, we assessed the performance of Markov chain-based mobility model in both single-resident and multi-resident settings. In terms of predictability, we found that the performance of Markov chain models is approximately 6% lower than the theoretical upper bound in single-resident settings and 11% lower in multi-resident settings. The results indicate that more representative models could be developed for indoor human mobility especially in multi-resident homes. If a mobility model is to be constructed to achieve a performance above the theoretical limit of predictability, information in addition to the residents’ past trajectory, such as the time of day, the day of the week, and the length of resident stay at specific locations will be needed.

Having established the regularity of resident daily movement in smart environments, we introduce a formalization of the multi-resident tracking problem in the form of a Bayes filtering problem. The multi-resident tracking methods proposed in the following chapter leverage the regularity of human indoor mobility exhibited in the sensor sequence and offer
an unsupervised solution that only consumes unannotated sensor events.
The objective of multi-resident tracking in smart environments is to associate sensor events with the corresponding residents. Considering challenges such as scalability, privacy constraints, and practicality of real-life deployment, we propose a solution to this problem in sMRT, an unsupervised multi-resident tracking method based on sensor vectorization. sMRT operates directly on smart home-based raw sensor events and does not require the availability of any additional information such as floor plan, sensor location, resident activity labels, number of residents in the space, or data association annotations. Furthermore, considering real-life scenarios, sMRT permits an arbitrary number of residents in the smart environment, rather than tracking only a fixed number of residents.

We formulate the MRT problem as a sequential Bayes estimation (or filtering) problem in the framework of finite set statistics (FISST) [111]. By translating the sensor events and sensor observations into a latent hyper-dimensional space, sMRT, presented in Section 5.1, estimates and propagates the probability density of all residents in the smart environment and computes associations based on new sensor observations. This filtering process relies on a resident dynamic model and a measurement model. The dynamic model predicts the state of a resident for the next time step based on its current state vector. The measurement model relates the sensor observation with the resident state vector. Associations between sensor events and residents are determined by an MAP estimator.
During tracking, we model the movement of each resident in the smart environment as a point target maneuvering in the latent space. To simplify the computation, we further hypothesize that the dynamics of the point targets can be approximated with a constant velocity model. Thus, the resident state space becomes partially observable. The relationship between the measurements (i.e., the mapping of sensor observations to the latent space via sensor vectorization) and the resident state vectors can be captured by a simple linear function.

Noticing that the sMRT is actually a generative model, instead of the constant velocity dynamic model assumption, we propose an unsupervised training method, sMRT-ML (Section 5.2). sMRT-ML compares the posterior probability density computed based on the dynamic model and past sensor observations and the current sensor observations. By maximizing the likelihood of current sensor measurements, we can learn the linear multiplier and other parameters governing the dynamic and measurement models. Finally, in Section 5.3, we discuss the track maintenance strategies to improve the tracking consistency over long period of time. The experiments and results of evaluating sMRT on actual smart home data are shown in Chapter 6.

5.1 sMRT: sensor-vectorized Multi-Resident Tracking

sMRT is an unsupervised multi-resident tracking solution based on sensor vectorization. The sMRT is composed of two phases: a learning phase and a tracking phase. During the learning phase, we first construct a hyper-dimensional measurement space and trans-
late sensor observations into vector embeddings by mining the spatio-temporal relationship using the observed sensor sequence. We hypothesize that the movement of residents in the smart environment can thus be mapped to a set of point targets maneuvering in the measurement space. In this way, the resident state captures information about the point target dynamics. Hence, a dynamic model and measurement model can be constructed to predict the resident states and relate the resident states to the sensor observations. During the tracking phase, we utilize a Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter, a closed-form first order moment approximation to the multi-target Bayes filter, to predict and propagate the Probability Hypothesis Density (PHD) of the resident states. Finally, the resident states are derived by a maximum a posterior (MAP) estimator. Associations between sensor events and residents are computed by maximizing the likelihood of new sensor observation based on the posterior PHD of resident states.

5.1.1 Problem Statement

In sMRT, we formulate the MRT problem within the mathematical framework of FISST. The state of each resident in the smart environment is represented as a random vector $x$ in a hyper-dimensional state space $\mathcal{X}$. Thus, the state of all residents that are currently in the smart home at time step $k$ can be modeled as a random finite set (RFS) $X_k = \{x_1, x_2, \ldots, x_n\} \in \mathcal{F}(\mathcal{X})$, where $\mathcal{F}(\mathcal{X})$ is the collection of all finite subsets of the state space $\mathcal{X}$. Each element $x_i$ ($1 \leq i \leq n$) of the RFS $X$ is a state vector of an active resident. The total number of active residents in the smart home, $n$ (i.e., the cardinality
$|X_k|$ of the RFS $X_k$), is a random variable defined on $\mathbb{Z}_0^+$. The movement of residents in the smart environment can thus be represented as multiple point targets maneuvering in the state space.

On the other hand, the sensor observations, derived from the sensor messages collected in a smart environment (as explained in Section 3.3), are translated into measurements of such point targets in the state space. Each measurement is represented as a vector $z$ in another hyper-dimensional space $\mathcal{Z}$. A measurement model, represented as a conditional probability density, $f(Z|X)$, is used to relate the measurements with the resident states. In this case, $X$ is the resident state RFS and $Z$ is a RFS representing the corresponding sensor observations ($Z \in \mathcal{F}(\mathcal{Z})$). In MRT, the measurements originate from two sources: false alarms caused by sensor faults or communication glitches, and actual measurements of one or more active resident (based on the state vector of the corresponding point target). Considering that the sensor events associated with residents are independently generated based on the whereabouts of each resident, we hypothesize that each point target (representing each active resident) generates its measurement independently. Hence, the actual measurement of an active resident can be formulated using the single target measurement model, represented as a conditional probability density, $f(z|x)$. The false alarms, also referred to as a clutter process, are modeled using a Poisson Point Process (PPP).

Similarly, with RFS as the models of resident states in a smart environment, an ideal dynamic model should capture the state updates of all residents, represented as a conditional probability density $f(X_k|X_{k-1})$, where $X_k$ and $X_{k-1}$ are the RFSs representing
the resident states at time step $k$ and $k - 1$, respectively. As the numbers of elements in both RFSs are unbounded, numerically computing the $f(X_k|X_{k-1})$ is intractable. To simplify the computation, we assume the independence of the resident during tracking and use the single resident dynamic model $f(x_k|x_{k-1})$ instead, where $x_k$ and $x_{k-1}$ are the state vectors of a resident at time step $k$ and $k - 1$, respectively.

With the dynamic model and measurement model constructed, sMRT calculates a posterior multi-target probability density of all resident states, $f_k(X)$, at time step $k$ based on the prior multi-target probability density $f_{k-1}(X)$ at the previous time step. By maximizing the posterior probability, sMRT derives the resident states in the state space, and the number of active residents, or the cardinality of the RFS $X_k$, can be simultaneously derived.

5.1.2 Learning Phase

The objective of the learning phase is to construct the measurement space, compute the vector embeddings of the sensors, and construct the dynamic and measurement models. In previous work, the dynamic model that encapsulates resident movement in a smart home was represented as a Markov chain, or a sensor graph, where the states of the Markov chain are mapped directly to the smart home sensors [7,9]. In a smart home with $q$ sensors, a total of $q^2$ transition matrix parameters, each representing the probability of a resident moving from one state (sensor location) to another, are estimated through counting [7,9] or a conditional least squares method [84]. However, in those approaches, annotated
sensor events and additional sensor layout information are required to make an accurate prediction. By mapping the states directly to sensors, these dynamic models perform state prediction based solely on the current resident state without taking into account resident states in any of the previous steps.

In contrast, we represent each sensor as a $m \times 1$ vector in a $m$-dimensional space $\mathcal{Z}$. The space $\mathcal{Z}$ represents the measurement space. The dimensionality $m$ of the measurement space is a hyper-parameter that can be chosen depending on the number and density of the sensors that are deployed in the smart home. The conditional probability of a resident transitioning from one sensor to another can be estimated using the distance between their vector representations in the measurement space $\mathcal{Z}$. In a smart home with $q$ sensors, a total of $q \cdot m$ parameters must be estimated. Selecting $m < q$ effectively injects dependencies between the conditional probabilities of a resident transiting between two sensors, a departure from earlier work in multi-resident tracking. By injecting these dependencies, a lesser amount of sensor data is needed to accurately learn the dynamic model parameters. Additionally, the model parameters are learned without the need for additional information such as the number of residents or sensor layout. With inspiration from word embedding used in natural language processing applications [112], we adopt a similar skip-gram model to leverage the co-occurrence of sensor events and train a generative model to learn the vector mapping between the sensors deployed in the smart home and the measurement space $\mathcal{Z}$. (see Section 5.1.2).

On the other hand, rather than mapping resident states directly to smart home sensors, we hypothesize that each resident’s movement can be represented as a point target
maneuvering at constant velocity in the measurement space $\mathcal{Z}$. This hypothesis can be considered as a relaxation of the Markov assumption in sensor graphs, where the velocities along all $m$ axes represent the information related to the resident states in all previous time steps and can be estimated during the tracking.

**Sensor Vectorization**

Consider a smart home where a total of $q$ binary sensors $(s_1, s_2, \ldots, s_q)$ are deployed. Sensors $s_i$ and $s_j$ are adjacent if a resident can travel from $s_i$ to $s_j$ without triggering another sensor $s_k$ ($i \neq j \neq k$). The goal of sensor vectorization is to find the corresponding vector representation $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_q \in \mathcal{Z}$ such that if two sensors are adjacent (they can be activated in sequence without activating other sensors), they are mapped to two vectors close to each other in the measurement space. In other words, the closer $\mathbf{z}_i$ and $\mathbf{z}_j$ are, the higher is the conditional probability of triggering sensor $s_i$ after sensor event $s_j$. As a result, we can further hypothesize that resident movement in the smart home is equivalent to a point target maneuvering in the measurement space.

In a smart home with a single resident, adjacent sensors always show up next to each other in the sensor sequence. In a multi-resident scenario, the recorded sensor sequence is a time-ordered collection of the active sensor messages associated with all residents in the smart home, possibly moving through different parts of the home. As a result, adjacent sensors are not necessarily next to each other in the sensor event sequence. However, they are more likely to show up within $c$ sensor messages apart, where $c$ is an integer that can be selected based on the expected number of smart home residents. Thus, we construct a generative model that predicts the probability of two sensors being adjacent parameterized
by their vector representations in measurement space. This probability needs to fit the sensor pair’s co-occurrence observed in the recorded sensor sequence within a window of $c$ sensor messages.

Formally, given a sensor sequence containing $M$ sensor messages, $(t^{(1)}, s^{(1)})$, ..., $(t^{(M)}, s^{(M)})$, where $t^{(i)}$ is the time of the $i^{th}$ sensor message and $s^{(i)}$ is the corresponding sensor ID, we generate a training set where each sensor pair is observed within a window of $c$ sensor messages in the sensor sequence, as shown in Equation (5.1).

$$training\ set = \{(s^{(i)}, s^{(j)}) | 0 < j - i \leq c\}$$ \hspace{1cm} (5.1)

We construct a generative model (as shown in Figure 5.1) that predicts the probability of a sensor pair, $s$ and $s'$, being adjacent, denoted as $P(s|s') = P(s'|s)$. The training objective of the model is to map sensors $s_1, \ldots, s_q$ into vectors $z_1, \ldots, z_q \in \mathcal{Z}$ so that the average log likelihood $\mathcal{L}$, as shown in Equation (5.2), is maximized in the training set.
\[ \mathcal{L} = \frac{1}{M} \sum_{i=1}^{M} \sum_{0 < j - i \leq c} \log P(s^{(j)}|s^{(i)}) \]  

(5.2)

The probability of sensor \( s_i \) being adjacent to sensor \( s_j \) can be defined using a SoftMax function based on a score assigned to them, as shown in Equation (5.3).

\[
P(s_j|s_i) = \frac{\exp(score(s_j|s_i))}{\sum_{k=1}^{q} \exp(score(s_k|s_i))}
\]

(5.3)

The score value \( score(s_j|s_i) \) needs to be larger when the distance between the corresponding vectors is smaller. We use a dot product as the similarity measure that defines the score function, as shown in Equation (5.4).

\[
score(s_j|s_i) = score(s_i|s_j) = z_i \cdot z_j^T
\]

(5.4)

In a smart home containing a small number of sensors, the vector representations of sensors in the measurement space can be learned directly using SoftMax cross-entropy loss. To reduce the large computational cost of directly learning vector representations for a large number of sensors, noise contrast estimation (NCE) [113] is employed.

A example vector mapping of all the sensors in Kyoto testbed into a 3-dimensional measurement space is shown in Figure 5.2. The sensors that are adjacent according to the floor plan and sensor location, for example M39 and M49 in Koyoto (Figure 3.8), are located near each other in the measurement space.

**Dynamic Model and Measurement Model**

With each sensor in the smart home mapped into the measurement space, we use a constant velocity model of a point target maneuvering in the measurement space to
approximate the movement of each resident in the smart home. The state vector of each resident is a \((2m + 1) \times 1\) vector \(\mathbf{x} = \begin{bmatrix} x^T & v^T & r \end{bmatrix}^T\), where \(x\) is an \(m \times 1\) vector representing the location of the resident in space \(Z\), \(v\) is an \(m \times 1\) vector representing the velocity of the resident, and \(r\) is an integer representing the resident ID. Given the state of the resident, \(x'\), the resident state \(x\) at the next time step can be estimated using the linear equation as shown in Equation (5.5). Here, \(F\) represents the linear motion multiplier, \(G\) represents the linear error multiplier, and \(w\) represents the velocity error.

\[
x = F \cdot x' + G \cdot w
\]  

(5.5)
If $w$ can be modeled using a Gaussian distribution, the probability distribution of
the resident state at the next time step can be expressed using a linear Gaussian model as
in Equation 5.6. Here, $Q$ is the resulting covariance matrix.

$$f(x|x') = \mathcal{N}(x; Fx', Q)$$ (5.6)

Residents maneuver in the measurement space. Thus, sensor observations (repres-
ented by the corresponding sensor vectors) offer a noisy measurement of true resident
states. If we assume that such measurement errors can be modeled as a Gaussian dis-
tribution with zero mean and a covariance matrix $R$, the relationship between a sensor
observation $z$ and the state vector $x$ of the resident can be represented using a linear
Gaussian model as shown in Equation 5.7 with linear multiplier $H$.

$$f(z|x) = \mathcal{N}(z; H \cdot x, R)$$ (5.7)

Movement mapped from resident actual trajectories to the measurement space may
not strictly follow the constant velocity assumption. However, with the help of the GM-
PHD filter and track maintenance algorithm introduced in Section 4.2, errors between
reality and the constant velocity assumption can be captured by the Gaussian noise in
the dynamic and measurement models shown in Equations (5.6) and (5.7). Thus, the
GM-PHD filter can correct these errors based on new sensor observations obtained at each
step.
5.1.3 Tracking Phase

During the tracking phase, a series of sensor observations is extracted from the sensor event stream by taking a snapshot of active sensors in the smart home every time a sensor is activated. Each active sensor represents a measurement of a resident in the smart home.

By replacing each active sensor with its vector representation in the measurement space, we define an observation set as $Z_k = \{z_1, \ldots, z_{n_z}\}$ at time step $k$, where $n_z$ is the number of active sensors and each element $z_i$ is the vector representation of the corresponding sensor.

Among these $n_z$ sensor observations, some are accurate measurements of active residents and some are false alarms (or clutter) due to communication errors or sensor failures. Alternatively, some residents may still be at home but may not be currently detected by the sensors. Thus, instead of creating a one-to-one mapping between each sensor observation and the corresponding resident, we also need to consider the possibilities of a
new resident entering the home, an existing resident leaving the home, residents not being detected, sensor observations not being associated with any resident, and one-to-many or many-to-one associations between sensor observations and residents.

To model all of these possibilities, we use a Gaussian mixture probability density (GM-PHD) filter [114] that propagates the first-order moment of the multi-target probability density, or the probability hypothesis density (PHD), based on the dynamic and measurement models constructed during the learning phase. Additionally, we propose clustering-based track maintenance to associate the PHD predicted by the GM-PHD filter with resident identifiers to detect new residents while maintaining the traces of existing residents. Finally, each sensor observation, represented as a vector in the measurement space, is associated with the resident that is most likely to generate the observation. The steps of the tracking phase are illustrated in Figure 5.3.

**Multi-Target Bayes Filter and PHD Filter**

We start by considering the trivial case with a single resident. The single-target Bayes filter tracks the location of a single target in a space with noisy observation. At each time step, a single sensor observation is acquired about the location of the target. Since there are error and noise in the sensor observation, the actual location of the target is estimated by evaluating the amount of error in the sensor observation and the expected movement of the target in the space. The sensor observation is represented by a measurement model, formulated as a conditional probability density $f(z|x)$. The measurement model predicts the likelihood of a sensor observation $z$ given the target state $x$. The movement of the target in the space is represented by a dynamic model, formulated as a conditional probability
density \( f(x|x') \).

We start by considering a single-target scenario, where one target is present in the space. At each time step, a single sensor observation is acquired. The dynamic model of the target is presented in the form of a conditional probability density \( f(x|x') \), which predicts the next state of the target, \( x \), based on its current state, \( x' \). The measurement model is also formulated as a conditional probability density, \( f(z|x) \), representing the likelihood of a sensor observation \( z \) given the target state \( x \). Provided with the probability distribution of target state \( f_{k-1}(x) \) at time \( k - 1 \), the target state probability distribution at time step \( k \), \( f_k(x) \), can be calculated using probability theory’s chain rule and Bayes’ rule according to Equations (5.8) and (5.9).

\[
f_{k|k-1}(x) = \int f(x|x') f_{k-1}(x') dx'
\quad (5.8)
\]

\[
f_k(x) = \frac{f(z|x) f_{k|k-1}(x)}{\int f(z|x') f_{k|k-1}(x') dx'}
\quad (5.9)
\]

Equation 5.8 represents the predictor that generates the target state probability density, \( f_{k|k-1}(x) \), based on the target state at time step \( k - 1 \) and the dynamic model. Equation 5.9, representing the corrector, updates the probability density generated by the predictor so that the likelihood of the sensor observation at time step \( k \) is maximized. If the dynamic model and the measurement model are both linear and take the form of Gaussian distributions, the single target Bayes filter can be reduced to the traditional Kalman filter [115]. Otherwise, sequential Monte Carlo sampling methods, also known as particle filters, can be used to numerically solve both equations [116].
Figure 5.4: The propagation pipeline of single-target Bayes filter, multi-target Bayes filter and PHD filter.
The situation gets more complicated when there are multiple targets to track and the number of targets is unknown. In this case, the states of all targets in the space are modeled as a set \( X = x_1, \ldots, x_n \in \mathcal{F}(\mathcal{X}) \) where \( \mathcal{F}(\mathcal{X}) \) represents the collection of all finite subsets of the state space \( \mathcal{X} \). Since the number of elements \( n \) in the set \( X \) is a random variable in \( \mathbb{Z}_+ \), the set \( X \) is called a random finite set (RFS). Building on FISST [111], Mahler [115] demonstrated that a multi-target probability density function \( f(X) \) can be defined for the RFS \( X \) and propagated similarly to the single-target Bayes filter shown in Figure 5.4. The equations for the predictor and the corrector filter are shown in Equations (5.10) and (5.11), respectively.

\[
f_{k|k-1}(X) = \int f(X|X')f_{k-1}(X')dX' \tag{5.10}
\]

\[
f_k(X) = \frac{f(Z_k|X)f_{k|k-1}(X)}{\int f(Z_k|X')f_{k|k-1}(X')\delta X'} \tag{5.11}
\]

Equation 5.10 mirrors Equation (5.8) by replacing the single target state variable \( x \) with the multi-target state set \( X \), and the vector integral with a set integral. The multi-target dynamic model, represented by the conditional probability \( f(X|X') \), predicts the state distribution of all targets given the states of all targets at previous steps. In addition to the dynamics of persisting targets (i.e., the existing residents still in the space), the multi-target dynamic model must also consider the birth of a new target (i.e., a resident entering the space or becoming active) and the death of an existing target (i.e., a resident leaving the space or becoming inactive). The term \( Z_k = \{z_1^{(k)}, \ldots, z_{n_Z}^{(k)}\} \in \mathcal{F}(\mathcal{Z}) \) in Equation (5.11) is a set of \( n_Z^{(k)} \) sensor observations taken at time step \( k \). The multi-
target measurement model, \( f_k(Z|X) \), represents the probability density of a set of sensor observations, \( Z \), when the states of all residents in the space is characterized by the set \( X \). The multi-target measurement model encapsulates the information about the likelihood of the sensor observations being triggered by the existing targets in the space (i.e., an existing resident triggering a sensor), the possibility that an existing target fails to be detected by any sensor (i.e., an “active” resident moving to a location not covered by any sensor), and the cases where some sensor observations belong to the clutter process due to sensor failures or communication errors.

The major challenge of the multi-target Bayes filter is the exponential growth of the computational complexity during the numerical calculation of the set integrals and multi-target probability density in Equations (5.10) and (5.11) when the cardinality of \( X \) increases. To reduce the computational complexity, Mahler [117] proposed a PHD filter that propagates the first-order moment, or probability hypothesis density (PHD), instead. Thus, both the predictor and the corrector equation can be solved in polynomial time using sequential Monte Carlo sampling methods. To further simplify the computation of the PHD filter, Vo and Ma [114] proposed the GM-PHD filter, which is a closed-form solution of the PHD filter where each PHD is represented using Gaussian mixtures.

The PHD of an RFS \( X \), denoted \( D(x) \) (\( x \in \mathcal{X} \)) and parameterized by a multi-target probability density function \( f(X) \), is a probability density function whose integral on any region \( S \) of the state space \( \mathcal{X} \) (\( S \subseteq \mathcal{X} \)) equals the expected number of targets, \( N(S) \), in the region \( S \) [115]. Mathematically, the PHD is defined as shown in Equation (5.12), or equivalently in Equation (5.13).
Figure 5.5: The relationship between PHD of resident states and the probability density of each resident state. In the graph, \( f(x_1) \) and \( f(x_2) \) are the probability density of the states of residents R1 and R2 currently being “active” in the smart home. \( D(x) \) is the equivalent PHD of the resident states, \( X = \{x_1, x_2\} \). In this example, the integral of \( D(x) \) is 2.

\[
\int_S D(x)dx = N(S) = \int |X \cap S| \cdot f(X)\delta X
\]  

(5.12)

\[
D(x) = \int f(\{x\} \cup W)\delta W
\]  

(5.13)

Figure 5.5 illustrates an example of the PHD of resident states in a smart home with two residents, R1 and R2. Vectors \( x_1 \) and \( x_2 \) represents the states of R1 and R2, respectively. On the left, we plot the probability densities, \( f(x_1) \) and \( f(x_2) \), of the states of residents R1 and R2. Because in this case, we know there are two residents in the
smart home, the probability density of the RFS $X = \{x_1, x_2\}$ is shown in Equation (5.14), assuming $x_1$ and $x_2$ are independent.

$$f(X) = \begin{cases} f(\{x_1, x_2\}) = f(x_1)f(x_2), & \text{if } |X| = 2 \\ 0, & \text{otherwise} \end{cases}$$

(5.14)

The PHD of resident states, $D(x)$, illustrated on the right side of Figure 5.5, can be computed according to Equation 5.17.

$$D(x) = \int f(\{x\} \cup W)\delta W$$

(5.15)

$$= \int f(x_1)f(x_2)dx_1 + \int f(x_1)f(x_2)dx_2$$

(5.16)

$$= f(x_1) + f(x_2)$$

(5.17)

As a result, the integral of $D(x)$ over the whole space equals to the number of residents, which is 2 in this example.

Figure 5.6 illustrates the PHD filter propagation pipeline adapted to the context of multi-resident tracking. The GM-PHD filter is composed of a predictor and a corrector. Given the PHD of multiple residents at time step $k - 1$, $D_{k-1}(x)$, the predictor estimates the multi-resident PHD at time step $k$, or $D_{k|k-1}(x)$, based on the linear Gaussian dynamic model in Equation (5.6). The corrector then refines the predicted PHD, $D_{k|k-1}(x)$, based on the measurement model and sensor observations, $Z_k$. The output of the corrector is the Bayes optimal estimation of the posterior multi-resident PHD at time step $k$, $D_k(x)$, which can be used to associate sensor events with residents in the smart home. The
Figure 5.6: The propagation of multi-target PHD in a GM-PHD filter implemented in sMRT to solve the multi-resident tracking problem.
PHD propagation equation of the predictor and corrector can be derived if the following assumptions are true.

- Each target (resident) evolves and generates sensor observations independently of the others. Thus, we can predict the state of each resident independently using the dynamic model characterized by the conditional probability distribution $f(x|x')$. Similarly, we can also estimate the likelihood of a sensor observation triggered by some resident in the space independently using the measurement model characterized by the conditional probability distribution $f(z|x)$.

- The clutter process is a Poisson point process (PPP) and is independent of target-originated measurements. The number of sensor observations that are not associated with any resident in the smart home at any given time step follows a Poisson distribution with parameter $\lambda_c$, while each false alarm follows a spatial distribution $c(z)$ $(z \in Z)$.

- The distribution of resident states governed by PHD $D_k(x)$ at any time step $k$ is Poisson.

- At every time step, an existing resident may leave the home or become “inactive”. The probability of an existing resident still being “active” in the following time step is characterized by the target survival probability $p_a$. Similarly, we can use the target detection probability $p_d$ to characterize an “active” resident who fails to be detected by any sensor. In order to derive the PHD filter equations, both the survival probability $p_a$ and the detection probability $p_d$ are constant and independent of
resident states.

- At time step $k$, a PHD, $b_k(x)$, representing the target birth process (i.e., a resident being “active” again or a new resident entering the home) is injected into the predictor. The integral of the target birth PHD, according to the definition of PHD in Equation (5.12), equals the expected number of new residents at the corresponding time step.

With the above assumptions, the equations for the predictor and corrector in the PHD classifier can be derived using Equations (5.10) and (5.11), respectively. The posterior PHD of the predictor, as shown in Equation (5.18), is the summation of the target birth PHD, $b_k(x)$, and the predicted PHD, $D_{s,k-1}(x)$, of all persisting residents. The term $D_{s,k-1}(x)$ can be calculated using Equation (5.19). The posterior PHD of the corrector, as shown in Equation (5.20), is composed of two terms. The first term, $(1 - p_d)D_{k|k-1}(x)$, corresponds to the PHD of residents that are not detected by any sensor. The second term, $D_{d,k}(x)$, is the PHD of the detected residents that are corrected by the sensor observations $Z^{(k)}$ at time step $k$. The term $D_{d,k}(x)$ can be calculated according to Equation (5.21).

\[
D_{k|k-1}(x) = b_k(x) + D_{s,k-1}(x) \tag{5.18}
\]

\[
D_{s,k-1}(x) = \int p_s f(x|x')D_{k-1}(x')dx' \tag{5.19}
\]

\[
D_k(x) = (1 - p_d)D_{k|k-1}(x) + D_{d,k}(x) \tag{5.20}
\]
\[ D_{d,k}(x) = \sum_{z \in Z_k} \frac{p_d f(z | x) D_{k|k-1}(x)}{\lambda_c(z)} + \int p_d f(z | x') D_{k|k-1}(x') dx' \]  

(5.21)

In addition to the assumptions of the PHD filter, the GM-PHD filter further makes the following assumptions.

- The dynamic model and the measurement model can be represented as a linear Gaussian model as shown in Equations (5.22) and (5.23).

\[ f(x | x') = \mathcal{N}(x; Fx, Q) \]  

(5.22)

\[ f(z | x) = \mathcal{N}(z; Hx, R) \]  

(5.23)

- The intensity of the target birth PHD, \( b_k(x) \), can be represented in the form of a Gaussian mixture as shown in Equation (5.24), where \( J_{b,k} \) is the number of Gaussian components in the target birth PHD, and \( w_{b,k}^{(i)}, m_{b,k}^{(i)} \) and \( P_{b,k}^{(i)} \) are the weight, mean vector and covariance matrix of the \( i \)th Gaussian component in the target birth PHD.

\[ b_k(x) = \sum_{i=1}^{J_{b,k}} w_{b,k}^{(i)} \mathcal{N}(x; m_{b,k}^{(i)}, P_{b,k}^{(i)}) \]  

(5.24)

Suppose that the PHD of resident states at time step \( k - 1 \) is a Gaussian mixture as shown in Equation (5.25).

\[ D_{k-1}(x) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}(x; m_{k-1}^{(i)}, P_{k-1}^{(i)}) \]  

(5.25)
By substituting Equations (5.22), (5.24) and (5.25) into Equation (5.18), the posterior PHD of the predictor can be represented in the form of a Gaussian mixture as shown in Equation (5.23).

\[ D_{k|k-1}(x) = \sum_{i=1}^{J_{b,k}} w_{b,k}^{(i)} \mathcal{N}(x; m_{b,k}^{(i)}, P_{b,k}^{(i)}) \]

(5.26)

\[ + p_d \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}(x; m_{k-1}^{(i)}, (Q + FP_{k-1} F^T)) \]

For simplicity, we rewrite the posterior PHD of the predictor as shown in Equation (5.27).

\[ D_{k|k-1}(x) = \sum_{i=1}^{J_{k|k-1}} w_{k|k-1}^{(i)} \mathcal{N}(x; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) \]

(5.27)

By substituting Equations (5.23) and (5.27) into Equation (5.20), the posterior PHD of all residents at time \( k \) can be calculated as shown in Equation (5.28).

\[ D_k(x) = (1 - p_d) \sum_{i=1}^{J_{k|k-1}} w_{k|k-1}^{(i)} \mathcal{N}(x; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) \]

(5.28)

\[ + \sum_{z \in Z_k} \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(z) \mathcal{N}(x; m_k^{(j)}(z), P_k^{(j)}) \]

In Equation (5.28),

\[ w_k^{(j)}(z) = \frac{p_d w_{k|k-1}^{(j)}(z)}{\lambda_c(z) + p_d \sum_{i=1}^{J_{k|k-1}} w_{k|k-1}^{(i)} q_k^{(i)}(z)} \]

(5.29)

\[ q_k^{(j)}(z) = \mathcal{N}(z; Hm_k^{(j)}(z), R + HP_k^{(j)}(z)H^T) \]

(5.30)
\[ m^{(j)}_k(z) = m^{(j)}_{k|k-1} + K^{(j)}_k(z - Hm^{(j)}_{k|k-1}) \]  
(5.31)

\[ P^{(j)}_k = (I - K^{(j)}_k H)P^{(j)}_{k|k-1} \]  
(5.32)

\[ K^{(j)}_k = P^{(j)}_{k|k-1} H^T (HP^{(j)}_{k|k-1} H^T + R)^{-1} \]  
(5.33)

According to the GM-PHD filter predictor and corrector shown in Equations (5.26) and (5.28), the number of Gaussian components in the PHD grows from \( J_{k-1} \) at time step \( k - 1 \) to \( (J_{b,k} + J_{k-1})|Z^{(k)}| \) at time step \( k \). To balance computational complexity with accuracy, a maximum number of \( J_{\text{max}} \) Gaussian components with the highest weights are kept and propagated through time.

**Track Maintenance**

Given the posterior PHD at time step \( k \), we propose a clustering-based track maintenance algorithm that estimates the state of each resident, assigns identifiers to the newly-identified residents, and associates sensor observations with each resident based on the state probability distribution of each identified resident. According to the definition of PHD (Equation 5.13), the expected number of residents in the smart home can be calculated by integrating the PHD over the entire state space, as shown in Equation (5.34).

\[ N_k = \int \sum_{i=1}^{J_k} w^{(i)}_k N(x; m^{(i)}_k, P^{(i)}_k) dx = \sum_{i=1}^{J_k} w^{(i)}_k \]  
(5.34)

We first assume that, at any time step, there is at most one newly-detected resident. Thus, during the predictor step, we can assign a new resident identifier to the resident ID field of the Gaussian mean state vectors for the target birth PHD. Given the measurement
model and the dynamic model defined in Section 5.1.2, the resident identifier in the mean vector of each Gaussian component will remain unchanged while the Gaussian components are propagated in time through the GM-PHD filter. By grouping the Gaussian components that share the same resident identifier in the mean vector, the state probability distribution of each resident can be derived.

We now consider the case that multiple residents, $R_1^{(k)} \ldots R_n^{(k)}$, enter the smart home at time $k$. As we assign a single resident identifier, $r^{(k)}$, to all Gaussian components in the target birth PHD, the Gaussian components of the PHD, representing the states of all residents entering the smart home, share the same resident identifier $r^{(k)}$. As the residents move through time, the cardinality of the PHD will eventually approximate the actual number of residents, $N^{(k)}$, who enter the home. As a result, when tracking each resident $R_i^{(k)}$, the Gaussian components representing the PHD of those $N^{(k)}$ residents need to be separated into $N^{(k)}$ clusters with a unique resident identifier assigned to the Gaussian components for each cluster.

In sMRT, we introduce a clustering-based track maintenance algorithm that monitors the integral of the PHD associated with each resident identifier. The track maintenance algorithm is an iterative six-step process as follows.

1. Given the PHD with resident identifier $r$ in the form of a Gaussian mixture as shown in Equation (5.35), calculate the number of expected residents $N'_{k,r}$ as shown in Equation (5.36).
\[ D_{k,r}(x) = \sum_{i=1}^{J_{k,r}} w_{k,r}^{(i)} N \left( x; m_{k,r}^{(i)}, P_{k,r}^{(i)} \right) \]  

(5.35)

\[ N_{k,r}' = \lceil N_{k,r} - 0.5 \rceil = \left\lceil \sum_{i=1}^{J_{k,r}} w_{k,r}^{(i)} - 0.5 \right\rceil \]  

(5.36)

2. Initialize the center of \( N_{k,r}' \) clusters randomly as \( \alpha_1, \ldots, \alpha_{N_{k,r}'} \).

3. For each cluster, find the Gaussian components in \( D_{k,r}(x) \) with the smallest distance between the mean of the Gaussian component and the center of the corresponding cluster. Assign those Gaussian components to the cluster so that the summation of the weights of all those Gaussian components does not exceed \( N_{k,r}/N_{k,r}' \). If there are Gaussian components left not assigned to any cluster, assign each of these to the nearest cluster determined by the distance between the center of the cluster and the mean of the Gaussian component.

4. Update the cluster center \( \alpha_j \) to be the weighted mean of all Gaussian components assigned to the cluster, as shown in Equation (5.37).

\[ \alpha_j = \frac{1}{\sum_{i=1}^{J_{k,r,j}} w_{k,r,j}^{(i)}} \sum_{i=1}^{J_{k,r,j}} w_{k,r,j}^{(i)} m_{k,r,j}^{(i)} \]  

(5.37)

In Equation (5.37), \( J_{k,r,j} \) represents the number of Gaussian components assigned to cluster \( j \). The \( w_{k,r,j}^{(i)} \), \( m_{k,r,j}^{(i)} \) terms represent the weight and mean of those Gaussian components.

5. Repeat steps 3 and 4 until there are no further changes to the association between Gaussian components and clusters, or a maximum number of iterations is reached.
6. With the Gaussian components segregated into $N'_{k,r}$ clusters, a new resident identifier is assigned to each cluster and is inserted into the resident ID field in the mean vector of each Gaussian component assigned to that cluster.

Finally, each sensor observation $\mathbf{z}_i \in \mathbb{Z}_k$ is associated with the resident ID $r$ so that the likelihood of producing the sensor observation $\mathbf{z}_i$ is maximized, as shown in Equation (5.38).

$$r = \arg \max_r \int f(\mathbf{z}_i | \mathbf{x}) \sum_{i=1}^{J_{k,r}} w_{i,k,r} \mathcal{N}(\mathbf{x}; \mathbf{m}_{i,k,r}^{(i)}, \mathbf{P}_{i,k,r}^{(i)}) \, d\mathbf{x}$$

$$= \arg \max_r \sum_{i=1}^{J_{k,r}} w_{i,k,r} \mathcal{N}(\mathbf{z}_i; \mathbf{Hm}_{i,k,r}^{(i)} \mathbf{R} + \mathbf{HP}_{i,k,r}^{(i)} \mathbf{H}^T)$$

(5.38)

5.2 sMRT-ML: sMRT with Unsupervised Training

The multi-resident tracking performance of sMRT relies on the dynamic model and the measurement model constructed during the learning phase. As presented in the previous section, sMRT is built upon the hypothesis that resident movement in the smart environment can be approximated with a point target maneuvering in the measurement space with a constant velocity model. The vector embeddings of sensor observations are trained with an unsupervised approach by mining spatio-temporal relation of sensors in the recorded sensor sequence. However, considering the generative nature of the Bayes filter process in the tracking phase, we demonstrate that the dynamic and measurement model can be constructed by maximizing the likelihood of the observed sensor events.
The unsupervised training process of sMRT-ML is illustrated in Figure 5.7. Initially, the parameters of the dynamic model and measurement model are initialized in the same way as the sMRT learning phase. At time step $k$, based on the previous sensor observations from time step 1 to $k-1$, $Z^{(1:k-1)}$, the PHD of residents in the smart environment, $D_{k|k-1}(x)$, can be estimated according to the GM-PHD filter predictor, as shown in Equation 5.39.

$$D_{k|k-1}(x) = b_k(x) + \int p_s f(x|x')D_{k-1}(x')dx'$$  \hspace{1cm} (5.39)

At time step $k$, the set of sensor observations $Z^{(k)}$ is composed of false alarms generated by the clutter process (with Poisson parameter $\lambda_c$ and spacial distribution $c(z)$) and the positive measurements of detected active residents, governed by the detection proba-
bility $p_S$ and the measurement model $f(z|x)$. Hence, the PHD of sensor observations at
time step $k$ can be computed according to Equation 5.40.

$$D_k(z) = \int p_S f(z|x) D_{k|k-1}(x)d\mathbf{x} + \lambda_c c(z)$$ (5.40)

According to the PHD filter assumptions stated in Section 5.1.3, $\lambda_c$ and $c(z)$ in
Equation 5.40 are both constant. Due to the fact that the measurement model $f(z|x)$
is linear Gaussian and $D_{k|k-1}(x)$ can be represented in the form of a Gaussian mixture,
the integral $\int p_S f(z|x) D_{k|k-1}(x)d\mathbf{x}$ is in the form of a Gaussian mixture, and hence, is
differentiable.

The goal of the unsupervised learning process is to maximize the likelihood of the
sensor observations. Given the observation set $Z^{(k)}$, the parameters of the dynamic model
and measurement model can be found by minimizing the loss function, denoted $L(Z^{(k)})$.

In traditional single-target probability density estimation, the negative log-likelihood, as
shown in Equation 5.41, is commonly used [118].

$$L_{\text{log}}(Z^{(k)}) = -\frac{1}{|Z^{(k)}|} \sum_{z \in Z^{(k)}} \log D_k(z)$$ (5.41)

However, in a multi-target scenario, the integral of PHD, $D(z)$ in this case, equals
the number of sensor observations. As a result, parameters resulting from a simple log-
likelihood loss function based on the observed sensor measurements will cause an inflated
estimation of the PHD over time. To serve as a counterweight to the negative log-likelihood
generated by the positive sensor measurements, we append the log-likelihood of negative
samples to the loss function. Such negative samples can be generated according to a
particular distribution in the space, e.g. uniform distribution. In MRT, since the measurements are generated at the vector location of the sensor embeddings, we use the sensor embeddings that are not contained in the current observation set $Z^{(k)}$ as negative samples, resulting in a loss function as shown in Equation 5.42.

$$
\mathcal{L} = \frac{1}{|S|} \sum_{z \in S} (-1)^{1(z \in Z^{(k)})} \log D_k(z)
$$

In Equation 5.42, $S$ is the set of all sensor vectors in the measurement space and $1(x)$ is an indicator function, which returns 1 if $x$ is true and 0 otherwise.

Parameters in sMRT have physical meanings. For example, $F$ is the linear multiplier and the values of each elements within the multiplier are not bounded. In contrast, $p_s$ and $p_d$, representing the probability of target survival and target detection, are both bounded between 0 and 1. $R$ and $Q$, representing the covariance matrices for the linear Gaussian dynamic model and linear Gaussian measurement model, respectively, need to be positive determinant. As a result, during the discriminative training, the values in the linear multiplier $F$ can be estimated via back propagation directly, while meta-parameters and transform functions are needed for bounded model parameters. Thus, in sMRT-ML, as $p_s$ and $p_d$ are bounded between 0 and 1, we define two meta-parameters, $\theta_s, \theta_d \in \mathbb{R}$ such that $p_s = \text{sigmoid}(\theta_s)$ and $p_d = \text{sigmoid}(\theta_d)$, where the sigmoid function $\text{sigmoid}(x)$ is defined according to Equation 5.43.

$$
\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}
$$

In sMRT-ML, the covariance matrices $R$ and $Q$ are represented by their Chelosky
decomposed lower triangular matrices $R_{\text{chol}}$ and $Q_{\text{chol}}$, respectively. Thus, the covariance matrices $R$ and $Q$ can be constructed using $R_{\text{chol}}$ and $Q_{\text{chol}}$ according to Equations (5.44) and (5.45).

$$R = R_{\text{chol}} R_{\text{chol}}^T$$ (5.44)

$$Q = Q_{\text{chol}} Q_{\text{chol}}^T$$ (5.45)

During the training process of sMRT-ML, the parameters of the dynamic model and measurement model may not fit the recorded sensor observation. As a result, the computation of the GM-PHD filter’s corrector and predictor may exhibit numerical instability when a simplified covariance matrix update equation, as shown in Equation (5.32), is used to reduce computation. To resolve the numerical instability, we adopt the following optimization strategies. First, instead of using the simplified covariance update equation, a Joseph form, as shown in Equation (5.46), is used instead.

$$P^{(j)}_k = (I - K_k^{(j)} H)P^{(j)}_{k|k-1}(I - K_k^{(j)} H)^T + K_k^{(j)} R K_k^{(j)T}$$ (5.46)

Furthermore, instead of propagating the covariance during the numerical computation, the Chelosky decomposed lower triangular matrices are propagated for each Gaussian component in the posterior PHD of the corrector and the predictor. During the training, we also noticed that the covariance matrices may become non-positive determinant due to rounding errors in numerical computation, leading to the failure in the Chelosky decomposition. To improve the stability of the training process, we monitor the determinant of
the covariance matrices, and a small randomized positive diagonal matrix is added to the covariance matrix to mitigate these corner cases.

5.3 Track Consistency

From the experimental results presented in Section 6.5, we discovered that sMRT has a tendency to renew new resident IDs, causing an increase in target mismatches. After investigating these mismatch scenarios, we unveil two possible causes to the problem: the split of an existing track when the corresponding weights of the PHD overshoot the target and the creation of new track due to the “active” resident not being observed by sensors. Hence, additional optimization methods need to be introduced to improve track consistency.

During the track maintenance introduced in Section 5.1.3, an existing track can be split if the total weights of the Gaussian components in the PHD, rounded to the nearest integer, exceeds 1. However, due to the false alarms and the possibility that a new sensor is triggered before a previous sensor returns to an inactive state, it is possible for the total weights to overshoot the optimal values within a short time period, causing the spawn of a new track.

To compensate for this issue, we adopt a strategy of delayed track splitting. Instead of splitting the track as soon as the total weights of the Gaussian components in the PHD exceed 1, we keep a record of the total weights. If the increase of the track number is confirmed over multiple time steps, the Gaussian components of the target PHD then go
through the iterative clustering process, as depicted in Section 5.1.3, and a new track is spawned.

A new resident ID originating from a birth PHD overtaking an existing “active” resident is the second cause of track ID mismatches observed during the experiments. In sMRT and sMRT-ML, the balance between detecting a new resident and maintaining the track of an existing “active” resident is determined by the relationship between the weights of birth PHD, $b(x)$, and the target detection probability, $p_d$. Let us consider the situation when an “active” resident, denoted R1, moves to an area not covered by the ambient sensors, while another resident, denoted R2, actively performs an activity that results in a lot of sensor events. Based on the sMRT filtering process, the PHD corresponding to resident R1 will decay exponentially according to the corrector, as there is no associated sensor observations during those steps. When resident R1 moves back to a region detected by the ambient sensors, there are two possible outcomes. Either sMRT recognizes and attributes the sensor observation to R1, or the algorithm creates a new resident ID, R3. Whether a new resident ID is created or not depends on the weights of the PHD corresponding to R1 and the weights of the birth PHD. If the weights of the birth PHD are higher, a new ID’ will be assigned, resulting in an target ID mismatch error.

However, sMRT should not simply penalize the weights of a birth PHD. If those weights are too small, many time steps will pass before sMRT or sMRT-ML can confirm the existence of a new resident, as the filter needs to evaluate the potential matching between sensor observations and the resident dynamic model. If a resident is observed with a small number of time steps, he/she may not have the chance to be confirmed by
the multi-resident tracker, leading to a high rate of target misses. On the contrary, if sMRT sets the birth weights high, due to the exponential decay in the weights of the PHD corresponding to the "active" resident, a new resident ID is more likely to be assigned, resulting in a shorter track length and a high rate of target mismatch errors.

Instead, we opt to join the individual tracks after each track has been identified. If a new track is spawned at the same location as the death of a previously "active" resident within a certain time window, we assume that both tracks correspond to the same resident. In our experiments, we vary the length of the time window and study its impact on the resident ID mismatch. This behavior is analyzed in Section 6.5.

5.4 Summary

In this section, we formalize the multi-resident tracking problem in the framework of FISST. The movement of each smart environment resident is represented as a point target maneuvering in a latent measurement space. We construct the dynamic model and the measurement model of these point targets by applying unsupervised learning approaches to the unannotated sensor observations.

We first introduced sMRT, a multi-resident tracking algorithm based on sensor vectorization. By mining the spatio-temporal correlation exhibited in the sensor sequence, we construct the measurement space by finding vector embeddings for all the sensors deployed in the smart home. We further hypothesize that the maneuver of the point targets in the measurement space can be approximated with a constant velocity model, and sMRT oper-
ates on this assumption. Based on this foundation, a GM-PHD filter is used to propagate the PHD of resident states, modeled as a RFS, through time. In addition, we propose a clustering-based track maintenance algorithm to manage the propagation of resident identifiers and the spawn of new tracks in case multiple residents are identified as moving together.

We further improved sMRT by leveraging the generative nature of the sMRT filter process. We proposed an unsupervised discriminative learning process to train the model parameters. During the unsupervised learning process, the PHD of sensor observations is estimated based on the resident states and is compared to the recorded sensor observations from the data. In addition, we proposed additional optimization steps to reduce the track mismatch errors observed during our experiments.

To evaluate the proposed sMRT and sMRT-ML, we used two annotated datasets from WSU CASAS group, as mentioned in Section 3.1. The experimental results are presented in Chapter 6 followed by a discussion about the performance and limitations of the proposed methods.
In this chapter, we evaluate the proposed unsupervised Multi-Resident Tracking using sensor events recorded in real-life smart environments. To evaluate the performance of sMRT, we consider two other methods, nearest neighbor with sensor graph (NN-SG) and global nearest neighbor with sensor graph (GNN-SG), as baseline for comparison. Both methods, presented in Section 6.1, are built upon the sensor graph derived from additional information about the floor plan and sensor location in the smart environment. The output of the tracking algorithms are compared against the ground truth annotation generated by our lab annotators, as mentioned in Section 3.4.

The performance metrics used to evaluate the multi-resident tracking solutions is described in Section 6.2. Phrased as a data association problem, the performance of multi-resident tracking is typically measured using traditional multi-class classification metrics, such as accuracy and Hamming loss. In addition, we also compare alternative multi-resident tracking approaches based on the estimation accuracy of the number of “active” residents in the smart environment. Finally, we adapt the multi-object tracking accuracy (MOTA), commonly used for multi-object tracking in video surveillance application to the multi-resident tracking problem, and propose the multi-resident tracking accuracy (MRTA). By focusing on the error categories, including target misses, false positives, and target identifier mismatch errors, MRTA provides additional statistics and insights to debug and improve the algorithm.
6.1 Baseline Methods: NN-SG and GNN-SG

Both baseline methods, NN-SG and GNN-SG, employ a sensor graph as the corpus of information for multi-resident tracking. The sensor graph for each smart environment is generated using the floor plan and sensor location information. The residents are only allowed to travel between adjacent sensors. Thus, simple rules can be formed to handle the generation of a new target (a new resident entering the smart environment, or a resident becoming “active”), and the death of existing target (an “active” resident becoming “inactive” or leaving the environment). The association between existing “active” residents and sensor observations is handled by the nearest neighbor (NN) algorithm or the global nearest neighbor (GNN) algorithm for NN-SG and GNN-SG, respectively.

6.1.1 NN-SG: Nearest Neighbor with Sensor Graph

The NN-SG algorithm is an extension of the GR/ED algorithm proposed in earlier work by Crandall and Cook [9]. This algorithm employs a sensor graph, which is a bidirectional graph where the vertices of the graph are mapped to the smart home sensors. If the movement of a resident can trigger sensor $s_i$ and sensor $s_j$ consecutively without activating any other sensor in the smart home, sensor $s_i$ and sensor $s_j$ are adjacent in the sensor graph. As an example, Figure 6.1 illustrates the adjacencies between PIR motion sensors deployed in smart home TM004.

The weight on the directional edge from sensor $s_i$ to sensor $s_j$ in the sensor graph rep-
Figure 6.1: Sensor adjacency in smart home TM004. Adjacent sensors in smart home TM004 are joined with a blue line. Two sensors are adjacent if a resident can activate the sensors consecutively without triggering any other sensor in the smart home.

represents the conditional probability of a resident activating sensor $j$ after sensor $i$, $Pr(s_j|s_i)$.

In other words, the sensor graph is equivalent to a Markov chain, where the states of the Markov chain correspond to the nodes in the sensor graph. The weights on the directional
Figure 6.2: Transition matrix of the sensor graph in TM004. Each entry in the figure represents the probability of a resident moving from the sensor in the row to the sensor in the column. For example, value 0.47 in the top row represents the conditional probability of a resident activating sensor “BedroomAArea” after activating sensor “BedroomADoor”.

The edges of the sensor graph form the Markov chain transition matrix $P$, with $p_{ij} = Pr(s_j|s_i)$. If sensors $s_i$ and $s_j$ are not adjacent, $p_{ij} = 0$. Thus, given a recorded sensor sequence with
annotated labels for resident association, the values in the transition matrix can be estimated by maximizing the likelihood of generating the corresponding sensor sequence. As an example, based on the association labels provided by the annotator for the TM004 dataset, the estimated transition matrix is shown in Figure 6.2.

Given a sensor graph, NN-SG uses the nearest neighbor algorithm to associate sensor events with existing residents in the smart homes. However, in order to initiate a new track for a resident who just entered the house, or remove an old track when the corresponding resident leaves the house or becomes inactive, the following set of rules, originally developed in prior work of Crandall and Cook [9], are adopted.

**Rule of target death** An existing target (resident) is assumed to have left the house or become “inactive” if the target has not been detected by any sensors for a period of 50 sensor events (the parameter is suggested in GR/ED).

**Rule of target birth** If a sensor event is not found associated with any existing targets (residents), a new target will be formed and associated with the sensor event.

Whenever a new sensor event arrives, the NN-sg method first search through the existing active tracks. If an existing track is previously spotted by an adjacent sensor, the track is associated with the sensor event. However, if multiple existing tracks are found, the one with the highest likelihood of activating the current sensor is associated with the sensor event. When no existing track is found previously spotted by an adjacent sensor, according to the rule of target birth, a new target is spawned. NN-SG then check each existing target against target death rule and removes the dead target from the list before
moving on to the next sensor event.

6.1.2 GNN-SG: Global Nearest Neighbor with Sensor Graph

GNN-SG contrasts with NN-SG by associating targets with sensor observations. At each time step, GNN-SG generates a list of all possible one-to-one associations between the sensor observations (all active sensors) and the existing residents. A score is assigned to each association hypothesis by accumulating the probability of each existing track to the new sensor location according to the sensor graph. The hypothesis with the best score is selected, and any sensor observation that is not associated with any resident is considered the start of a new track and issued a new target identifier. The hypothesis selection process is equivalent to the binary assignment problem, which can be solved efficiently using the Hungarian algorithm [119].

6.2 Performance Metrics

In this section, we introduce three sets of performance metrics to evaluate the MRT algorithms presented in this dissertation. First, we evaluate the output of multi-resident tracking algorithms in the framework of multi-class classification. We use accuracy score, Hamming loss, precision, recall, and F1-score to compare the performance of each tracking algorithm against the ground truth. This set of metrics is commonly used in past research, especially when the number of residents in the smart home is assumed to be fixed. In addition, we also want to evaluate how well the tracking algorithm can estimate
the number of active residents in the smart home. Thus, the second metric we use is the average error in the number of active residents estimated by the residents. Finally, we adapt the multi-object tracking accuracy (MOTA), commonly used for multi-object tracking in video surveillance applications, to the multi-resident tracking problem, and propose a new multi-resident tracking accuracy (MRTA) metric. By focusing on the error categories, including target misses, false positives, and target identifier mismatch errors, MRTA provides additional statistics and insights to debug and improve tracking algorithms.

6.2.1 Tracking as Multi-class Classification

The goal of a multi-resident tracking algorithm is to associate each sensor event with the smart home residents. If the number of the residents in the smart home is fixed or the maximum number of residents specified for a particular dataset, we can treat the output of the multi-resident tracking algorithm as labeling each sensor event with one of multiple classes, each of which represents a resident in the smart home. Thus, common performance measures for multi-class classification problem, such as accuracy score, Hamming loss, precision, recall, and F1-score can be used to compare the performance between tracking strategies.

Before computing the metrics, the target identifiers generated by the tracking algorithm need to be mapped to the resident identifiers annotated in the ground truth. To create such a correspondence, we first group the sensor events associated with each target identifier. We then find the resident identifier which associates with most of those
sensor events according to the ground truth. Thus, a one-to-one mapping between target identifiers of the tracking algorithm and resident identifiers in the ground truth is formed. Based on the mapping, each sensor event is updated with resident identifiers labels, and the multi-class classification metrics can be calculated.

We define association accuracy as the fraction of total sensor events, $D$, in which the ground truth $Y^{(i)}$ equals the set of predicted resident IDs $\hat{Y}^{(i)}$, as shown in Equation 6.1. Resident identifiers include the empty set (no resident) or a set of identifiers for one or more residents.

$$
\text{accuracy} = \frac{1}{D} \sum_{i=1}^{D} 1(Y^{(i)} = \hat{Y}^{(i)})
$$

(6.1)

Hamming loss, on the other hand, gives credit to partial matches between $Y^{(i)}$ and $\hat{Y}^{(i)}$. The definition of Hamming loss is shown in Equation 6.2. In Equation 6.2, $N_R$ represents the total number of residents in the dataset.

$$
\text{Hamming loss} = \frac{1}{D} \cdot \frac{1}{N_R} \sum_{i=1}^{D} \sum_{j=1}^{N_R} 1(Y^{(i)}_j = \hat{Y}^{(i)}_j)
$$

(6.2)

Moreover, if we focus on each resident that is annotated in the ground truth, we can also view sensor event-to-resident association as a binary classification problem. The two classes represent events that are associated with a particular resident (+) and events not associated with that resident (-). In this approach, we can measure the precision, recall, and F1-score for each resident.
6.2.2 Error in Estimated Number of Active Residents

However, as the multi-class classification metrics are computed with a constant number of total classes for each sensor event, they fail to address the scenario in which the number of active residents in the smart home may vary. For a tracking algorithm to work in a real-life environment, estimation of the number of active residents in the house is critical. Every target identifier generated by the tracking algorithms represents a potential resident. Thus, we calculate the number of active target identifiers at each time step and compute the error against the number of active residents annotated in the ground truth. In earlier multi-resident tracking research, a resident is considered to be inactive if the resident has not been detected by any sensors for over 100 seconds, or 50 consecutive sensor events on average [9]. This rule is applied to both the ground truth and the target identifiers generated by NN-SG and GNN-SG method. In the case of sMRT, the likelihood of a resident being active at any time step, can be calculated by integrating the corresponding PHD, as shown in Equation 5.36. If the likelihood is greater than 0.5, we consider the target identifier to be active.

6.2.3 Multi-Resident Tracking Accuracy (MRTA)

Past research on multi-object tracking in computer vision applications has proposed the MOTA metric to extract the accuracy aspect of the system output. The MOTA metric focuses on the potential errors that may occur in the output of a tracking system, including
target miss, false positive target identification, and target identifier mismatch. In computer vision applications, the association between the target identified by the tracking system and the ground truth can be established by the size of overlap area or the physical distance in a video frame. However, in multi-resident tracking applications, the observation and target identifier are discrete, and a one-to-one association may be violated. Thus, we propose a new evaluation metric, MRTA. MRTA results from adapting MOTA to the context of multi-resident tracking.

As with the computation of multi-class classification metrics, we first establish the correspondence between target identifiers generated by tracking algorithms and the ground truth resident identifiers. We then classify the errors between the tracking algorithm output and the ground truth labels into the following three categories: misses, false positives, and mismatches.

**Misses** If a sensor is associated with a resident while in the tracking algorithm, but there is no track identified that is mapped to that resident, the association is counted as a miss.

**False Positives** If a sensor event is associated with a resident and there are multiple tracks generated by the tracking algorithm which all map to the same resident, the association is considered a false positive. Similarly, if a track identified by the tracking algorithm is associated with a resident that is not linked to the sensor event according to ground truth, this association is considered a false positive.

**Mismatch** If a resident is still “active” according to ground truth, while the track identi-
fier changes in the algorithm output, the corresponding associations are considered mismatches.

The MRTA score can be calculated according to Equation 6.3.

\[
\text{MRTA} = 1 - \frac{N_{\text{misses}} + N_{fp} + N_{\text{mismatch}}}{N_{\text{association}}}
\]  

(6.3)

In the equation, \( N_{\text{misses}} \) is the number of target misses, \( N_{fp} \) is the number of false positives and \( N_{\text{mismatch}} \) is the number of target identifier mismatches. The \( N_{\text{association}} \) in the denominator represents the total number of identified event-to-resident associations that were annotated in the ground truth. For example, if a sensor event is associated with two residents, the number of ground truth associations is also two.

The accuracy score and Hamming loss only focus on the correctness of the association hypothesis generated by the tracking algorithms. However, when the tracking algorithm generates multiple target identifiers corresponding to the same ground truth resident at the same time, the accuracy score and Hamming loss metrics do not penalize those errors. On the contrary, MRTA counts those extra target identifiers as false positives.

### 6.3 Evaluation of sMRT

In this experiment, we evaluate the performance of sMRT in comparison against the two baseline methods, NN-SG and GNN-SG. sMRT, introduced in Section 5.1, is built upon the assumption of a constant velocity dynamic model of residents in the measurement space. The measurement space is constructed by mining the spatio-temporal
relationship between sensors using the recorded sensor sequence. In the implementation, the model parameters are determined either heuristically or through a grid search to yield best performance.

In the experiment, we require that each valid track be composed of at least three sensor events. In earlier activity recognition research, the shortest detectable activities contained at least three events (the “enter home” and “leave home” activities). Thus, if a target identifier in the output of a tracking algorithm is associated with fewer than three sensor events, we consider those sensor events to be false alarms and discard the target identifier.

Table 6.1 shows the multi-classification accuracy score and Hamming loss of NN-SG, GNN-SG, sMRT, and sMRT-ML. As shown in Table 6.1, sMRT ties with NN-SG with an accuracy of 0.80, while GNN-SG scores the best with an accuracy of 0.83. In terms of Hamming loss, sMRT scores 0.08, which is 0.01 better than the NN-SG method. GNN-SG performs the best with a Hamming loss of 0.07. Based on the Hamming loss results, GNN-SG achieved the best performance in terms of multi-class classification metrics and identified 93% of sensor event-to-resident associations correctly. When interpreting these results, it is important to note that both NN-SG and GNN-SG require sensor adjacency in the smart home as a prerequisite, which likely boosts the performance of these methods. In contrast, sMRT does not require this information, yet achieved a performance that was 1% lower than GNN-SG and 1% lower than NN-SG without such information.

Performance using per-resident classification metrics for the TM004 and Kyoto datasets are shown in Tables 6.2 and 6.3, respectively. While macro averages are commonly reported
Table 6.1: Multi-label accuracy and Hamming loss of sMRT, NN-sg, and GNN-sg measured using the TM004 dataset. Best performance values are shown in bold. The best performance values that are statistically significant ($p < 0.5$) are marked with an asterisk.

<table>
<thead>
<tr>
<th>Methods</th>
<th>sMRT</th>
<th>NN-sg</th>
<th>GNN-sg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.80</td>
<td>0.80</td>
<td>0.83</td>
</tr>
<tr>
<td>Hamming loss</td>
<td>0.08</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td># Tracks</td>
<td>2834</td>
<td>569</td>
<td>1441</td>
</tr>
<tr>
<td># Sensor Events</td>
<td>51,358</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the classes are imbalanced, we are also interested in results on a per-datapoint bases. Thus, we provide micro and macro averages in Tables 6.2 and 6.3.

For TM004, sMRT’s accuracy is 0.80, similar to the performance of NN-sg, and 0.03 lower than the GNN-sg method. Using the Hamming loss metric, sMRT scores 0.08, which is 0.01 better than the NN-sg method and 0.01 higher than the GNN-sg method. The Hamming loss of sMRT shows that only 8% of the associations are not identified by the sMRT algorithm. Unlike NN-sg and GNN-sg, the sMRT results are achieved without using annotated data or sensor topologies.

When we consider each separate resident in the TM004 dataset, as shown in Table 6.2, sMRT achieves a higher precision but a lower recall compared to the other two methods. Among the four residents labeled in the TM004 dataset, residents R1 and R2 are in the
<table>
<thead>
<tr>
<th>Methods</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th># Events</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sMRT</td>
<td>NN-sg</td>
<td>GNN-sg</td>
<td>sMRT</td>
</tr>
<tr>
<td>R1</td>
<td>0.94</td>
<td>0.89</td>
<td>0.92</td>
<td>0.84</td>
</tr>
<tr>
<td>R2</td>
<td>0.86</td>
<td>0.80</td>
<td>0.86</td>
<td>0.76</td>
</tr>
<tr>
<td>R3</td>
<td>0.85</td>
<td>0.77</td>
<td>0.76</td>
<td>0.71</td>
</tr>
<tr>
<td>R4</td>
<td>0.00</td>
<td>0.00</td>
<td>0.67</td>
<td>0.00</td>
</tr>
<tr>
<td>Average (Micro)</td>
<td>0.91</td>
<td>0.85</td>
<td>0.90</td>
<td>0.81</td>
</tr>
<tr>
<td>Average (Macro)</td>
<td>0.66</td>
<td>0.61</td>
<td>0.80</td>
<td>0.58</td>
</tr>
</tbody>
</table>

Table 6.2: Performance of sMRT, NN-sg and GNN-sg using the TM004 dataset, measured based on binary classification accuracy on a per-resident basis.
<table>
<thead>
<tr>
<th>Metrics</th>
<th>Methods</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th># Events</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>sMRT</td>
<td>NN-sg</td>
<td>GNN-sg</td>
<td>sMRT</td>
</tr>
<tr>
<td>R1</td>
<td></td>
<td>0.74</td>
<td>0.92</td>
<td>0.91</td>
<td>0.72</td>
</tr>
<tr>
<td>R2</td>
<td></td>
<td>0.65</td>
<td>0.83</td>
<td>0.90</td>
<td>0.79</td>
</tr>
<tr>
<td>R3</td>
<td></td>
<td>0.68</td>
<td>0.76</td>
<td>0.82</td>
<td>0.59</td>
</tr>
<tr>
<td></td>
<td>Average (Micro)</td>
<td>0.69</td>
<td>0.85</td>
<td>0.90</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>Average (Macro)</td>
<td>0.69</td>
<td>0.84</td>
<td>0.88</td>
<td>0.70</td>
</tr>
</tbody>
</table>

Table 6.3: Performance of sMRT, NN-sg and GNN-sg using the Kyoto dataset, measured based on binary classification accuracy on a per-resident basis.
smart home most of the time, associated with 32,272 and 17,873 sensor events, respectively. Residents R3 and R4 represent visitors who likely trigger only 1,202 and 11 sensor events, respectively. However, the 11 sensor events associated with R4 are separated by sensor events that were triggered by other residents. As a result, those 11 sensor events are regarded as isolated sensor events by both NN-sg and sMRT, and no resident identifier is produced.

However, in the Kyoto dataset where the sensors are more densely deployed with greater noise and less reliability, sMRT has a difficult time reliably tracking the residents compared to NN-sg and GNN-sg. By analyzing the sensor vectors learned by sMRT and the tracking results, we find that the main cause of the sMRT performance decrease is that some sensors that are not physically adjacent to each other according to the sensor layout have a relatively short distance in the measurement space. The sensors exhibiting such an error either generate events only a few times in the dataset (recorded within 3 days) or have a higher probability of noise. For example, on one of the days, the bathroom door on the second floor was not closed properly, resulting in sensor D005 continually sending "OPEN" and "CLOSE" messages while another resident was downstairs in the living room triggering motion sensor M004. As a result, sMRT identifies sensors D005 and M004 as being close to each other in the measurement space while they are actually far from each other in the home.

Figures 6.3 and 6.4 show the accuracy and Hamming loss values of sMRT, NN-sg and GNN-sg when there are different numbers of residents in the smart home using the TM004 dataset. As the number of active residents in the smart home increases, the performances
Figure 6.3: Accuracy score as a function of the number of active residents for sMRT, NN-sg and GNN-sg using the TM004 dataset.

of sMRT and NN-sg decrease. However, baseline GNN-sg achieves a better accuracy when there are 4 residents in the smart home. One explanation for this anomaly is the small sample size for 4 residents. There are only 11 time steps when 4 residents are in the home, as shown in Table 6.2. In contrast, there are > 1,000 sensor events for the other cases. According to the Hamming loss shown in Figure 6.4, we find that sMRT is more accurate in grouping the sensor events triggered by the same resident than NN-sg, though
the performance is 0.01 lower than GNN-sg when there are two or three residents.

We also evaluate these methods based on their ability to estimate the number of active residents currently present in the smart home. In earlier multi-resident tracking research, a resident is considered to be inactive if the resident has not been detected by any sensors for over 100 seconds, or 50 consecutive sensor events on average [9]. Since sMRT and both baseline methods operate based on discrete time steps, we count an existing
Table 6.4: Average error of sMRT, NN-sg and GNN-sg in estimation of the number of active residents in the smart homes. Best performance values are shown in bold. The best performance values that are statistically significant $(p < 0.5)$ are marked with an asterisk.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TM004</th>
<th>Kyoto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>sMRT</td>
<td>NN-sg</td>
</tr>
<tr>
<td>Average Error</td>
<td>0.59</td>
<td><strong>0.41</strong></td>
</tr>
</tbody>
</table>

Ideally, we want the sensor events associated with the same residents to match the
Table 6.5: MRTA performance of NN-SG, GNN-SG and sMRT. The best performance values are shown in bold. The best performance values that are statistically significant \((p < 0.5)\) are marked with an asterisk.

resident identifier predicted by the tracking algorithm. However, in the experiments, we find the segmentation errors, where the sensor events that are associated with the same resident are split into multiple tracks with different resident identifiers, may affect the real-life performance when the tracking algorithm is used for activity-aware applications. By counting the number of valid resident identifiers generated by each method, as shown in Table 6.1, both sMRT and GNN-sg result in a higher number of valid resident identifiers compared to the NN-sg method. The result indicates that both sMRT and GNN-sg tend to generate more resident identifiers and associate the sensor events that are triggered by same resident associated with those identifiers.

The MRTA performances of NN-SG, GNN-SG and sMRT are shown in Table 6.5.
NN-SG achieves the best MRTA score of 0.68, with sMRT trailing at 0.47 and GNN-SG at 0.40. When we break down the tracking errors into misses, false positives, and mismatches, we find that NN-SG has the lowest numbers of false positives and mismatches. GNN-SG has the lowest number of target misses, but exhibits extremely high counts of false positives and track ID mismatches. However, the result of sMRT shows high numbers of misses and false positives compared with NN-SG and GNN-SG. At the same time, the algorithm achieves a MRTA of 0.47, higher than GNN-SG.

6.4 Evaluation of sMRT-ML

sMRT-ML, introduced in Section 5.2, learns the parameters of the dynamic model and measurement model by maximizing the likelihood of the observed sensor measurements at each time step. In this section, we present the performance of sMRT-ML evaluated using the TM004 dataset.

Table 6.6 shows the multi-classification accuracy score and Hamming loss of sMRT-ML. In comparison with sMRT, NN-sg and GNN-sg, as shown in Table 6.1, the sMRT-ML exceeds the performance of both sMRT and NN-sg by 0.01 and achieved the same best score as GNN-sg in term of Hamming loss.

Performance of sMRT-ML using the per-resident classification metrics evaluated for the TM004 dataset is shown in Table 6.7. Compared against the results of sMRT, NN-sg, and GNN-sg (Table 6.2), the binary classification precision on R1 is 0.04 lower than sMRT, but 0.04 and 0.02 higher for R2 and R3, respectively. On the per-class recall, the
<table>
<thead>
<tr>
<th>Methods</th>
<th>sMRT-ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.81</td>
</tr>
<tr>
<td>Hamming loss</td>
<td>0.07</td>
</tr>
<tr>
<td># Tracks</td>
<td>1581</td>
</tr>
<tr>
<td># Sensor Events</td>
<td>51,358</td>
</tr>
</tbody>
</table>

Table 6.6: Multi-label accuracy and Hamming loss of sMRT-ML measured using the TM004 dataset.

sMRT-ML performs 0.06 and 0.02 better than sMRT for R1 and R2, and slightly worse for R3. Overall, the unsupervised training provided by sMRT-ML improves the micro-average of recall by 0.03 and F1-score by 0.04.

We further compare sMRT-ML performance using MRTA metrics, as shown in Table 6.8. Comparing with the sMRT MRTA score shown in Table 6.5, the sMRT-ML MRTA score is 0.15 higher. This performance improvement is due to the massive reduction in false positives and target misses. On the other hand, sMRT-ML mismatches outnumber those of sMRT by 900, out of a total of more than 51 thousand associations. Compared to the MRTA performance of NN-SG and GNN-SG, sMRT-ML is 0.08 lower than the best performance achieved by NN-SG. sMRT-ML results in a similar number of target misses, but experiences a slightly higher number of false positives and target mismatches.

As the target mismatches, target misses and false positives in sMRT strongly related
<table>
<thead>
<tr>
<th>Metrics</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th># Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
<td>32272</td>
</tr>
<tr>
<td>R2</td>
<td>0.90</td>
<td>0.74</td>
<td>0.81</td>
<td>17873</td>
</tr>
<tr>
<td>R3</td>
<td>0.87</td>
<td>0.66</td>
<td>0.75</td>
<td>1202</td>
</tr>
<tr>
<td>R4</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>11</td>
</tr>
<tr>
<td>Average (Micro)</td>
<td>0.90</td>
<td>0.84</td>
<td>0.87</td>
<td>51358</td>
</tr>
<tr>
<td>Average (Macro)</td>
<td>0.67</td>
<td>0.57</td>
<td>0.62</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.7: Performance of sMRT-ML using the TM004 dataset, measured based on binary classification accuracy. Accuracy is calculated on a per-resident basis.

<table>
<thead>
<tr>
<th>sMRT-ML</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MRTA</td>
<td>0.61</td>
</tr>
<tr>
<td>Misses</td>
<td>7,529</td>
</tr>
<tr>
<td>False positives</td>
<td>7,017</td>
</tr>
<tr>
<td>Mismatches</td>
<td>5,259</td>
</tr>
<tr>
<td>Total Associations</td>
<td>51,358</td>
</tr>
</tbody>
</table>

Table 6.8: MRTA performance of sMRT-ML evaluated using the TM004 dataset.
Figure 6.5: MRTA performance versus the minimum length of sensor events. By varying the number of sensor events a valid target identifier should be associated with, we plot the MRTA score (top left), ratio of target misses $N_{\text{misses}}/N_{\text{associations}}$ (top right), ratio of false positives $N_{\text{fp}}/N_{\text{associations}}$ (bottom left), and ratio of target identifier mismatches $N_{\text{mismatches}}/N_{\text{associations}}$ (bottom right).

to the minimum number of sensor events required for an identified track to be valid. Figure 6.5 shows the change of different MRTA score and percentage of each errors with respect to the minimum valid track length. If the minimum number of sensor events associated
with a valid target increases, an increase in target misses is observed among all three algorithms, with sMRT increasing most rapidly. However, if we require each valid target is associated with more sensor events, the false positives and target mismatches of sMRT drops rapidly, and the MRTA score of sMRT may reach 0.56 and sMRT-ML may reach 0.64. The MRTA scores of NN-SG and GNN-SG are more resilient to such changes. However, comparing sMRT-ML with sMRT, sMRT-ML is generally better than sMRT with higher MRTA scores, lower misses, false positives and target mismatches across the board.

Finally, we also plot the MRTA metrics with respect to the number of residents in the smart environment, as shown in Figure 6.6. In this experiment, NN-SG achieves the best performance among all the methods evaluated in the experiment. sMRT-ML tails NN-SG by about 0.05 in terms of MRTA score and about 0.03 on average in false positives when there are 1-3 residents in the environment. In terms of target misses, sMRT-ML slightly outperforms NN-SG when there are 2-3 residents in the smart home. Target mismatch errors again are confirmed to be the main cause of the MRTA performance difference between sMRT-ML and NN-SG. However, sMRT-ML reliably performs better than sMRT in MRTA scores, target misses and false positive metrics under all scenarios.

6.5 Track Consistency

Target mismatch errors occur when the associated resident identifier changes in the algorithm output while the corresponding resident is still considered “active” according to the ground truth. The target mismatch errors can originate from the generation of a new
Figure 6.6: MRTA performance versus the number of residents. We plot the MRTA score (top left), ratio of target misses $N_{\text{misses}}/N_{\text{associations}}$ (top right), ratio of false positives $N_{\text{fp}}/N_{\text{associations}}$ (bottom left), and ratio of target identifier mismatches $N_{\text{mismatches}}/N_{\text{associations}}$ (bottom right) against the number of residents present in the smart environments based on experiments conducted using TM004 dataset.

A resident identifier rather than contributing the sensor measurement to an existing “active” resident. Additionally, such an error can occur with the spawn of a new resident identifier when the algorithm estimates that there are actually multiple residents associated with a
single track.

As proposed in Section 5.3, we designed rule-based methods to improve track consistency. When a new track is spawned and associated with a sensor measurement, we identify the previously “active” resident last seen at this location. If such a resident can be found within a time window, we consider that both resident identifiers generated by the algorithm correspond to the same resident. The size of the time window is denoted as a “gap” between the previously and newly-spawned tracks.

However, in addition to the upper bound of the number of time steps between the end of the previous track and the spawn of the new track (represented by “gap”), there can be a lower bound as well. If the lower bound is greater than 0, it means that the previous resident has turned to “inactive” before the spawn of a new resident identifier. However, in some scenarios, it may take a while for the weight of the previous resident to decay. Hence, we also consider the “overlap” between two tracks. In order for two tracks to join, the number of time steps that elapses between the end of the previous track and the spawning of the new one should be within the range defined by parameters “overlap” and “gap”.

In the experiment, based on the output of sMRT-ML, we vary the values of “overlap” and “gap”. The accuracy score, Hamming loss, and MRTA metrics are shown in Figure 6.7.

In terms of multi-class classification metrics, Hamming loss does not penalize the target mismatch errors. As a result, the larger the value of “overlap” and “gap”, the more tracks will join. Hence, the Hamming loss will only increase. However, judging from the
Figure 6.7: The effect of track consistency optimization strategy on tracking performance measured in accuracy, Hamming loss and MRTA metrics.
accuracy score, the highest accuracy is achieved with “overlap” set to 1 and “gap” set to 50.

Because MRTA metrics penalize the algorithm with target mismatch errors, the trade-offs between target misses, target mismatches, and false positives can be directly spotted in Figure 6.7. Consider cases with larger values of “gap” and “overlap”. Here, as more tracks join together, the target mismatch errors can decrease from 9.5% down to 6.9%. However, the cost is an increase in target misses (from 14.9% to 18.2%) and false positives (13.7% to 18.0%). Judging from MRTA metrics, the sweet spot occurs if “overlap” is set to 1 and “gap” is set to 5.

6.6 Discussion and Limitations

sMRT, proposed in this dissertation, provides a mathematical rigorous model for multi-resident tracking in smart environments equipped with ambient sensors. sMRT-ML shares the problem formalization and the tracking phase of sMRT. However, this enhancement of sMRT offers improvements through an unsupervised training process. The tracking consistency optimization trades off target mismatches with false positives and target miss errors. However, due to the approximation adopted during the tracking phase and the linear assumption of dynamic and measurement models, sMRT and sMRT-ML face limitations during tracking. Because the limitations apply to both sMRT and sMRT-ML, we let “sMRT” refer to both algorithms throughout this discussion.

During the propagation of PHD in the tracking phase, a one-to-one relationship
between sensor observations and residents is still assumed. sMRT relaxes this assumption. Specifically, they handle the many-to-one relationship between sensor observations and residents by properly setting the parameters for the clutter process, namely $\lambda_c$ and $c(z)$. These two parameters serve as counterweights to prevent the rapid increase of cardinality estimation when a resident triggers two sensor observations at the same time. However, when a resident is at a location where the sensors are more densely populated, chances are that a many-to-one association between sensor observations and residents may be higher than what the clutter process parameters can handle. As a result, sMRT will spawn a new resident track. Not long afterward, the original track may terminate itself. This behavior may thus lead to a higher count of valid resident identifiers while still maintaining an accurate estimation of the number of active residents in the smart home. However, adjacent sensor events that are associated to the same resident are likely to still be adjacent, even when a new resident identifier is assigned.

In contrast, the segmentation errors observed for GNN-sg are caused by keeping multiple resident identifiers valid at the same time. In similar cases where a resident is spotted by multiple sensor observations, GNN-sg simultaneously creates multiple resident identifiers associated with each of these sensor observations. Based on the GNN policy, the sensor events are separated into different tracks in an interleaved fashion. It is thus more likely, compared to sMRT, that adjacent sensor events associated with the same resident are assigned to different resident identifiers. This behavior causes an increase in the predicted number of valid resident identifiers and an inaccurate estimation of the number of active residents in the smart home. Moreover, even though GNN-sg results in improved accuracy
and Hamming loss score (i.e., GNN-sg is more accurate in grouping together the sensor events triggered by same resident), the result is less effective in activity-aware applications where the continuity of sensor events is an important factor.

In a multi-resident smart home, the assumption of one-to-one associations between sensor events and residents may not hold, especially when multiple residents are performing joint activities and are moving together as a group (e.g., cooking together). Even though all of the tracking algorithms assume a one-to-one association, differences in enforcing this assumption result in different levels of tracking capabilities for collaborative-activity scenarios. Both baseline methods, NN-sg and GNN-sg, treat the one-to-one association as a rule. Therefore, if multiple residents trigger the activation of the same sensor, the algorithms can only associate the sensor observation to one of the residents. In cases when multiple residents are performing activities together in a local area, GNN-sg will generate additional parallel tracks, leading to an over-estimation of the number of active residents in the smart home as shown in Table 6.4.

In contrast, even though sMRT does not explicitly model joint activities, it predicts the sensor they will next jointly trigger through sensor vectorization in combination with a constant velocity model. As a result, sMRT can track collaborative resident activities in a local space. However, in the case of multiple residents moving together, e.g. going downstairs together to the kitchen in the morning, the performance of sMRT depends on the length of the sequence when residents move together. Since sMRT’s corrector is derived based on the assumption that there is a one-to-one association between sensor observation and resident, the integral of the PHD corresponding to each resident (i.e.,
the sum of weights of Gaussian components associated with each resident) will decrease. However, as there is still a sensor observation during the procedure and both residents are likely to be associated with the sensor observation, the sum of weights of both residents will decrease but their sum will still reach 1. Thus, when the multiple residents go their separate ways, the PHD integral of each resident may still be greater than the birth PHD and the tracks identified by the sMRT algorithm can be maintained.

sMRT uses a constant-velocity model to map resident movement in the smart environment into the measurement space. The unsupervised learning process of sMRT-ML relaxes the constant velocity assumption by learning parameters of a linear dynamic model. However, mobility in the smart environment may not be linear, even when mapped to a latent space. In some cases, movement can even be time variant. Removal of the linearity of the dynamic model means that multi-resident tracking cannot be solved in closed forms, such as the Gaussian Mixture used by both sMRT and sMRT-ML. As a result, further advances in numerical computation and training methods for a PHD filter will need to be developed and applied to this problem. Though it is beyond the scope of the work presented in this dissertation, this is a promising direction for future endeavors into building an easy-to-deploy and user-friendly unsupervised multi-resident tracking solutions.

6.7 Computational Complexity

Finally, we compare the computational complexity of sMRT (and sMRT-ML) and the baseline algorithms. NN-sg associates each existing resident with the nearest, i.e.
most-likely, sensor observation. In these analyses, we assume a constant-cost computation of distance for the nearest neighbor algorithms. NN-sg performs a nearest-neighbor search for the \( N \) currently-active residents. Given that each nearest-neighbor call incurs a cost of \( O(n) \), the total runtime for NN-sg is thus \( O(nN) \). GNN-sg attempts to find the best one-to-one assignment between sensor observations and existing residents. Using an efficient method such as the Hungarian algorithm, this could be accomplished in time \( O(n^3) \).

The tracking phase for sMRT is composed of a GM-PHD filter and a clustering-based track maintenance algorithm. Assuming a maximum of \( J \) Gaussian components in the mixture and a \( m \)-dimensional measurement space, GM-PHD updating is bounded by \( O(nJm^3) \). The worst-case complexity of sMRT’s track maintenance is \( O(NJmi) \), where \( i \) is the number of iterations that lead to clustering convergence.
Ambient binary sensors, such as PIR motion sensors and magnetic door sensors, offer a low-cost, unobtrusive, and easy-to-deploy solution to ambient assisted living and smart environment applications. However, the challenge of handling multi-resident scenarios hinders widespread adoption of the technology. In this dissertation, we provide a data-driven unsupervised multi-resident tracking solution to detect and track residents based on ambient sensor data collected in a smart environment.

First, we lay the theoretical foundation of the predictability of resident indoor mobility in both single and multi-resident settings. We hypothesize that the sensor events recorded in a smart environment can be modeled with an stationary stochastic process. Thus, the predictability of resident indoor mobility is the average of the maximum certainty the stochastic process exhibits when making a prediction about the next sensor event the residents in the smart home can trigger. According to Fano’s inequality, the upper bound of the predictability can be derived from the entropy rate of the stochastic process. Provided with the ergodic property of the stationary process, the entropy rate can be estimated from the recorded sensor sequence. With sensor events recorded in over 100 smart homes, each for a couple of weeks to multiple years, we found that the predictability of human mobility in single resident settings lies around 83%. In multi-resident settings, the predictability is about 7% lower. The analysis on predictability confirms and quantifies the amount of regularities observed in our daily movement routines.
Following the predictability analysis, we proposed sMRT, a data-driven unsupervised multi-resident tracking algorithm. sMRT consumes the recorded sensor events as the sole input, without the need of additional information such as the smart environment floor plan, sensor locations, or pre-annotated sensor events. sMRT first translates the smart environment sensors into vectors of a latent measurement space by mining the spatio-temporal relationship from the sensor sequence. We hypothesized that resident movement in the smart environment can be equivalent to a point target maneuvering in the measurement space following a constant velocity model. During the tracking phase, sMRT utilizes a GM-PHD filter to estimate the location of each resident in the measurement space as well as the number of residents in the smart environment.

In addition, we offer a refinement to sMRT, called sMRT-ML, by introducing an unsupervised learning process to train the parameters of the sMRT models. Based on the history of sensor observations, sMRT-ML estimates the current resident states and derives a probability distribution of the future sensor observations. Thus, by maximizing the likelihood of the recorded sensor observations, the model parameters can be trained using optimization algorithms, such as gradient descent. sMRT-ML also relaxes the constant velocity assumption by applying the training approach to the dynamic model linear multiplier.

To evaluate the performance of sMRT and sMRT-ML, we used both algorithms to process sensor events recorded in real-life datasets. Ground truth labels were generated by lab annotators with a visualization tool, ActViz, specifically designed for multi-resident environments. We compared the performance of sMRT and sMRT-ML against two base-
line methods, NN-sg and GNN-sg, both based on sensor graphs hand-crafted from a smart environment floor plan and sensor location information. Performance was measured both using the traditional multi-class classification metrics (precision, recall, and F-1 score). Additionally, we defined novel multi-resident tracking accuracy (MRTA) metrics and evaluated the algorithms based on this new measure. Considering that sMRT and sMRT-ML solves the multi-resident tracking without additional information, which NN-SG and GNN-SG do require, the results provide evidence that sMRT and sMRT-ML, as an initiative for unsupervised multi-resident tracking, are capable of associating sensor events with residents in real-life settings.

This research explores the unsupervised data-driven multi-resident tracking approaches and investigates the training methods for multi-resident models. Continued research in finding an unsupervised multi-resident tracking solution could help assisted ambient living and smart environment technology to scale to multi-resident scenarios, thus providing practical benefit to individuals and families needing activity monitoring and activity-aware services. Here, we summarize a few directions for future research.

First of all, the predictability studied in this dissertation confirms the existence of regularity in human mobility in smart environment. However, in future work the measured predictability can also potentially be used as a theoretical metric to measure the fitness of a trained dynamic model and guide the training process of the dynamic model.

Second, sMRT and sMRT-ML require that the dynamic model of resident movement in smart environment be represented in a linear Gaussian form to ensure the closed-form solution of the GM-PHD filter. However, mobility in the smart environment may not be
linear, even when mapped to a latent space. As a result, numerical algorithms combined with non-linear dynamic models need to be introduced to relax the linearity assumption. One of the possible ways to achieve this goal is to replace the GM-PHD filter with a cardinality-balanced multi-target multi-Bernoulli filter with sequential Monte-Carlo implementation [120]. However, the unsupervised training stage must be adapted to the new formalization.

Moreover, the dynamic model adopted in sMRT and sMRT-ML relies on the resident states without taking into account the time that elapsed between two consecutive sensor observations. In many cases, the elapsed time between two sensor events offers valuable information about the resident state. Incorporating elapsed time as an additional parameter to the dynamic model may fit better with sensor observations recorded in real-life applications.

Finally, the unsupervised training process of sMRT-ML is the first work that combines discriminative training with multi-target filters. Creation of additional mapping between input data and measurement space, the learning procedure can be introduced into other multi-target tracking applications, such as multi-object tracking in videos.
REFERENCES


