Simulation-based Sensor Configuration Optimization to Detect Human Activities in Smart Indoor Spaces

by

Shadan Golestan-Irani

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

Doctor of Philosophy

Department of Computing Science
University of Alberta

Abstract

Smart indoor spaces include a network of interconnected sensors, predictive models, and actuators to sense occupants' activities and act to improve living. These ubiquitous systems are increasingly popular due to their potential to improve energy efficiency, comfort, and safety in buildings. Typically, the sensors collect data on the environment, while the predictive models use this data to predict the environment's activities and state. The actuators then act on the environment to change the environment's state to a desired one.

The design of smart indoor spaces is a challenging task that requires a lot of effort and time. One critical aspect of the design process is finding a suitable sensor configuration, as different applications require different sensor configuration deployments. The quality of the sensor configuration is critical for the system's overall effectiveness, as inaccurate or incomplete sensor data can lead to poor performance.

To address these challenges, designers of such systems iteratively deploy and modify different sensor configurations to find the one that meets their needs [147]. This process is complex and time-consuming. Alternatively, simulation methodologies have been proposed to facilitate the design process because they offer fast, reproducible, and easy prototyping. Simulation allows designers to test and evaluate different sensor configurations in a virtual environment, reducing the need for expensive and time-consuming physical deployments. However, current simulation approaches still require significant effort and expertise to optimize the sensor configuration accurately.

This thesis proposes a simulation-driven sensor configuration evaluation framework that can optimize the sensor configuration effectively for various applications.

Such a framework evaluates and optimizes different sensor configurations efficiently, leading to the development of more effective smart indoor spaces. The framework is evaluated in terms of 1) the fidelity of its simulation methodologies, 2) the quality of the sensor configurations it proposes, and 3) the framework's scalability considering the sensor numbers and types.

Preface

The central chapters of this thesis are based on papers that are either published or that are currently under review. In particular, Chapter 2 is written based on papers that are published in conference papers [1, 2]. Chapter 3 is written based on a journal paper [3]. Most of Chapter 4 is based on two papers that are published in a conference paper [4] and a journal [5]. Finally, Chapter 5 is written based on a conference paper [6].

[1] Golestan, S.; Petcovici, A.; Nikolaidis, I.; Stroulia, E. Simulation-Based deployment configuration of smart indoor spaces. 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). IEEE, 2019, pp. 358–363, DOI: 10.1109/WF-IoT.2019.8767206.

[2] Golestan, S.; Kazemian, S.; Ardakanian, O. Data-driven models for building occupancy estimation. Proceedings of the Ninth International Conference on Future Energy Systems, 2018, pp. 277–281, DOI: 10.1145/3208903.3208940.

[3] Golestan, S.; Stroulia, E.; Nikolaidis, I. Smart Indoor Space Simulation Methodologies: A Review. IEEE Sensors Journal 2022, DOI: 10.1109/JSEN.2022.3159205.

[4] Zhao, Y.; Pour, F.F.; Golestan, S.; Stroulia, E. BIM Sim/3D: multi-agent human activity simulation in indoor spaces. 2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS). IEEE, 2019, pp. 18–24, DOI: 10.1109/SEsCPS.2019.00011.

[5] Golestan, S.; Nikolaidis, I.; Stroulia, E. Towards a Simulation Framework for Smart Indoor Spaces. Sensors 2020, 20, 7137, DOI: 10.3390/s20247137.

[6] Golestan, S.; Ardakanian, O.; Boulanger, P. Sensor Configuration for Accurate Activity Recognition in Indoor Environments using Bayesian Optimization and Building Simulation. IJCAI 2023, Under Review, DOI: 10.3390/s20247137.

To my parents,

my partner,

and

my brother

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Pierre Boulanger, for his unwavering guidance, support, and encouragement throughout the entirety of this research project. His expertise, feedback, and constructive criticism were invaluable in shaping the direction of this thesis.

I also extend my heartfelt thanks to my committee members, Dr. Omid Ardakanian, Dr. Matthew Guzdial, Dr. Marek Reformat, and Dr. Kevin Bouchard. Their insights, feedback, and suggestions have been instrumental in improving the quality of this research.

I am also grateful to Dr. Matthew E. Taylor for his invaluable insights, feedback, and support, which have greatly contributed to the success of this thesis.

I would like to express my sincere gratitude to Athar MahmoudiNejad, my amazing and wonderful partner. Athar has undoubtedly had a profound influence on me as a researcher. Her meticulous attention to detail and exceptional skills have been a great source of inspiration, while her boundless energy and enthusiasm have kept me motivated and enabled me to push my limits.

I was fortunate to be surrounded by friends who made this journey joyful: Payam Jome Yazdian, Mahsa Keramati, Sepehr Mohammad Lavasani, Anita Khalafbeigi, Ehsan Kamalloo, Saeed Sarabchi, Shiva Zarezadeh, Shohreh Zarezadeh, Mahtab Farrokh, Faezeh Haghverd and Amirmohsen Sattarifard. I have fond memories of Pouneh Gorji and Arash Pourzarabi who tragically were among the victims of Flight PS752.

Table of Contents

1 Introduction **1**

1.1 Thesis Statement 4

1.1.1 Smart Indoor Space Simulation Methodology 4

1.1.2 Sensor Configuration Evaluation Framework 5

1.2 Key Contributions 6

1.3 Thesis Outlines 8

2 Using Low-level Sensor Data for High-level Sensor Configuration Evaluation **10**

2.1 Data-driven Models for SIS Occupancy Estimation 11

2.1.1 Introduction and Background 11

2.1.2 Methodology 13

2.1.3 Results 16

2.2 Using Contextual Information Inference for Evaluating Different Sensor Configurations 20

2.2.1 Introduction and Background 20

2.2.2 Methodology 22

2.2.3 Results 26

2.2.4 Discussion 29

2.3 Conclusions 32

3 Taxonomy Specification of Indoor Space Simulation Methodologies **33**

3.1 Survey Methodology 34
3.1.1 Snowballing Procedure 34
3.1.2 Bibliometric Overview 35
3.1.3 Key Features of SIS Simulation Methodologies 37
3.1.4 Publication Clustering 43
3.2 In-Depth Review of the Field 43
3.2.1 Uncertainty Simulation 45
3.2.2 Inference Debugging Tools 49
3.2.3 Co-simulation 51
3.2.4 Emulation 53
3.2.5 Data-Driven Agent Modelling 57
3.2.6 Dataset Generation 62
3.2.7 Software Tools 77
3.2.8 User Experience Simulation 79
3.2.9 Services Debugging 81
3.3 Discussion 85
3.4 Conclusion 88

4 SIM_{sis} : A Simulator for Smart Indoor Spaces 89

4.1 Introduction 89
4.2 Related Work 91
4.3 Simulation Methodology, Models, and Validity Metrics 98
4.3.1 Human-Activity Simulation 100
4.3.2 Sensor-Event Simulation 103
4.3.3 Validity of the Agent Traces 106
4.3.4 Validity of Sensor Events 106
4.4 Experimental Evaluation 111

4.4.1 Example . . . 114
4.4.2 Agent Traces Validation Results . . . 116
4.4.3 Sensor Events Validation Results . . . 117
4.4.4 Discussion . . . 122
4.5 Conclusions . . . 127
4.6 Appendix . . . 128

5 Simulation-based Sensor Configuration Evaluation 131
5.1 Introduction . . . 132
5.2 Related Work . . . 135
5.2.1 Manual Approach . . . 135
5.2.2 Area Coverage Approach . . . 135
5.2.3 Targeted Coverage Approach . . . 136
5.3 Methodology . . . 138
5.3.1 Simulator of Smart Indoor Spaces . . . 139
5.3.2 Activity Classifier . . . 140
5.3.3 Optimization Algorithm . . . 141
5.4 Algorithm Evaluation . . . 145
5.4.1 Case Study . . . 145
5.4.2 Results . . . 146
5.5 Scalability and Modularity Tests . . . 151
5.5.1 Scalability Test . . . 153
5.5.2 Modularity Test . . . 154
5.6 Discussion . . . 155
5.7 Conclusion . . . 156

6 Conclusion and Future Work 158
6.1 Summary of Contributions . . . 158
6.2 Future Work . . . 161

6.2.1 Simulation Methodology 161
6.2.2 Sensor Configuration Evaluation 162

Bibliography **164**

List of Tables

2.1 Our testbed for the evaluation of our contextual information inference prototype 14

2.2 The RMSE of PF and NARX (Ri: the ith room in Building 2) 17

2.3 The RMSE of PF for different numbers of particles. 20

2.4 Our testbed for sensor configuration evaluation proof of concept. . . . 27

3.1 Selected papers for the snowballing procedure's initial set. 35

3.2 Clusters information obtained in our literature review. 44

3.3 Taxonomy specifications of the papers in the uncertainty simulation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 46

3.4 Taxonomy specifications of the papers in the inference debugging tools cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 50

3.5 Taxonomy specifications of the papers in the co-simulation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 51

3.6 Taxonomy specifications of the papers in the emulation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 55

3.7 Taxonomy specifications of the papers in the data-driven agent modelling cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 58
3.8 Taxonomy specifications of the papers in the dataset generation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 63
3.8 Taxonomy specifications of the papers in the dataset generation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type (continued). 64
3.8 Taxonomy specifications of the papers in the dataset generation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type (continued). 65
3.9 Taxonomy specifications of the papers in the software tools cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 77
3.10 Taxonomy specifications of the papers in the user experience simulation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 79
3.11 Taxonomy specifications of the papers in the services debugging cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type. 82

4.1 Scripted sequence of activities of daily living performed by participants in [94] followed by their estimated time of completion. 112
4.2 SIM_{sis} evaluation testbed. 112
4.3 Similarity measure obtained from DTW for real-world agent traces and synthetic agent traces. The average cost of the optimal warping path. 117
4.4 SAS analysis for motion sensors with $W = 7$ min. 125

4.5 SAS analysis for beacon sensors with $W = 7$ min. 125
4.6 TSR analysis for $W = 7$ min. 126

5.1 Occupants' ADLs in our case study. ADLs and a few detailed activities with the same superscript (a or b) can be shuffled. 148
5.2 The performance of GA, Greedy, and BO in terms of the macro-averaged F1-score (Equation 5.2) in T1. A dash is used when a sensor configuration is not found using 1000 queries. 149
5.3 The performance of GA, Greedy, and BO in terms of the macro-averaged F1-score (Equation 5.2) in T2. There is no significant difference between results that have the same superscript. A dash is used when a sensor configuration is not found using 1000 queries. 150
5.4 The performance of our component with BO in terms of the macro-averaged F1-Score (Equation 5.2) in T1 using 7 motion sensors and varying LS sensors (type and number). 153

List of Figures

2.1 Time Series Neural Network block diagram. 16

2.2 Estimated occupancy count using PF in Building 1 (RMSE = 0.4) . . 17

2.3 Estimated occupancy count using NARX in Building 1 (RMSE = 0.3) 18

2.4 Estimated occupancy count using PF in Building 2, Room 4 (RMSE = 2.9) 19

2.5 Estimated occupancy count using NARX in Building 2, Room 4 (RMSE = 0.8) 19

2.6 Overall architecture of a prototype for sensor configuration evaluation. 23

2.7 The CASi floor plan editor shows rooms (large polygons), doors (red dots), obstacles/objects (grey polygons) and their interactive sides (red lines). 24

2.8 Floor plan and motion sensor locations (dots) for the two real-world spaces. The radius of motions sensors is 60 cm. 27

2.9 Estimated and synthetic ground-truth location of one occupant in each of our testbed environments. For better readability, the X and Y coordinates are shown separately (upper is X, lower is Y). 28

2.10 OHE calculation for two test cases; yellow and black colors show minimum and maximum errors respectively; white color shows areas for which we lack information. We increase the information granularity for Waldo library to $10m * 10m$ blocks level. 30

2.11 CAEH calculation for two test cases, with color range similar to OHE calculation (Figure2.10). Here we also increase the information granularity for Waldo library to $10m * 10m$ blocks level. 30
3.1 Histogram of studies included in our survey throughout the years. . . 36
3.2 Histogram of the studies in terms of different citation counts. 36
3.3 Taxonomy of SIS simulation methodologies. 37
3.4 Dendrogram of the studies. Each color shows a separate cluster. . . . 42
4.1 The software architecture of the SIM_{SIS} toolkit. Real-world measurements module is shown in red color. 98
4.2 Screenshots of the BIM Editor used in SIM_{sis} toolkit. 101
4.3 Our testbed for SIM_{sis} evaluation and motion sensors and Bluetooth Low Energy (BLE) beacons configurations. 113
4.4 A simple environment as an example for our methodology. The green color shows which motion sensor gets activated in each step. 115
4.5 Algorithms 2 and 3 execution examples. 116
4.6 Dynamic time warping (DTW) results of synthetic agent traces compared to real-world agent traces (ground truth). 117
4.7 Ground-truth sensor events (blue dots) versus synthetic (red dots) for motion sensors only. Red dots are plotted slightly higher that the line of the particular sensor to avoid occlusion. 118
4.8 Ground-truth sensor events (blue dots) versus synthetic events (red dots) for for beacons only. Red dots are plotted slightly higher that the line of the particular sensor to avoid occlusion. 119
4.9 SAS analysis for motion sensors for different window sizes. 121
4.10 SAS analysis for beacon sensors for different window sizes. 121
4.11 Normalized frequency of sensors activation duration. 124

4.12 TSR analysis: (left column) RMSE value, (middle column) precision score, and (right column) recall score of motion sensors (shown in different colors) for different window sizes in each session. 129
4.13 TSR analysis: (left column) RMSE value, (middle column) precision score, and (right column) recall score of beacon sensors (shown in different colors) for different window sizes in each session. 130

5.1 Comparison of (a) search-based methods, *e.g.* Genetic Algorithm, versus (b) Estimation of Distribution Algorithms (EDAs), *e.g.* Bayesian Optimization, in finding the maximum value of a function $f(x)$. . . . 134
5.2 Sensor configuration optimization for accurate activity recognition via simulation. 140
5.3 The floor plan/space model of the two apartments: (left) an 8×8 (m^2) one-bedroom suite; (right) a 8×5.2 (m^2) studio. 146
5.4 Average of the best performance of GA, greedy, and BO with different no. sensors across black-box function queries. 147
5.5 Best performed GA, greedy and BO with different no. sensors on T1 and T2 across black-box function queries. 151
5.6 Motion sensor configurations found by GA, greedy, and BO for different ϵ values in T1 and T2. The coincident sensors in the BO sensor configuration are shown with a small offset. 152
5.7 Best performed BO with 7 motion sensors and different no. and types of IS sensors on T1 across black-box function queries with ϵ=0.5. . . . 155

Abbreviations

AAL Ambient Assisted Living.

ADL Activity of Daily Living.

BIM Building Information Modelling.

BO Bayesian Optimization.

EDA Estimation of Distribution Algorithm.

GA Genetic Algorithm.

IFC International Foundation Class.

IoT Internet of Things.

QoC Quality of Context.

QoD Quality of Devices.

QoI Quality of Information.

QoS Quality of distribution Service.

SIS Smart Indoor Spaces.

WSN Wireless Sensor Network.

Chapter 1

Introduction

The term Internet of Things (IoT) was first coined in 1999 by Kevin Ashton at the Massachusetts Institute of Technology (MIT) in the context of using Radio Frequency IDs (RFID) to track products in a supply chain [10]. This project sparked the idea of enhancing objects or "things" by allowing them to wirelessly connect and share their locations and descriptive information. The concept of IoT has been used as an umbrella term for several systems, both in academia and industry, such as smart homes, *i.e.* Smart Indoor Spaces (SIS), transportation, and power grids [59].

In particular, SIS system uses a network of connected devices (sensors and actuators) for indoor applications by integrating the physical world into computer-connected systems. Research on SIS applications has received increased attention for applications such as Ambient Assisted Living (AAL) [86], improving the efficiency and energy consumption of buildings [118], and security-related applications [77]. These applications require deploying sensors configured to monitor the indoor space: for example, to monitor occupants' presence, movements, and activities. In most applications, software services use real-time low-level sensor data to infer high-level information about the occupant's activities or the state of the indoor space. Actuators are present in some applications, which modify the state of the indoor space using the sensed information.

The devices and occupants' interaction in an indoor space create contextual in-

formation. In this context, the term *contextual information* means the perceivable information from the environment that services infer by utilizing sensor data. The type of contextual information directly depends on the kind of SIS applications, varying from occupants' traces, *i.e.* location [94] and/or activity [139], and spatial traces such as air circulation in rooms [126].

Despite the extensive research on SIS applications, there are two critical limitations to the ongoing research:

1. It is strongly data-driven;

2. It involves determining a proper sensor and/or actuator deployment configuration that fits the service's intentions.

In such systems, specific datasets are collected consisting of sensor data and their corresponding contextual information (mainly via manual annotations). The data collected are specific to the sensor capabilities and are influenced by the occupants' activities. For example, data-driven learning approaches are used for occupants' behaviour modelling in [119]. The acquisition of SIS datasets is subject to constraints such as availability, flexibility and quality. An alternative is to simulate the environment in which the SIS applications operate to potentially mitigate the layout and configuration problems, such as in [134, 103]. As pointed out in [103], simulation tools provide the extensibility, flexibility, and scalability needed to address various contexts and large-scale applications. Simulation methodologies offer inexpensive, repeatable, and rapid prototyping of SIS applications. Therefore, designers of such systems can quickly perform several experiments and collect sample datasets before the actual deployment. As shown by Bruneau and Consel [23], it takes only one hour, on average, for users with basic software development skills to simulate a simple SIS application using their methodology. In contrast, a similar task would take considerably more time and effort in real-world settings. In addition, McGlinn et al. [90] found that 67% of the SIS designers could complete their tasks by using their

simulation's visualization of contextual events. The percentage drops to 47% if the information is unavailable to the designers. Researchers are granted complete control over the SIS system in simulation. They can modify the elements of the modelled space, *e.g.* walls, doors, windows, furnishings, etc., and simulate the configuration of sensors/actuators at any number, type and location. One of the key advantages of simulation is that it allows designers to explore various scenarios of the occupants' movement, disposition, and actions that will enable them to optimize suitable sensor and/or actuator configurations to detect specific conditions necessary for the intended applications. However, one of the disadvantages of using a simulator is that it may not fully capture the complex and unpredictable nature of real-world environments, potentially leading to inaccurate results.

The sensor deployment configuration (type, number, and location of sensors) is a crucial element of design decisions. The configuration directly affects the quality of the services [114, 154] where a well-designed sensor configuration can result in detecting contextual information accurately at the lowest possible cost. However, finding the best sensor configuration requires several trials and errors, which are costly and time-consuming. The simulators offer reproducibility of specific situations for evaluation purposes which are expensive to obtain in real-world settings because simulators can be executed several times from a starting state and replicate the desired situation. Based on the reproducibility property of SIS simulation methodologies, Zhan and Haddadi [160] showed that a combination of a set of metrics for sensor configuration evaluation and a SIS simulation tool provides an inexpensive and rapid prototyping framework for evaluating different sensor configuration deployments. Furthermore, several studies such as [139, 44, 58] have proposed to automate sensor configuration design using a minimum (or fixed) number of sensors. In [139, 147], an automatic sensor configuration design is faster, easier, and significantly more accurate than manual configurations.

1.1 Thesis Statement

This thesis introduces a simulation-driven systematic sensor configuration framework that incorporates the quality of inferring contextual information as the metric for evaluating various sensor configurations. This thesis aims at introducing principled practices towards this goal. To this end, we first explain the critical aspects of the problem.

1.1.1 Smart Indoor Space Simulation Methodology

Generally, SIS simulation methodologies are required to realistically model indoor spaces, deployed sensors and/or actuators, as well as occupants. In principle, simulators should satisfy two properties: *generality* and *high-fidelity*. Generality refers to the level of abstraction that the simulator adopts in representing the simulated scenarios. High-fidelity is defined as how realistic and faithful the system can simulate the indoor space, the elements it contains, and the activities that take place in it. Simulators with these two properties enable the development of a broad range of sensor-based applications to efficiently experiment with different sensor-deployment configurations and evaluate how effective each configuration is in collecting the raw data the application needs to meet its quality requirements.

Traditionally, modelling indoor spaces require that a designer use a 3D/2D Computer-Aided Design (CAD) editor. The model's accuracy determines how well the simulator can accurately represent the real environment. The simulator software does not necessarily offer capabilities to model any intended indoor space. Moreover, this approach makes the simulator less realistic because it may not be able to take into account the complete geometry specifications accurately. Instead, specific automated methods, such as a standard data model called International Foundation Class (IFC) for Building Information Modelling (BIM) [13], can be used to generate digital twins of arbitrary indoor spaces to make the process less burdensome and more generalized.

Moreover, modelling occupants' activities are based on virtual characters interacting with objects based on some behavioural policies, like motivation-driven [88] and hierarchy-based [117, 69]. It has been shown that the hierarchy-based behaviour modelling can be adjusted to fit specific contexts, *e.g.* performing activities of daily living ADLs or office routines. Often, agent models are validated by comparing the activity distributions in simulated versus real data [88]. However, a more promising approach was proposed by Renoux et al. [117] in which they temporally compared the synthetic timeline of activities versus the real world. In this thesis, we argue that temporal evaluation of occupants' behaviour modelling is necessary for SIS applications as it is required to anticipate the time and order of activities in several applications, such as energy-saving policies of a building. Sensor data must also be validated to their real-world counterpart using time-series analysis. In [69], the authors compared synthetic motion sensor data with their real-world counterparts. They showed that their simulation methodology is accurate if synthetic and ground truth datasets are aggregated in one-hour window sizes. The fidelity of a simulation methodology and dataset aggregation granularity, in terms of occupants and sensor modelling, decides the range of the applications the methodology supports. In this thesis, the primary focus of the SIS applications is to develop a system that can be used for ageing-in-place applications. The goal is to monitor if elderly individuals can perform regular ADLs while living alone or detect falls, confusion, and other potentially dangerous conditions. Therefore, the aggregation granularity should be low-level enough to guarantee that synthetic datasets contain corresponding valuable information for further analysis.

1.1.2 Sensor Configuration Evaluation Framework

A substantial body of literature focuses on methodologies to find high-quality configurations for sensor deployment. The evaluation metrics can be divided into two main categories: 1) Metrics that consider maximizing the coverage areas of indoor

spaces by sensors, and 2) Metrics that consider the quality of contextual information detection using the sensors' data.

Several approaches have been proposed to maximize the sensors' coverage area [43, 44, 57]. However, one of the primary limitations of these approaches is that their evaluation metrics do not account for contextual information. For instance, covering the corners of a room is less critical than covering areas toward its centre, as most of the occupants' traces (location and activities) are unlikely to be present in the corners. Moreover, these approaches are unsuitable for sensors that do not cover a specific area but are sensitive to particular events as part of contextual information, such as an electricity sensor that detects power consumption.

On the other hand, various studies propose targeted coverage approaches, *i.e.* prioritizing some areas of interest and events such as [16, 76, 147, 139]. Targeted coverage approaches leverage data quality evaluation in detecting contextual information, potentially leading to better sensor configurations than area coverage approaches. These studies translate contextual information as specific valuable observations. However, these approaches have two drawbacks. First, most approaches translate contextual information as occupants' location, meaning that all the target points have the same level of importance, which dismisses the differentiation between hazardous and safe activities such as bathing and sleeping in an aged-care monitoring application, respectively. Secondly, these approaches require the collection of a corresponding dataset, which is a challenging, time-consuming, and expensive process. Using existing datasets is also limited by applicability, availability and flexibility, as different applications require various datasets.

1.2 Key Contributions

The main goal of this thesis consists of building a sensor configuration optimization framework for accurate activity recognition in indoor environments using simulation. The proposed techniques are designed to make SIS application designs faster, safer,

and more efficient. The research questions that we address in this thesis are as follows:

R1: The capability to infer some *contextual information* from raw sensor data. It is essential to determine the kinds of contextual information that can be inferred from raw sensor data.

R2: The ability to evaluate different sensor configuration deployments by assessing the quality of detecting contextual information.

R3: A SIS *simulation methodology* that can simulate various applications and produce high-fidelity synthetic datasets.

R4: A novel simulation-driven sensor configuration optimization component for SIS applications, considering the quality of detecting contextual information as the objective function.

In summary, the key contributions of the present work are:

- **Development of a high-fidelity SIS simulation methodology**: In the present work, we present a SIMulator for Smart Indoor Spaces (SIM_{sis}), that makes the following important contributions to the state-of-the-art:
  - We propose accurate modelling of the indoor space, relying on BIM in the IFC format, the de-facto representation standard of building information models. IFC enables the accurate specification of the space geometry and the furnishings and objects in it. Our simulation methodology augments the IFC building model with specifications of the affordances of the objects in the space, so that virtual occupants, given a set of objectives, move through the space and interact with the objects to accomplish their goals.
  - We develop a sensor modeling component that realistically models popular off-the-shelf sensors in SIS applications.
  - We investigate the fidelity of the simulation methodology in terms of occupants' behavior and sensor readings by developing a digital twin of a real-world SIS application.

- **Sensor Configuration for Accurate Activity Recognition in SIS applications using Bayesian Optimization and Building Simulation**: We present an automatic sensor configuration component based on Bayesian optimization and building simulation for efficient identification of sensor configuration that better supports accurate activity recognition in an aged-care facility. The contributions of this component are:

  - We cast the problem of configuring sensors in an indoor space as a Bayesian optimization problem.
  - We propose a simulation-based assessment and optimization framework that allows for using synthetic activities and movement trajectories, instead of real-world traces that are difficult to collect
  - We demonstrate the efficacy of the proposed methodology in finding different sensor configurations in the digital twins of two real-world apartment buildings.

1.3 Thesis Outlines

This dissertation takes the form of a paper-based thesis, wherein each chapter represents independent research paper(s). This format allows for a concise presentation of various aspects of our research topic. Each chapter is self-contained, encompassing an introduction, methodology, results, and conclusion, contributing to an overall comprehensive exploration of our research findings.

This dissertation is organized into six chapters. Following the introduction Chapter 1, Chapter 2 describes two proof of concepts that examine building data-driven models for contextual information inference using raw sensor data. Then, the chapter investigates using contextual information to design high-level sensor configuration evaluation metrics. Chapter 3 presents a unified taxonomy for the definition and development of SIS simulation methodologies, that can cluster related work found

in the literature into meaningful groups. It followed by identifying requirements for developing a simulation that supports a wide range of SIS applications. Chapter 4 presents a SIS simulation methodology and examines its fidelity in a case study. Chapter 5 proposes a simulation-driven sensor configuration component in SIS applications using Bayesian Optimization (BO). Then, we comprehensively evaluate the component's scalability and modularity in this chapter. Finally, Chapter 6 summarizes the contributions and discusses potential future research avenues.

Chapter 2

Using Low-level Sensor Data for High-level Sensor Configuration Evaluation

There have been several examples of using SIS applications in academia and industry over the past decade. In most applications, a predictive model is designed to infer the desired contextual information from raw sensor data. For example, there is a rich and growing corpus of work on estimating the occupancy schedules of different rooms in a building. However, some outsized challenges inhibit the models' safe, reliable, and generalized deployment in real-world environments. This chapter presents two proof of concepts. First, we study the application of two data-driven models for SIS occupancy estimation, as contextual information, using sensor readings. This study shows great promise for applications in both residential and commercial buildings that data-driven models can predict certain contextual information using dedicated and common occupancy-indicative sensors. Secondly, we study the effectiveness of using indoor occupants localization as an inferred contextual information to design high-level meaningful metrics for evaluating different sensor configuration deployments.

This chapter focuses on R1 and R2 research questions, specified in Section 1.2, respectively:

- **R1**: The capability to infer some contextual information from raw sensor data. It is essential to determine the kinds of contextual information that can be

inferred from raw sensor data

- **R2**: The ability to evaluate different sensor configuration deployments by assessing the quality of detecting contextual information.

2.1 Data-driven Models for SIS Occupancy Estimation

2.1.1 Introduction and Background

Building occupancy is one of the main factors determining its energy consumption [35]. There are different types of spaces in a building, each with a unique occupancy trace, *i.e.*location and activity. Despite the fact that huge variations in occupancy traces over space and time, the whole building is often treated as a uniform environment controlled with a fixed ventilation rate and static schedules for air conditioning and lighting, thereby wasting much energy in conditioning empty or partially occupied spaces. However, fine-grained occupancy trace information is unavailable in buildings, and camera-based occupancy monitoring is intrusive and costly at scale. This has given rise to several techniques that are less intrusive, employing other modes of sensing to estimate the occupancy state (*i.e.* empty or occupied) of different spaces within the building [52, 5, 141].

Despite recent advances in detecting human presence in buildings, existing techniques offer limited predictive power when discerning the number of occupants. This is an inherently difficult problem owing to the highly uncertain and complex nature of occupancy trace dynamics. To address this, some researchers have utilized physics-based models of the indoor environment, such as the Resistance-Capacitance (RC) model [65, 121], to infer the occupancy schedules of a building from experimental or simulated data. The advantage of such models is their high granularity of temperature modelling, but they are high-dimensional and must be customized for each room. In addition, it is difficult to distinguish the effect of occupancy trace from

other latent factors rendering them computationally expensive. Data-driven models can also be developed to discern the number of occupants of the many rooms in a building [68, 5, 7]. These models are easier to build and can substitute complex physics-based models with an insignificant loss of prediction accuracy [163]. Most related work focuses on the application of machine learning and time-series analysis techniques to detect human presence and determining the number of occupants in a single room or a multistory building [42, 68, 107, 28, 163, 113]. Our approach is similar to these data-driven modelling techniques with two main differences. First, we adopt particle filters and dynamic neural networks, which are powerful techniques for estimating hidden states in nonlinear dynamical systems. Second, we compare the accuracy of these methods on two datasets with common and dedicated sensors. To our knowledge, these techniques have not been compared in previous work with respect to the accuracy of predicting the number of occupants.

This section explores the application of two advanced data-driven occupancy modelling techniques which fuse data from multiple occupancy-indicative sensors, including dedicated occupancy monitoring sensors and sensors that are commonly available in commercial buildings (*e.g.* the HVAC sensors). We postulate that over a broad spectrum of sensor modalities, a person in a building will always leave a *trace*, detectable in installed sensors readings through sufficient analysis. We evaluate the accuracy of these techniques on a small number of rooms in two test buildings and show that these models are capable of predicting the number of occupants in every room with a relatively high accuracy. This chapter presents our preliminary results, which support our claims that data-driven models using raw sensor data are capable of occupancy trace recognition.

2.1.2 Methodology

Testbed

Our testbed is comprised of two buildings[1] equipped with different types of occupancy-indicative[2] sensors as shown in Table 2.1. Building 1 [72] is a residential building equipped with dedicated sensors including a Volatile Organic Compounds (VOC) sensor measuring indoor air quality, a Bluetooth Low Energy (BLE) receiver reporting the number of BLE beacons in its proximity, a calendar feed indicating scheduled occupancy events, and a flag for distinguishing between weekdays and weekends. The BLE beacons are attached to key fobs carried by the occupants. The sensor measurements and ground truth occupancy data are available for seven weeks and the maximum occupant count recorded during this period is seven. Building 2 is a large commercial building with a Building Management System (BMS) that archives measurements of damper position and CO_2 sensors which are integral parts of the Variable Air Volume (VAV) system that exists in each zone. These sensors are commonly available in a commercial building with a centralized HVAC system. We have access to 16 days worth of VAV data and ground truth occupancy data from four different rooms in this building. The maximum numbers of occupants recorded in these rooms during this time period are 29, 35, 39, and 67, respectively.

Particle Filter

Particle Filter (PF) is a powerful and widely used method for solving optimal state estimation problems in non-linear non-Gaussian scenarios [140]. The PF algorithm is a special case of the Sequential Monte Carlo (SMC) algorithm. Sequential data analysis methods have been used in the past to solve the binary occupancy detection problem [107, 28]. Our filtering problem of interest is how to incorporate measurements of different sensors to estimate the number of occupants in each room.

[1] Both data sets have been previously used for occupancy detection [46, 121].

[2] A sensor is deemed occupancy-indicative in a given space if there is a correlation between its readings and the occupancy state of that space.

Table 2.1: Our testbed for the evaluation of our contextual information inference prototype

Datasets	Sensors
Building 1	VOC: *Volatile organic compounds concentration*
	BLE: *No. BLE beacons in the range of the receiver*
	CAL: *Calendar with scheduled events*
	DAY: *Flag indicating a weekday or a weekend*
Building 2	CO_2: *Carbon-dioxide concentration*
	Damper POS: *VAV Damper position*

In general, two models are required to estimate the hidden state using a PF: a model describing the evolution of the state with time (the *system model*) and a model relating the noisy measurements to the hidden state (the *measurement model*). The PF represents the posterior occupancy state of a room by an S number of random state samples or *particles* ($x_t^{[s]}$ with $1 \leq s \leq S$) drawn from the system model, and assigns weights to these samples according to the measurement model which represents the reliability of each sensor. Assuming a Markov process of order one, the recursive Bayesian estimation can be written as:

$$p(X_{0:t}|Z_{1:t}) = p(X_t|Z_t)p(X_t|X_{t-1})p(X_{0:t-1}|Z_{1:t-1}), \tag{2.1}$$

$$\overset{Bayes}{=} \eta\ p(Z_t|X_t)p(X_t|X_{t-1})p(X_{0:t-1}|Z_{1:t-1}), \tag{2.2}$$

where X_t is a categorical random variable modeling the hidden state, i.e., the number of occupants in the room at time t, and $Z_t = [Z_t^1, \cdots, Z_t^M]$ is an M-dimensional random variable modelling measurements of M different sensors at time t. If the maximum number of occupants in the room is N, then X_t takes values from $0, 1, ..., N$, and $p(X_t = k)$ denotes the probability of having $0 \leq k \leq N$ occupants at time t. Note that Z_t only depends on X_t; hence, given X_t, Z_t is independent of $X_{t'}, Z_{t'}$ with $t' \neq t$.

Learning system and measurement models: Our goal is to learn the system model, denoted $p(X_t|X_{t-1}, Z_t) = p(X_t|X_{t-1})$, and the measurement model, denoted $p(Z_t|X_t)$, from the training data which is 50% of our dataset. Once we have these two models, we can estimate the probability of having a certain number of occupants at each time given the measurements of a set of sensors.

The system model encodes the probability of going from X_t to X_{t+1}. To build the system model, we assume transition probabilities are time-dependent. To estimate the transition probability at a given time, we take into account all state transitions that happened in an interval of length $\tau = 10$ minutes (*i.e.* τ is the unit of time).

For the measurement model, we assume that for every sensor i, $p(Z_t^i|X_t = k)$ can be approximated by a Gaussian distribution with a conditional mean and variance with respect to the value of X_t which can be estimated from the training data. We can write

$$p(Z_t^i|X_t = k) \sim \mathbf{G}(\ \widehat{\mathbb{E}}(Z_t^i|X_t = k), \widehat{\mathbb{V}}(Z_t^i|X_t = k)\ ), \tag{2.3}$$

$$p(Z_t|X_t = k) = \prod_{i=1}^{M} p(Z_t^i|X_t = k), \tag{2.4}$$

where $\widehat{\mathbb{E}}(Z_t^i|X_t = x)$ and $\widehat{\mathbb{V}}(Z_t^i|X_t = x)$ are the estimated conditional mean and variance of the ith sensor, respectively.

Algorithm 1 describes the PF algorithm. Note that the more accurately a particle estimates the state, the higher weight (*importance*) it would get.

Dynamic Neural Networks

Dynamic neural networks can be also used for input-output modeling of nonlinear dynamical systems [39]. We adopt nonlinear autoregressive network with exogenous inputs (NARX) which is a recurrent dynamic network with feedback connections enclosing several layers of the network. The network is similar to a feedforward network, but in addition to the regular input Z_t it is fed with d previous output variables, i.e., $\{X_{t-1}, X_{t-2}, \cdots, X_{t-d}\}$, where d is defined in a way that the network

Algorithm 1 The Particle Filter Algorithm

Inputs: S: Number of particles, T: Length of the data set,
$Z_{1:T}$: Measurements of the sensor(s) up to time T
Output: $\mathcal{X}_{1:T}$: Set of particles up to time T

```
1: X_t = {}        ∀t ∈ {1, ··· , T}
2: for t = 1 : T do
3:    X̄ = {}
4:    for s = 1 : S do
5:       Sample x_t^[s] ~ p(X_t | X_{t-1} = x_{t-1}^[s])
6:       w_t^[s] = p(Z_t | X_t = x_t^[s])
7:       X̄ = X̄ + {x_t^[s], w_t^[s]}
8:    end for
9:    for s = 1 : S do
10:      Draw s with probability ∝ w_t^[s]
11:      Add x_t^[s] to X_t
12:   end for
13: end for
14: return c*
```

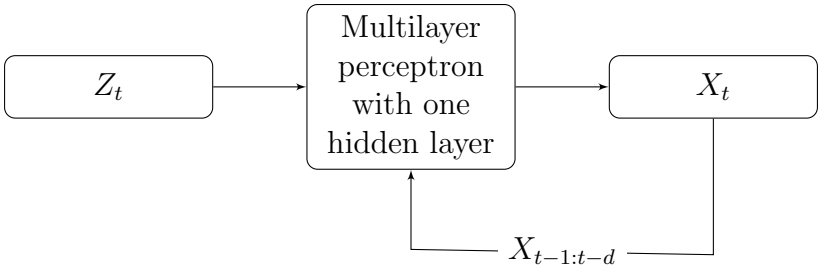


Figure 2.1: Time Series Neural Network block diagram.

receives the last 10-minute outputs for prediction. Figure 2.1 shows the network architecture. The network is trained using the Levenberg-Marquardt algorithm [95] utilizing MATLAB Neural Network Toolbox [89]. We use 50%, 30% and 20% of the data for training, validation, and testing, respectively.

2.1.3 Results

We run the proposed methods on our testbed to estimate the number of occupants in each room. Table 2.2 shows the performance of both PF (with 1000 particles) and NARX methods on the two datasets measured by Root Mean Squared Error

Table 2.2: The RMSE of PF and NARX (Ri: the ith room in Building 2)

Method	**Building 1**	**Building 2**			
		R1	**R2**	**R3**	**R4**
PF	0.4	1.5	0.8	1.4	2.9
NARX	0.3	0.4	0.4	0.5	0.8
Maximum no. occupants	7	29	35	39	67
Average no. occupants	0.4	2.7	2.5	3.6	7.4
Peak-to-avg. occ. ratio	0.06	0.09	0.07	0.09	0.11

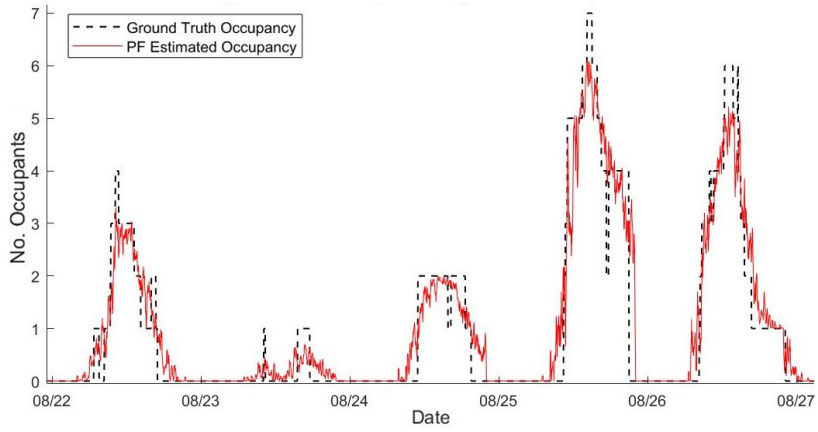


Figure 2.2: Estimated occupancy count using PF in Building 1 (RMSE = 0.4)

(RMSE). It can be readily seen that the NARX method outperforms the PF method in all rooms. Nevertheless, the PF method also yields a relatively accurate prediction of the number of occupants when compared to the maximum occupancy level of each room. Interestingly, the prediction errors of both methods are higher in those rooms that the peak-to-average occupancy ratio[3] is lower, suggesting that the difference in the prediction errors cannot be explained by the maximum or the average number of occupants alone.

Figures 2.2, 2.3, 2.4, 2.5 show the PF and NARX predictions and the ground truth occupancy data in Building 1 and Building 2 (Room 4). As depicted in Figure 2.2, the prediction error of the PF algorithm is higher when there is a short occupancy event,

[3]The peak-to-average occupancy ratio of a room is defined as the maximum number of occupants divided by the average number of occupants in that room.

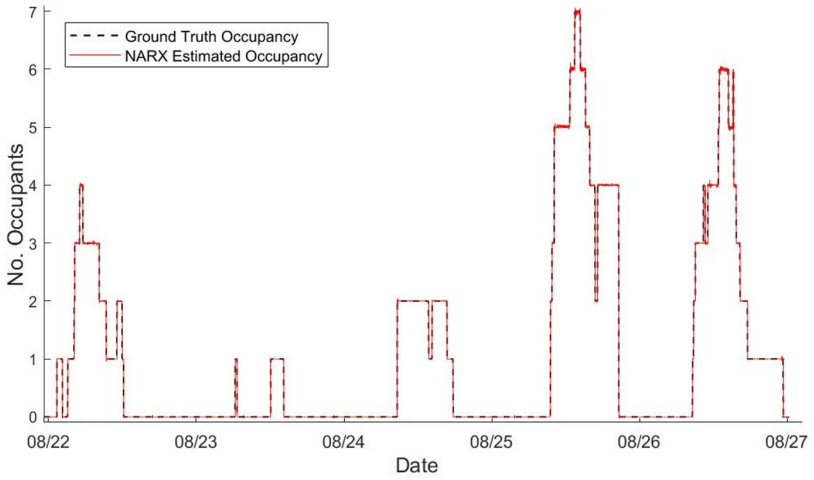


Figure 2.3: Estimated occupancy count using NARX in Building 1 (RMSE = 0.3)

e.g., on August 24th, the number of occupants decreased to one, and then quickly returned to two. The algorithm missed this rapid change in occupancy. This can also be seen in Figure 2.4. In addition, the predicted occupancy count decreases gradually when the room becomes unoccupied. These two observations can be attributed to the fact that the CO_2 concentration level builds up and drops slowly. Furthermore, the PF algorithm predicts the occupancy slightly ahead of the actual incident in Building 1. We attribute this to the fact that we take scheduled events (from the calendar) into account when we train the models. Thus, when there is a scheduled event, the probability of seeing a change in the occupancy state increases.

It should be noted that although NARX outperforms PF in both datasets, its predicted occupancy starts to fluctuate when number of occupants increases to seven in Building 1; this is evident in Figure 2.3. We further study how the room occupancy level affects the prediction accuracy of our methods. In particular, running the PF method with 1000 particles on Room 1 of Building 2, gives an RMSE of 0.72 when there are only 23 occupants in that room. Given that the maximum occupancy of Room 1 is 29, we can conclude that the PF algorithm makes more erroneous predictions as the number of occupants increases.

The performance of the PF method depends on the number of particles that it

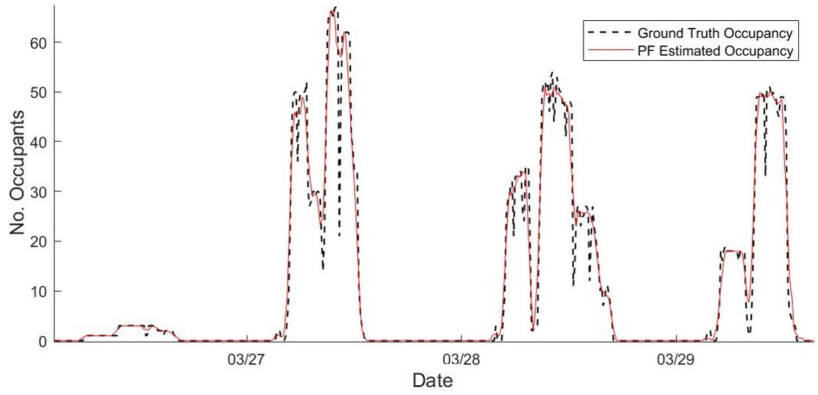


Figure 2.4: Estimated occupancy count using PF in Building 2, Room 4 (RMSE = 2.9)

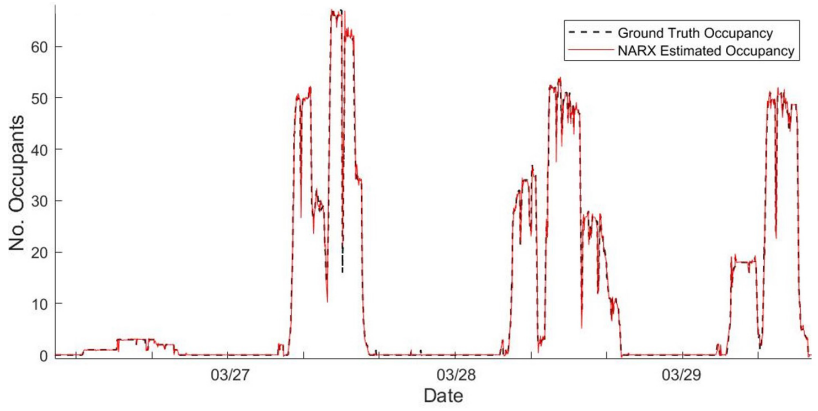


Figure 2.5: Estimated occupancy count using NARX in Building 2, Room 4 (RMSE = 0.8)

uses to represent the posterior. To study the sensitivity of the results to the number of particles, we run the PF algorithm with different numbers of particles on Building 2 (Room 1 and Room 4) where the algorithm performed relatively poorly. Table 2.3 shows the average RMSE over 10 runs for each case[4]. As expected, increasing the number of particles improves the prediction accuracy of the PF method, but this comes at the cost of increasing its computation time.

Discussion

Data-driven occupancy models show promising results in both residential and commercial buildings equipped with dedicated and common occupancy-indicative sensors.

[4]The standard deviation is very small ($\sim$ 50s).

Table 2.3: The RMSE of PF for different numbers of particles.

No. Particles	**Building 2 Room 1**	**Building 2 Room 4**	**Avg. Computation Time (s)**
500	4.7	4.6	2710.1
1000	2.7	3.6	5751.0
2000	2.4	3.2	11456.0
3500	1.7	2.8	21288.3
5000	1.7	2.7	33398.0
7000	1.4	2.4	58759.4

Our findings confirm that black-box data-driven models can be used to predict contextual information accurately. As shown in Table 2.2, both PF and NARX models exhibit better performance on Building 1. This can be partly attributed to the dedicated occupancy-indicative sensors that were installed in this building; however, it can also be due to the fact that we had fewer occupants in this building. We cannot draw a conclusion without analyzing data from a building that has a higher typical occupancy level and is equipped with dedicated sensors. In any case, both techniques also exhibit acceptable performance on Building 2, which has a higher occupancy level and only includes commonly available sensors.

This section shows that occupants leave distinct enough patterns in sensor reading space, so that data-driven models can detect specific contextual information accurately.

2.2 Using Contextual Information Inference for Evaluating Different Sensor Configurations

2.2.1 Introduction and Background

This section, while not dismissing the use of analytical techniques such as optimization, focuses on two categories of tools that can be used for pre-deployment sensor configuration evaluation. **Simulators** are, by far, the most popular approach for SIS

application deployment evaluation. There is a wide range of simulators such as [15, 112, 135] mainly focusing on low–level aspects of the underlying Wireless Sensor Network (WSN) network, *e.g.* message passing, protocols. However, context–aware simulators, *i.e.* simulators with contextual information modeling capability, provide high–level analysis of WSNs, which is desirable for SIS applications, and may incorporate several techniques such as machine learning, and domain-specificity, such as healthcare scenario simulators.

On the other hand, **emulators** are a combination of hardware implementation and software simulation in order to mimic a real–world WSN under controlled conditions. MiNT [38] is an example of emulators, that offers small–size testbeds for faster evaluation. Although emulators provide more realistic evaluation techniques, the costs of developing and evolving WSN configurations in emulators are higher.

Going beyond coverage and/or energy criteria, efforts have been made to introduce middleware systems with Quality of Context (QoC) support for SIS applications [56] in order to incorporate high-level evaluation of sensor configurations. Three parameters are mainly involved in QoC [24]: Quality of Information (QoI), Quality of Context (QoC), and Quality of Information (QoI). For example, Nazário et al. [100] used up-to-dateness, precision, completeness, and significance variables of sensor readings to quantify the QoC parameter in order to evaluate an e–health sensor platform, using a simulator called Siafu. They found that the QoC parameter can assist in detecting abnormal SIS behaviors related to the mentioned variables.

Less attention has been paid to high–level evaluation of sensor configurations. In this section, we present a methodology to aid the developer to interpret how QoC impacts the overall application behavior. The presented integrated methodology takes advantage of context–aware simulation environments such that, given a floor plan, it assists in the quick design and development of a SIS application. It considers the evaluation of alternative deployment configurations by providing a high–level application analysis module, which focuses on QoI parameters.

We chose to base our work on extensions to the CASi [33] simulator. CASi is an easy-to-use and open-source simulator that provides a user interface, *i.e.*Floor Plan Editor, for modeling the physical space and the deployment of sensors in it, and then simulates the occupants' movements and activities to generate sensor event traces that can be consumed to detect some contextual information. We have made two important extensions to the original CASi framework. First, we extended it to support the automated ingestion of XML files representing the space and deployment configuration of sensors, instead of requiring a designer to specify them via the CASi UI. We have further extended CASi so that, in addition to simulating the occupants' activities, it can also consume an externally provided occupants activities trace and generate the corresponding sensor events as output.

The output sensor-event streams are then analyzed to detect contextual information. The analysis can be modified in order to view results based on application–specific requirements, but for the rest of this section, we consider occupant localization applications for indoor spaces as an example of contextual information detection.

The contributions of this section are: 1) We present a streamlined simulation of a SIS application deployment, based on a model of the space and the deployed sensors, and simulated and/or input activity traces; 2) We evaluate different sensor configuration deployments from two high-level perspectives, *i.e. context–aware* and *overall* performance of localizing occupants (as a contextual information).

2.2.2 Methodology

Our methodology integrates and supports four modules (Figure 2.6): 1) Deployment Configuration Specification, 2) Occupant Activity Simulation, 3) Sensor-Event Observation and Analysis, and 4) Deployment Configuration and Exploration of the Configuration Space.

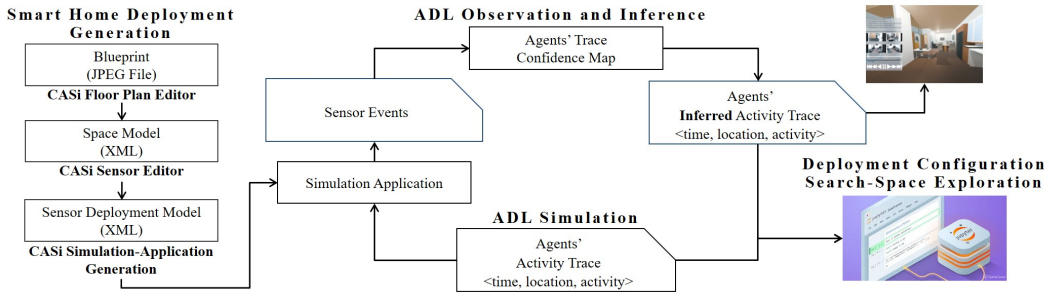


Figure 2.6: Overall architecture of a prototype for sensor configuration evaluation.

Deployment Configuration Specification

Through the CASi Floor Plan Editor, users can develop a model of the indoor space where the activity-recognition application will be deployed (Figure 2.7). The space model defines the floor plan, *i.e.* rooms, doors, obstacles, and interactive objects such as beds and chairs. Users can also superimpose on this floor plan a sensor deployment configuration through the CASi Sensor Editor. The sensor deployment configuration adds to space model a description of the sensors to be deployed in the space and feeds their events to the SIS application under examination, *e.g.* location and type. Finally, users are able to establish an MQTT broker [97] in order to send sensor events to another host for further analysis, through the CASi Simulation-Application Generation. We extended the original CASi framework, with an intermediate layer, through which the user interface produces a serialized representation of the model in XML. Models in this format can be input to the CASi Simulation-Application Generation module to generate a simulation. In this manner, users can easily clone and modify models to produce a family of simulations, each one with slightly different sensor deployment configurations.

Occupant Activity Simulation

This module takes as input a simulated activity trace, in the form of a stream of {*time, location, activity*} tuples, and converts it to the format expected by the CASi Simulation Application. The Simulation Application also reads as input the XML

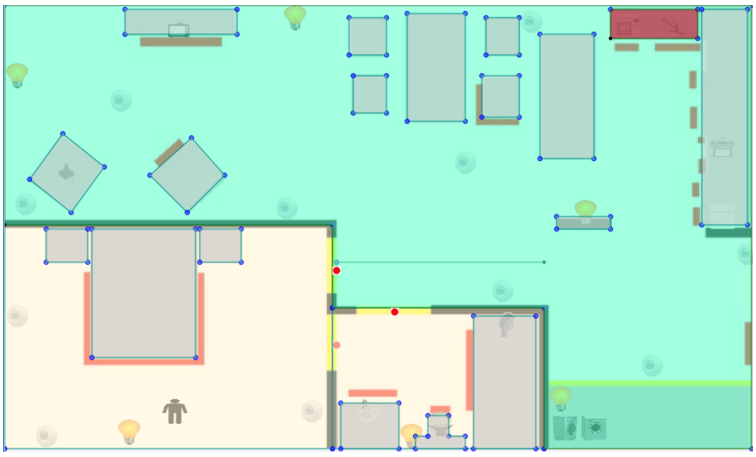


Figure 2.7: The CASi floor plan editor shows rooms (large polygons), doors (red dots), obstacles/objects (grey polygons) and their interactive sides (red lines).

file describing the space model and sensor-deployment configuration. The output of Simulation Application is a stream of sensor events, subsequently passed to the Observation and Analysis module via the MQTT broker.

Sensor-Event Observation and Analysis

This module represents the activity-recognition systems that our methodology is designed to support by enabling the simulation of their performance in different deployment scenarios. The underlying assumption is that these subject systems, henceforth *systems-under-test*, infer the agents' activity traces based on the input stream of sensor events, as produced by the simulation module described above. In this section, we illustrate our methodology using as system-under-test the Smart-Condo™ localization system described in [94].

Deployment Configuration and Exploration of the Configuration Space

By running the simulation and comparing the inferred activity trace produced by the system-under-test and against the synthetic ground truth, input to the simulation, our methodology enables the developers of the system-under-test to evaluate the system in different sensor-deployment configurations. This module computes two different indicators, *i.e. overall performance*, and *context-aware performance* (described next),

represented as error metrics and visualized at different levels of granularity. The visualization is by means of heatmaps, to capture the space-/position-related relative performance of the specific sensor configuration. The metrics are as follows:

- **Overall Error Heatmap (OEH)**: We define OEH as an $N \times M$ matrix, where N and M correspond to the length and width of the space in integer multiples of a reference length unit (equal to 1 meter in our examples). Each element in the matrix, $e_{i,j}$ $(0 \leq i < N, 0 \leq j < M)$ represents the average localization error resulting from the given sensor deployment conditioned on the occupant location being at grid point (i, j). Clearly, the metric cannot be defined if the occupant was always relatively far from (i, j) with respect to its synthetic ground-truth location. Specifically:

$$e_{i,j} = \frac{\Sigma_{p=1}^{P} \Sigma_{t=1}^{T_p} \{||(x_{est}, y_{est})_t^p - (x_{gt}, y_{gt})_t^p|| * A_t^p\}}{\Sigma_{p=1}^{P} \Sigma_{t=1}^{T_p} A_t^p} \tag{2.5}$$

$$A_t^p = \begin{cases} 1, & \text{if } ||(i,j) - (x_{gt}, y_{gt})_t^p|| \leq \text{r} \\ 0, & \text{otherwise} \end{cases},$$

  where P is the number of occupants in the simulation; T_p is the (discretized) total amount of time of occupant p simulation (i.e., being present in the simulated space); $(x_{est}, y_{est})_t^p$ and $(x_{gt}, y_{gt})_t^p$ are the estimated and synthetic ground-truth location of the occupant p in time t. A_t^p is a gating function that infers that the occupant was at grid point (i, j) if their synthetic ground-truth location is no further then r from from (i, j). r depends on (and is set equal to) the radius of the deployed motion sensors and it represents an intrinsic uncertainty due to the sensing range of the sensors.

- **Context–Aware Error Heatmap (CAEH)**: This indicator is a time-average alternative of the OEH indicator (which is event-average). In other words, if an occupant is rarely at a location and the error is high when at that location,

the error is of no particular importance. On the contrary, high error matters in highly visited locations. A designer may be willing to accept a high OEH at some locations if the CAEH at those locations is low. Specifically, CAEH, $e'_{i,j}$, is defined as follows:

$$e'_{i,j} = \frac{\Sigma_{p=1}^{P}\Sigma_{t=1}^{T_p}\{||(x_{est}, y_{est})_t^p - (x_{gt}, y_{gt})_t^p|| * A_t^p\}}{\Sigma_{p=1}^{P}\Sigma_{t=1}^{T_p}1} \tag{2.6}$$

It should be noted that Equation (2.6) considers the importance of each coordinate based on the average behavior of the occupants.

2.2.3 Results

Testbed

In this section, we illustrate our simulation-based methodology for evaluating the deployment configurations of smart applications in indoor spaces, using our own Smart-Condo™ activity recognition module as the system-under-test and considering its deployment in two different indoor spaces, shown in Table 2.4. The table shows the area and the type of activity traces for each environment.

Smart-Condo™ [130] is a one-bedroom condo equipped with various sensors, i.e. motion sensors, Bluetooth Low Energy (BLE) beacons, and pressure sensors. The condo, which is designed to simulate a home environment, allows researchers from different disciplines, such as medicine, rehabilitation, and computer science, to focus on health–related studies from their own perspectives. We obtain occupants' activity trace using ground truth data in [94], where 10 participants were asked to perform a scripted sequences of activities of daily living in the condo.

Mohammadi et al. [93] have utilized the Waldo Library at Western Michigan University as their testbed for indoor localization using Bluetooth Low Energy (BLE) signal strength. We use their publicly available dataset [20] and get occupants' activity trace using two out of four participants that were asked to walk inside of the

Table 2.4: Our testbed for sensor configuration evaluation proof of concept.

Name	**Description**
Smart-Condo™[130]	Area: $\sim 70\ m^2$ occupants' trace: 10 participants performing ADL
Waldo Library [93]	Area: $\sim 3240\ m^2$ occupants' trace: 2 participants walking in a library

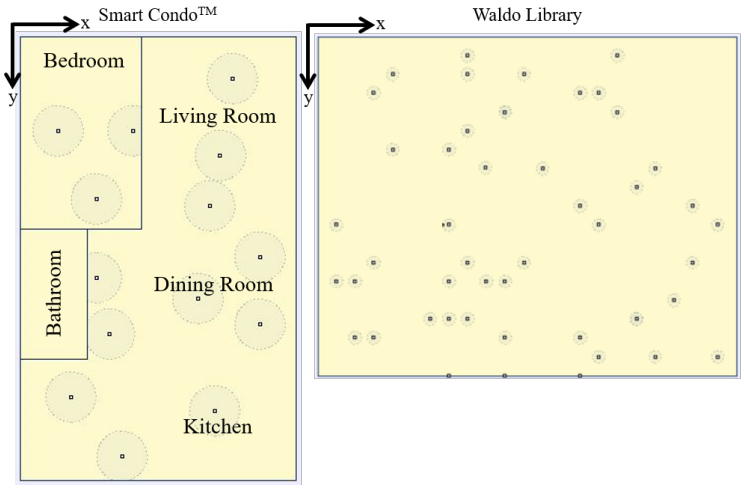


Figure 2.8: Floor plan and motion sensor locations (dots) for the two real-world spaces. The radius of motions sensors is 60 *cm*.

library.

Simulations

Using the *Deployment-Configuration Specification* module in our methodology, we create a space model for both environments and acquire a sensor-deployment model by defining several motion sensor locations. The sensor configuration in Smart-Condo™ and Waldo library consists of 14 and 48 motion sensors in random locations, respectively. Figure 2.8 shows the locations of the motion sensors in the two spaces.

We run our methodology on the testbed for all of the participants using the *Occupant Activity Simulation* and *Sensor-Event Observation and Analysis* modules. Figure 2.9 shows our estimated and synthetic ground-truth location, for one example

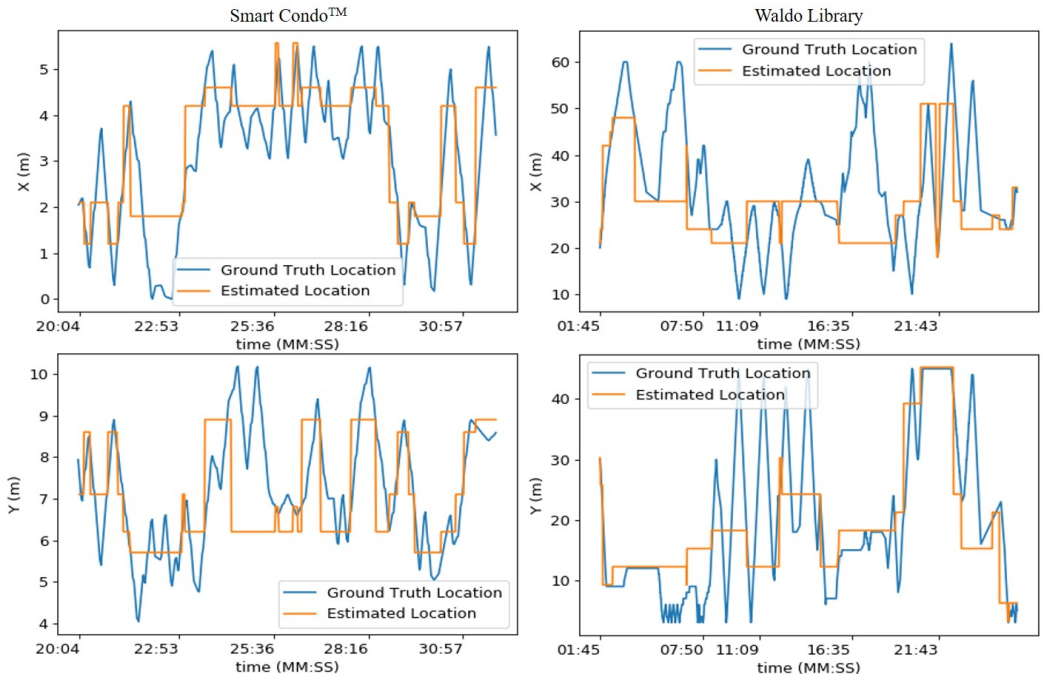


Figure 2.9: Estimated and synthetic ground-truth location of one occupant in each of our testbed environments. For better readability, the X and Y coordinates are shown separately (upper is X, lower is Y).

participant in each environment. We evaluate sensor deployment in our testbed from three perspectives: overall performance, context–aware performance, and different granularity of information.

Overall Performance

We calculate the average of Mean Squared Error (MSE) for all four estimations, running the simulation for all occupants. In Smart-Condo™, MSE in X and Y dimensions are 1.16 and 2.09 respectively, while the errors are 14.25 and 6.28 in Waldo Library. The error values indicate that the sensors in Smart-Condo™ and Waldo Library are not placed well in Y and X dimensions respectively based on the occupants' traces.

In terms of OHE, Figure 2.10 shows the average error for all occupants. Given this figure, we infer that, in order to improve the performance of our system-under-test, the deployment configuration would benefit from additional sensors, placed in areas with more error, or, if no additional budget is available, the current sensors could be

placed in a different configuration.

Context–Aware Performance

We calculate the CAEH for both test spaces (Figure 2.11). The figure shows the average error for all the occupants based on their traces. The occupants in Smart-Condo™ spent more time in the kitchen area where we see the CAEH error is high. A designer noticing the high CAEH and the high OHE in the kitchen area will infer that the particular area should be a priority if more sensors can be deployed, or if the sensors are to be re-positioned. By comparison, other areas around the center of the Smart-Condo™ have a high OHE but the CAEH is low; this implies that this area would be a less pressing area to pay attention to by putting (or re-positioning) sensors.

Information Granularity

Depending on the precision requirements for the system-under-test, the OHE and CAEH matrices can be calculated at different levels of spatial granularity. For example, in a space like the Waldo library, we may be interested to find areas with fewer occupants, in order to adjust the HVAC system. Therefore, a large grid-based granularity would suffice. That is why we demonstrate OHE and CAEH matrices in a grid for the library (every grid is a $10m \times 10m$ area).

2.2.4 Discussion

Simulation of SIS applications using different sensor configurations is an essential tool for researchers to gain a better understanding about, and more confidently forecast, the performance of their applications. The simulation tools provide a fast, cheap and easy-to-use prototyping framework for evaluation of SIS applications before the actual deployment. Despite the traditional approaches, replicating the occupants traces of the indoor spaces where the application is deployed and comparing them against the traces inferred by the application under test enables us to evaluate the sensor

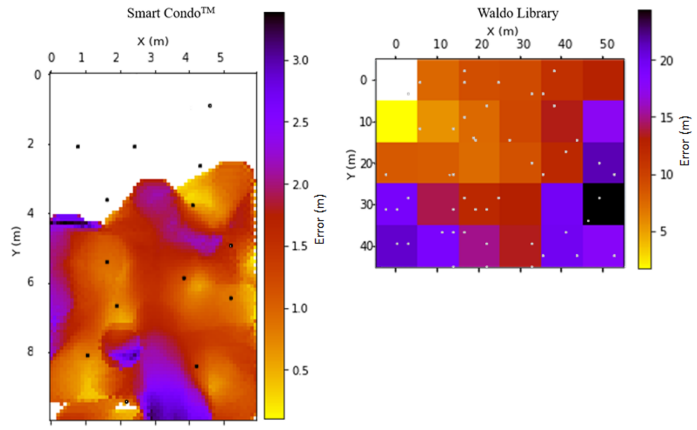


Figure 2.10: OHE calculation for two test cases; yellow and black colors show minimum and maximum errors respectively; white color shows areas for which we lack information. We increase the information granularity for Waldo library to $10m * 10m$ blocks level.

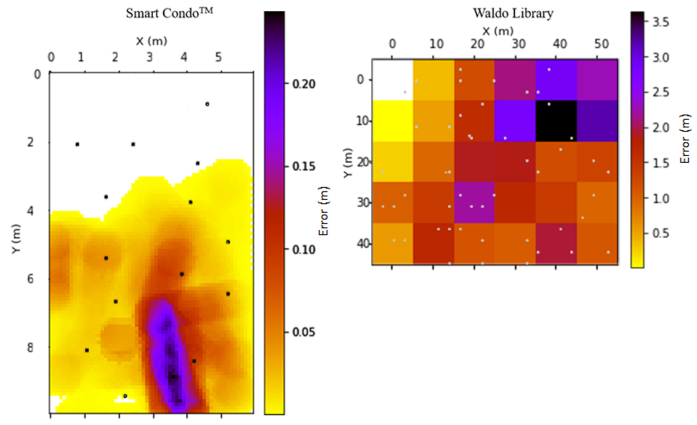


Figure 2.11: CAEH calculation for two test cases, with color range similar to OHE calculation (Figure2.10). Here we also increase the information granularity for Waldo library to $10m * 10m$ blocks level.

configuration from high-level indicators, overall performance, and context–aware performance, and at different levels of granularity. Notice that this process is feasible when only using a simulator due to the implications of real-world implementations and modifications.

Overall-performance analysis becomes important when no contextual information is available. Therefore, by generating random traces, we focus on the average MSE, comparison diagrams (like Figure 2.9(a)), and OHE parameters. In the Smart-Condo™ for example, it can be seen from Figure 2.8 and Figure 2.9(a) that our sensor deployment causes our localization method to err in the y dimension when $y >= 9.7m$ and $y <= 5.5m$. Keeping in mind that the MSE value in this dimension is higher, these observations help us to evaluate the distribution of sensors in each dimension. Moreover, by using the OHE indicator (Figure 2.10), we are able to observe places in the floor plan with higher error. This figure suggests that placing more sensors in the kitchen and the bathroom would potentially increase the overall performance of our application. Nevertheless, in order to have a generalized analysis, the set of occupants' traces should include various traces, which is dramatically faster and easier to obtain using simulation tools.

When we have contextual information available, context–aware performance analysis provides a different evaluation perspective. In the Smart-Condo™, the occupants' traces are replicated using the study that is conducted by Mohebbi et al. [94], where 10 participants were asked to follow a script of activities of daily living. As illustrated in Figure 2.11, the CAEH indicator suggests that the sensor deployment in Smart-Condo™ is mostly deficient in the kitchen area.

We conducted the same analysis of the localization method (proposed in [94], in the context of the Waldo library. In addition, we can adjust the granularity level of information to adapt the evaluation more to the application in hand. The deployment of sensors in a library would very likely be for the purpose of minimizing energy consumption in the building, as opposed to actually recognizing the specific locations

of the library members. Therefore, estimating the occupants' trace within $10m * 10m$ blocks would suffice for such spaces and applications. Moreover, a health–care application in Smart-Condo™ could potentially be more concerned with the amount of time elderly people spend in rooms such as bathroom. Thus, estimating agents' trace within rooms level granularity is adequate.

The OHE and CAEH metrics can be used to find areas that occupants did not visit and, thus, remained untested. New traces can be generated using the activity simulation module so that to make sure our sensor deployment is evaluated completely.

2.3 Conclusions

Modern buildings are increasingly being instrumented with a myriad of networked sensors to gain more insight into their operation, particularly occupants' behaviour and activities. On the other hand, context-aware simulation methodologies offer quick, cheap and easy prototyping of such systems for analyzing the performance of different sensor configurations in terms of high-level metrics.

This chapter investigated the effectiveness of two essential components in SIS sensor configuration evaluation pipeline. First, we showed that specific contextual information, *e.g.* occupants' location, could be recognized by analyzing raw sensor readings via data-driven models. Second, we showed that we could use the recognized contextual information to evaluate different sensor deployment configurations. Based on our proof of concept results, a much broader study of simulation-based context-aware sensor configuration evaluation for SIS applications is warranted.

Therefore, the rest of the dissertation is focused on defining a unified taxonomy of a SIS simulation methodologies to identify specifications of a general and high-fidelity simulation methodology, followed by designing and developing such methodology. Then, the simulation methodology is used as a building block of a framework for evaluating different sensor configurations based on their ability to infer contextual information from synthetic raw sensor readings.

Chapter 3

Taxonomy Specification of Indoor Space Simulation Methodologies

The SIS simulation methodologies provide fast and cheap-to-evaluate frameworks for quickly analyzing SIS applications' performance before the actual deployment. Although the tools have been used extensively, no unified taxonomy outlines the fundamental components of the tools. This chapter survey related works comprising SIS simulation methodologies and obtain a unified taxonomy accordingly. The taxonomy demonstrates the components that simulation tools model and describes the modelling approaches found in the literature for each component. It also specifies the domain of the tools, *i.e.* building and application types, as well as the typical evaluation methodologies. Such a taxonomy can be a standard for developing future SIS simulation tools. We utilize the taxonomy to analyze the literature systematically. We give tags to each paper based on the taxonomy definitions and apply hierarchical clustering to group the papers. Our bibliometric results show that the taxonomy can divide papers into meaningful groups.

The following sections describe our survey methodology, followed by the taxonomy definition. Next, each taxonomy component is described in detail with examples from the literature. Finally, we explain our bibliometric method and present its results.

This chapter conducts a comprehensive literature review to establish the foundational knowledge necessary to answer R3 research question, specified in Section 1.2:

A SIS simulation methodology that can simulate various applications and produce high-fidelity synthetic datasets.

3.1 Survey Methodology

We use a snowballing procedure [151] for the identification and collection of studies within the scope of the survey. Our research scope is studies that focus on the simulation of smart, *i.e.* sensorized, indoor spaces sensitive to occupants' movement-and-activity traces. Therefore, the collected publications are considered relevant if they put forward (a) modelling of occupants and (b) modelling of sensor behaviours. Our inclusion/exclusion criteria are as follows: non-peer-reviewed papers were excluded; among the papers that cover the same study, only the most recent study was included; and out of our research scope (mentioned above) papers were excluded. We describe our application of the snowballing procedure in the following steps.

3.1.1 Snowballing Procedure

Step 1: Establishing the Initial Set We utilize the combination of the following keywords on Google Scholar (the actual search was conducted in February 2021): *smart home simulation, smart home simulator, smart indoor space simulation, smart indoor space simulator, intelligent environments simulation, intelligent environments simulator*. In the start set selection process, we consider papers from different publishers, communities, and publication years. Accordingly, we choose the list of papers in Table. 3.1.

Step 2: Iterations For each of the papers in the initial set, we apply two types of iterations: backward and forward snowballing. In the backward snowballing, we study the paper's references and include papers based on our inclusion and exclusion criteria (mentioned in section 3.1). In the forward snowballing, we study the papers citing the paper and include the ones based on our inclusion and exclusion criteria.

Table 3.1: Selected papers for the snowballing procedure's initial set.

Authors	**Title**
I. Armac and D. Retkowitz [8]—2007	Simulation of smart environments
J. Park et al. [111]—2007	Cass: A context-aware simulation system for smart home
J. C. Augusto and M. J. Hornos [11]—2013	Software simulation and verification to increase the reliability of intelligent environments
K. Bouchard et al. [21]—2010	Simact: a 3d open source smart home simulator for activity recognition
K. McGlinn et al. [90]—2014	Simcon: A context simulator for supporting evaluation of smart building applications when faced with uncertainty
J. Synnott et al. [137]—2014	The creation of simulated activity datasets using a graphical intelligent environment simulation tool
J. W. Lee et al. [78]—2015	Persim 3d: Context-driven simulation and modeling of human activities in smart spaces
A. Vasilateanu et al. [144]—2016	Smart home simulation system
O. Kamara-Esteban et al. [69]—2017	Massha: an agent-based approach for human activity simulation in intelligent environments
N. Alshammari et al. [4]—2017	Openshs: Open smart home simulator
J. Renoux and F. Klugl [117]—2018	Simulating daily activities in a smart home for dataset generation
S. Golestan et al. [54]—2020	Towards a simulation framework for smart indoor spaces

We repeat the forward and backward snowballing steps on the selected papers in the previous iteration to identify new papers until no new papers are found.

3.1.2 Bibliometric Overview

This section analyzes the studies included in the snowballing process described in the methodology section. Following the inclusion and exclusion criteria in our methodology (mentioned in section 3.1), we choose 64 studies (from 213 unique authors) whose distribution over the years is shown in Fig. 3.1. In this methodology, it is impossible to statistically validate the outcome of the process because there is no ground truth available. There is no way to establish "the correct" set of publications to include.

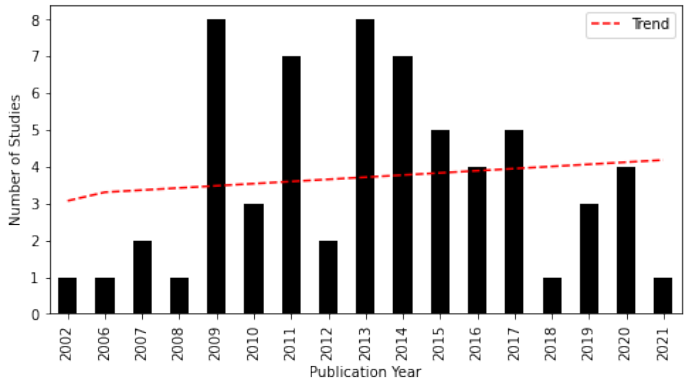


Figure 3.1: Histogram of studies included in our survey throughout the years.

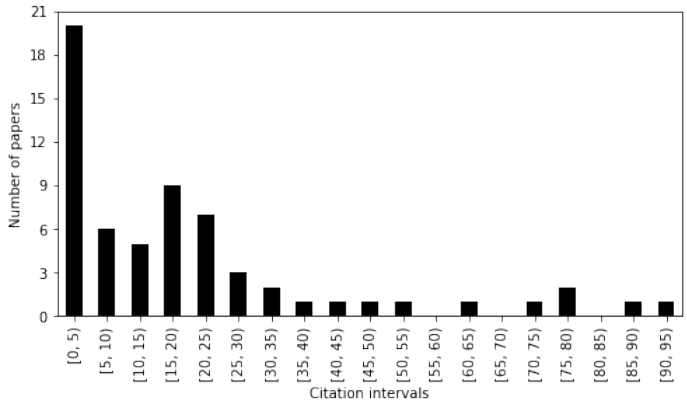


Figure 3.2: Histogram of the studies in terms of different citation counts.

Although we are not claiming that our literature review method collects all the studies in the literature (which is indeed impossible for any type of literature-review methodology), we argue that the collection of 64 studies from different research groups is a fair sample of studies in the field to be used in this paper. In addition, it should be noted that studies before 2002 do not align with more recent studies, due to the advancement of technologies (both hardware and software) such as sensors, actuators, simulation engines, and computing power. The increasing trend shown in Fig. 3.1 indicates more interest in using simulation tools for designing and prototyping SIS applications. Fig. 3.2 also shows the histogram of bins of studies where each bin corresponds to a different range of citation numbers. The figure shows that most of the studies have less than five citations which, combined with the growing trend observed

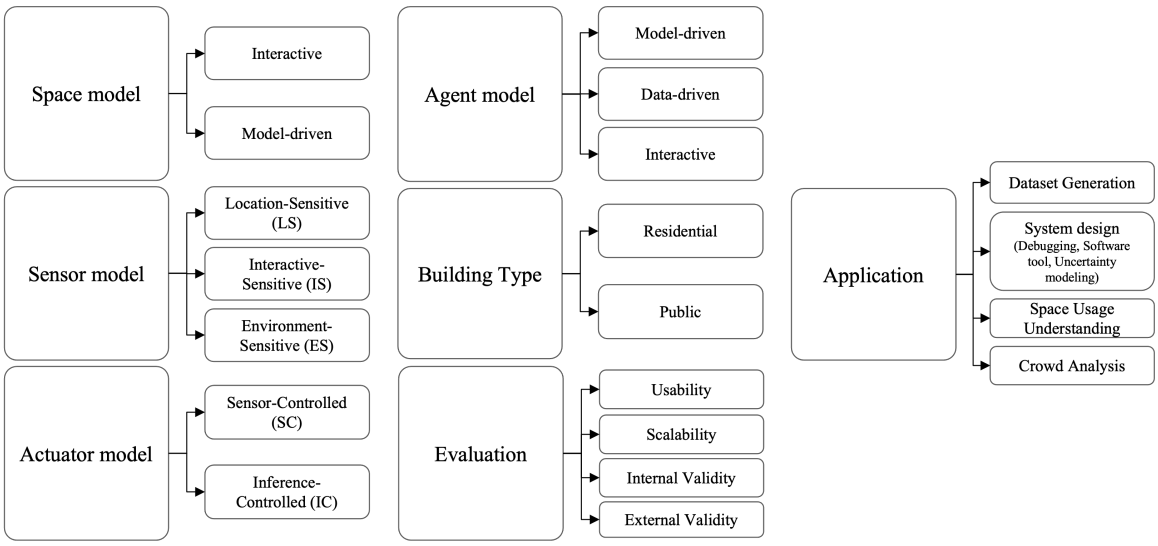


Figure 3.3: Taxonomy of SIS simulation methodologies.

in Fig. 3.1, the field of SIS simulation is new.

3.1.3 Key Features of SIS Simulation Methodologies

SIS simulation methodologies model the space, sensors, actuators and occupants (agents), and they are domain-specific, *i.e.* building type and intended application. A taxonomy of SIS simulation methodologies is shown in Fig. 3.3. We detail each element of the taxonomy in the following sections.

Space Model

The space model defines physical world specifications, including rooms, furniture, obstacles, etc. It also contains objects virtual agents can interact with, such as a bed, couch, and television. There are two approaches for space modelling in the literature, interactive and model driven. *Interactive* space modelling focuses on user interaction with simulation software. Users must design the intended indoor space layout and then place furniture and obstacles using a predefined list of popular before-mentioned objects in the software. For example, Ho et al. [61] provided a user interface to select and place prefab instances of structural and home objects. *Model-driven* space modelling utilizes data models to provide space specifications, and standardize their

relations, properties, and data exchange. Recently, studies like [54, 90] utilize Building Information Modelling (BIM), which is an industry-standard digital representation of buildings [13]. BIM can be represented in the Industry Foundation Classes (IFC) format. The IFC data model is an open specification and an International Standard ISO 16739-1:2018 for describing architectural, building and construction industry data.

The building model depends not only on the range of applications each simulator is envisioned to support but also on the specific space modelling approach used. Interactive, manual, space modelling is impractical for modelling existing buildings and is inevitable when there is a paucity of CAD, or preferably, BIM models. While the situation is rapidly improving, calling for a legacy building model to be built from scratch instead of ingested from an existing data source, limits the applicability of a tool. Even if the overall space model is ingested, further steps of introducing prefab objects, such as furnishings, can be challenging for a manual process, especially when handling large legacy buildings.

Sensor Model

The sensor model defines specifications, such as type, sensing area, and location, of the deployed sensors in space. Based on the agents' presence and interaction, the sensor model generates synthetic sensor events accordingly. There are three categories of sensors in the literature. To begin with, the *Location Sensitive* (LS) sensor type denotes the types of sensors that trigger based on measurements caused by agent(s) physical presence inside their sensing area. Examples of LS sensors are passive infrared (PIR) motion sensors, an RFID. Next, the *Interaction Sensitive* (IS) sensor type is a type of sensor whose state changes depend directly or indirectly on the agents' interaction with the space, such as door sensors. Examples of IS sensors are pressure sensors, switches and smart objects (*i.e.* report their on/off states). Finally, the *Environment Sensitive* (ES) sensor type denotes the sensors that measure

environmental parameters within a specific sensing area. Examples of ES sensors are temperature and humidity sensors.

Actuator Model

The actuator model consists of the specification, such as type, location, and actuating behaviour of the deployed actuators in the space. They mainly change the state of a particular object, *e.g.* closing/opening a window or adjusting environmental parameters such as temperature. Actuators' operations depend directly or indirectly on the sensor readings. Accordingly, there are two types of actuators in the literature. First, the *Sensor-Controlled* (SC) actuators operate based on the decisions directly derived from sensor readings. Most SC actuators employ rule-based methods for the decision-making process. For instance, an actuator turns off the heater when the indoor temperature is above 23 Celsius. Secondly, the *Inference-Controlled* (IC) actuators operate based on the outcome of an inference algorithm that ingests the sensor readings. For instance, an actuator changes the ambient light based on the current estimated activity of occupants.

Agent Model

The agent model specifies the behavioural characteristics of virtual agents and is often application dependent. Based on the agent model, agents interact with the indoor space to perform activities. Borrowing from Synnott et al. [138], we define three agent modelling approaches. To begin with, the *interactive* agent modelling focuses on user interaction with simulation software. Simulators provide a user interface to control the virtual agents, *i.e.* moving around and performing activities. For example, the OpenSHS simulator by Alshammari et al. [4] offers a first-person user interface to users for interactive agent modelling. Another approach is the *model-driven*, which denotes the specification of activities such as names, duration, affordance, or prerequisites, and a policy for the virtual agent. The agent's policy prioritizes the

set of activities and actions that virtual agents need to perform and varies from motivation-driven [88], hierarchy-based [117, 69], agenda-based [22]. Among them, the hierarchy-based approach is shown that can be adjusted to fit the intended context, *e.g.* performing Activities of Daily Living (ADL) or office routines [69]. Finally, the *data-driven* approach uses real-world agent activity traces as training data for generative models such as Markov-chain-based models in [37, 60]. An example is the simulator by Veronese et al. [146] in which the authors proposed a motivation-driven agent modelling approach with several parameters, such as the likelihood of performing actions given time. They used real-world datasets to tune corresponding parameters.

Application Area

The application area denotes the focus area of the simulators. Space, agent, sensor, actuator models, and building type decide the range of SIS applications that simulators support.

Most simulators can be used as synthetic *dataset generators*. The main intuition of synthetic dataset generation is that having access to sensory datasets is necessary for activity recognition tasks in indoor spaces. The target population of the activity recognition tasks are primarily people with special needs, *e.g.* people with dementia, physical disability, etc. Putting aside the real-world implementation challenges, even collecting such datasets is complicated by limitations posed by ethics protocols and by the end-user's willingness to participate or commit to a study. For example, Casaccia et al. [31] proposed a simulator to generate trajectory data associated with activities of daily living. They showed that the data could be used in a classification task to detect wandering versus typical trajectories. There are other studies whose intuition of dataset generation is to utilize the dataset for evaluating different sensor deployment configurations. Such simulators provide metrics to analyze the dataset accordingly. Golestan et al. [55] presented metrics based on Quality of Context (QoC)

for the Internet of Things (IoT) applications.

Another frequent application for simulators is that of *system design*. Designers can use emulators to quickly prototype SIS applications and systematically test and debug their intended system. There are several focus areas in this application type. To begin with, several simulators proposed debugging SIS applications regarding actuators' proper functionality and consistency based on sensor readings such as [101, 105]. Designers can repeat the simulation of specific scenarios without worrying about real-world environment noise and uncertainty to identify faulty behaviours of actuators. Among these studies, several works such as [70, 132] incorporate the functionality of both actual and virtual devices, *i.e.* emulation. This functionality is suitable for debugging already-existing systems. The designers can improve the controllability of the systems by using virtual devices for the parts of the systems that are understandably working correctly. As an example, Stahl et al. [132] proposed the idea of dual reality, which considers the synchronization of the real and virtual worlds. Another system design focus is to use software development tools for designing SIS applications. Such simulators adapt software design approaches for managing the design process. For example, the simulator proposed in [17] offers a domain-specific design language for SIS applications. Finally, some studies focus on uncertainty modelling, introducing a controlled dosage of uncertainty to one or more components of SIS applications, such as probabilistic sensor modelling [90, 54], or unexpected agent behaviour [123].

A few studies focus on *space usage understanding* applications. These applications are suitable for public buildings such as hospitals and libraries where there is no clear usage definition of spaces before observing how occupants use those areas. For example, Schaumann et al. [123] illustrated that a hospital corridor could be used as either a waiting space or a meeting space by defining semantics for particular areas based on the agent's activities in the areas throughout time.

Simulators focusing on *crowd analysis*, investigate the behaviour of a crowd in public buildings given a scenario, such as hazardous situations. These simulators

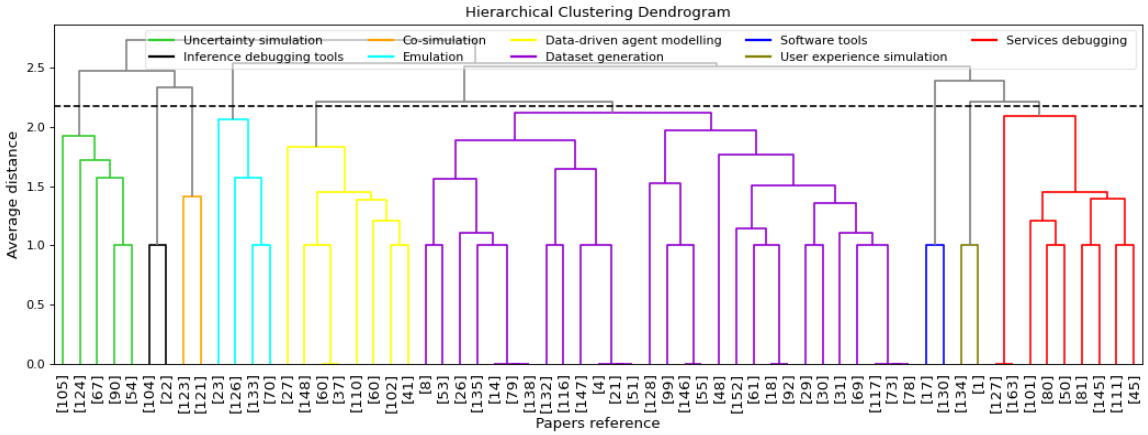


Figure 3.4: Dendrogram of the studies. Each color shows a separate cluster.

usually have coarse-grained agent model granularity, *i.e.* navigating to the nearest exit doors in an emergency. These studies suggest methods for handling large-scale applications in terms of computational resources, e.g., using multi-threading in [122].

Evaluation

Simulators are usually evaluated in terms of usability, scalability, and internal or external validity. *Usability* evaluation consists of measurements that determine how well a specific group of users (SIS designers in general) can use a simulator. The measurements could be a questionnaire such as [4, 127, 125], or measuring application design completion time by software engineer students like [22]. More sophisticated usability studies can be found in [90, 27]. For example, in the study by Burghardt et al. [27], the authors observed user interactions and identified issues associated with the SIS design process and proposed possible fixes. The *scalability* refers to measuring the required time to execute specific scenarios while increasing the simulation speed [22] or the size of SIS applications, in terms of devices [79], agents [120], or rules (services) [70, 101]. Internal validity refers to examining the capability of the simulators in terms of specific application requirements satisfaction and related dataset generation. For instance, Casaccia et al. [31] investigated if their simulator can produce agents with wandering patterns of behaviour, based on definitions by Martino Saltzman [87]. On

the other hand, external validity determines how realistic simulators can produce synthetic datasets compared to real-world datasets. For example, Golestan et al. [54] investigated, in a case study, if their methodology can produce a realistic dataset regarding agent traces and sensor readings.

3.1.4 Publication Clustering

To analyze the collected papers, we first apply hierarchical clustering to systematically group the studies based on their similarity in terms of taxonomies described in Fig. 3.3 as features. We exclude the evaluation facet because we observed that the type of evaluation depends on the development stage of the studies. In total, each paper corresponds to a vector of 16 binary features. Fig. 3.4 shows the dendrogram obtained from the clustering algorithm. The clusters are merged based on their average distance in the feature space. The horizontal dashed line shows a cut-off point (roughly equal to 2.1) introducing nine clusters. We name each cluster based on the papers that fall into the cluster. Table. 3.2 shows cluster names, their short descriptions, size, and the related papers' citations.

3.2 In-Depth Review of the Field

The first cluster, *uncertainty simulation*, includes papers that emphasize the stochastic parameters of SIS applications. Together with two other small-sized clusters, this cluster does not merge with the rest of the papers until the last step; therefore, they have the largest distance from the rest of the papers in the literature.

Fig. 3.4 shows that there are four binary clusters, *inference debugging tools*, *co-simulation*, *software tools*, and *user experience simulation*. These clusters focus, respectively, on debugging applications with IC actuators, using the co-simulation paradigm for handling large-scale SIS applications, adapting software engineering tools for managing SIS application development, and using simulation to evaluate user experience designs. The papers of these clusters explore atypical research areas

Table 3.2: Clusters information obtained in our literature review.

Name	Description	Size	Citations
Uncertainty simulation	Includes papers that model unpredictable events in agent and/or sensor models	5	[105, 90, 123, 54, 67]
Inference debugging tools	Includes papers that support quickly prototyping and debugging inference services	2	[22, 104]
Co-simulation	Includes papers that present scalable co-simulation solutions for large-scale crowd modelling	2	[120, 122]
Emulation	Includes papers that present emulation tools to support coexistence of real and virtual devices and/or easier transitions between virtual and real worlds	4	[125, 70, 23, 132]
Data-driven agent modelling	Includes papers that focus on modelling indoor space occupants behaviours using real-world datasets	9	[146, 60, 37, 41, 102, 91, 27, 11, 110]
Dataset generation	Includes papers that present simulation tools to generate datasets for developing indoor activity recognition algorithms	30	[18, 73, 79, 137, 78, 144, 117, 55, 31, 14, 21, 48, 51, 116, 4, 127, 145, 92, 99, 69, 29, 134, 26, 30, 131, 61, 8, 150, 53]
Software tools	Includes papers that adapt software engineering modelling languages and tools for more systematic SIS applications design	2	[129, 17]
User experience simulation	Includes papers that support quickly designing and testing user experience designs	2	[133, 1]
Services debugging	Includes papers that focus on debugging services containing rule-based actuator models	9	[111, 45, 50, 81, 101, 161, 80, 126, 143]

of SIS simulation tools. That is why these small clusters have large distances from other papers.

The largest cluster, *dataset generation*, contains papers offering the most typical usage of SIS simulation tools, *i.e.* generating synthetic datasets. The *data-driven agent modelling* is the closest cluster to the dataset generation cluster because it

contains papers whose goal is also to generate synthetic datasets with agent models constructed using real-world data.

The *services debugging* cluster includes papers that allow quickly designing SIS applications by incorporating the possibility of debugging SC actuators and reconfiguration of sensors and actuators accordingly. The closest clusters are software tools and user experience simulation, which are also concerned with designing SIS applications and reconfiguring the devices used.

Finally, *emulation* cluster contains papers that use a combination of real and virtual devices or offer a smooth transition to real-world setup after designing the system in a simulation.

In the following, we provide an overview of each cluster, summarize their papers, and provide insights.

3.2.1 Uncertainty Simulation

The uncertainty simulation cluster contains studies that assume SIS applications consist of both stochastic and deterministic events. Their objective is to model unpredictable System Under Test (SUT) behaviours as modelling either or both stochastic sensors and agents behaviours. The evaluation of these studies includes external-validity investigation [54], internal-validity [105], and usability [90].

Taxonomy specifications of this cluster are shown in Table 3.3. Most studies model uncertainty regarding agent behaviours [54, 105, 67, 123]. Their agent models include probabilistic methods that consider several parameters such as time and previous activities, and select the next activity to perform. The [54] and [90] simulators model probabilistic sensor modelling. The modelling approach assigns a triggering probability to each sensor considering its distance to the event of interest.

Table 3.3: Taxonomy specifications of the papers in the uncertainty simulation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
O'Neill et al. [105] - 2013	Model-driven (class hierarchy representation)	IS (RFID), LS (motion sensor)	–	Model-driven (sequential)	Public	System design	Internal Validity
McGlinn et al. [90] - 2014	Model-driven (BIM via IFC)	LS (presence, proximity and occupancy sensors), ES (temperature sensor)	–	Interactive	Public and residential	System design	Usability
Schaumann et al. [123] - 2015	Model-driven (Bim via IFC)	LS (occupancy sensor)	–	Model-driven (hierarchy-based with randomness)	Public	Space usage understanding	–
Golestan et al. [54] - 2020	Model-driven (BIM via IFC)	LS (motion and beacon sensors)	–	Model-driven (hierarchy based with randomness)	Residential	System design	Agent and sensor models external validity
Jiang and Mita [67] - 2021	Model-driven (topology-driven with user-defined requirements and predefined rules)	LS (motion sensor)	–	Model-driven (motivation-driven)	Public and residential	System design	–

Papers Summary

O'Neill et al. [105] argued that there are two key challenges faced by pervasive systems designers in the testing phase: 1) difficulty of monitoring deployments and 2) tracing the physical and digital causes of unwanted situations. Their simulator, InSitu, models situation specifications as a temporal definition regarding users, entities, zones, and their attributes. In a case study, they defined six situations and showed that their simulation detects the situation instances. However, the discrete-time intervals used by the simulation run, in principle, the risk of undetected situations.

McGlinn et al. [90] identified uncertainty modelling as one of the requirements that smart indoor space simulators should consider. They stated that the modelling

system's uncertainties are crucial because they could affect the design decisions. Every sensor reading can have a level of uncertainty. The uncertainty is captured by an offset following a Gaussian distribution with variance equal to a given parameter and mean equal to the sensor's location. The final sensor reading gets scrambled by replacing sensor outputs with values from the distribution. The paper's evaluation addressed the level of usability of the simulator. First, the configuration, simulation, and visualization of two applications by subjects were studied using a questionnaire to obtain the system usability scale and level of efficiency. The questionnaire also measures the effect of visualizing uncertainties compared to not having them. The results demonstrate that using SimCon visualization tools is promising in reducing time spent on design: by visualizing the level of uncertainty, design time is reduced by a factor of three compared to not visualizing uncertainty.

Schaumann et al. [123] presented a simulator with realistic human behaviour to analyze space usage understanding. They argued that human behaviours are divided into planned or unplanned activities. Stochastic human behaviour (in collaboration with other humans) generates contexts within different locations. Analyzing the contexts and corresponding locations leads to understanding usages of locations. Therefore, it will be possible to observe if buildings support end-user activities. They defined events as a combination of three types of information: actors (agents), activity, and space. Unplanned events, which are events with collaboration with other human behaviours, are a list of behaviours that may be performed if certain conditions are met. Sections of the space are labeled according to the type of behaviour being performed by the agents. The authors argued that the simulator offers more flexible modelling of agents.

Golestan et al. [54] proposed a simulation methodology for evaluation of different sensor deployment configurations. They argued that the simulator needs to produce realistic sensor events as well as realistic agent traces. Therefore, they injected a controlled level of uncertainty in both agent and sensor models. In terms of agent

modelling, they used a hierarchy-based approach with randomness in generating a path between the current and a destination location. For sensor modelling, the authors used a similar approach to [90]. For each sensor, they defined a probability proportional to the agent's location and the sensor's location, *i.e.* the sensor gets triggered with a higher probability as the agent is closer to the sensor and vice versa. In a case study, they found that the simulator realistically models human behaviours and sensors. However, they found that modeling certain sensors, such as BLE beacon sensors, is challenging and depends on the exact characteristics of the environment, e.g., how it impacts wireless propagation.

Jiang and Mita [67] proposed a simulator that generates a diverse set of simulation scenarios for an older individual living alone. The structure of their simulation methodology consists of a graphical user interface in which users determine high level parameters for space, and agent models, e.g. functional zones and individual characteristics respectively. Given the high level information by users, the simulator generates an indoor space and an activity schedule, specifically, a sequence of activities based on a possibility-based motivation-driven approach.

Insights

The proposed probabilistic agent modelling approaches mainly require adjusting several hyperparameters to drastically change the agent behaviours. Therefore, these approaches are required to be externally valid in at least some test cases. Nevertheless, only the paper by Golestan et al. [54] attempted to evaluate the external validity of their agent model.

In terms of faulty sensor behaviours, in [90], it was defined as a level of uncertainty for each sensor. Designers are required to manually adjust the uncertainty level in the design process. Although burdensome and time-consuming, this approach allows designers to adjust the parameter based on environmental factors. For example, sensors may be close to a source of interference, leading to faulty readings or occasional

inability to communicate their data. In [54], false negative sensor readings were modelled by increasing the probability of a motion sensor getting triggered the closer an occupant is to the sensor even if, technically, they are already within a broad sensor coverage range. However, their approach does not consider other factors, such as those limiting the wireless propagation which is crucial in RF-based beacon sensors.

Coping with uncertainty due to imperfect sensing is also addressed in [105] but by attributing such imperfections on flawed sensor deployment. Taken together, [54], [90], and [105], offer a comprehensive, complementary view of ways uncertainty can be modeled at the sensory level: from agent behavior, to environment, to placement of sensors.

3.2.2 Inference Debugging Tools

The inference debugging tools cluster incorporates IC actuator models and provide simulation-based prototyping frameworks to quickly develop and test the behaviour of actuators given sensor readings. The evaluation methods in this cluster are usability and scalability analysis.

Taxonomy specifications for this cluster is shown in Table 3.4. Both papers model IC actuators based on occupants' location predictions. The actuators use the predictions and accordingly change the state of specific objects in the space, *e.g.* turn lights on/off or show helpful information on screens. As it is stated in [104], several technical and non-technical factors affect the performance of the designed prototype. These factors make it difficult to measure the prototype's performance objectively; therefore, it is left to the end-users opinion.

Papers Summary

O'Neill et al. [104] argued that effectively identifying unwanted behaviours in contextual systems, and debugging such systems is challenging. The authors proposed a simulation-based user-centric testing of adaptive context-aware systems. Based on

Table 3.4: Taxonomy specifications of the papers in the inference debugging tools cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
O'Neill et al. [104] - 2009	Interactive	LS (motion sensor)	IC (light switch)	Model-driven (sequential)	Public	System design	Usability
Bruneau et al. [22] - 2013	Interactive	LS (motion sensor), ES (temperature sensor)	IC (light switch)	Model-driven (sequential)	Public	System design	Usability and scalability

the application, designers need to define a set of requirements (*e.g.* in a smart lighting system application, end-users should not stay in the dark). The simulator continuously monitors the state of the application and reports alerts if the defined requirements are violated. Their case study reports on the structure of the simulated building, which is a three-storey office building with 104 rooms furnished with 520 desks, 352 chairs, and 257 desktop computers. The paper does not report any metrics about simulated devices, or agents, or performance. Their simulator was evaluated in terms of efficiency (designers effort), reliability (how often the simulator has false positives in detecting the prototype's flaws), and repeatability (the ability to reproduce unwanted situations for further investigations).

Bruneau et al. [22] presented DiaSim. DiaSim's input is the intended application's description as a set of parameters and generates simulation logics and executes the application for visual debugging. Four requirements are defined for their simulator based on embedded systems simulators: area-specific simulator, transparent simulation, testing a wide range of scenarios, and simulation renderer. The architecture of their simulation methodology consists of three taxonomies: contexts, controllers (such as room occupancy detection and heat regulator), and entities (such as sensors or other sources of information like calendar). They indicated that the simulator can be used to visually monitor and debug pervasive computing applications. In order to

Table 3.5: Taxonomy specifications of the papers in the co-simulation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Sanmugalingam and Coulouris [122] - 2009	Interactive	IS (smart objects), LS (active bat)	–	Model-driven (sequential)	Public	Crowd Analysis	Internal validity, Scalability
Sánchez et al. [120] - 2017	Interactive	LS (inertial sensors)	–	Model-driven (hierarchy-based)	Public	Crowd Analysis	Scalability

show the scalability of their simulator, they modelled an engineering school. They studied CPU usage with respect to the simulation speed in two phases: low activity rate (*e.g.* nights) and high activity rate (*e.g.* during the breaks). They found that slower simulation speed uses less than 20% of the particular CPU. But increasing the simulation speed significantly increased CPU usage.

Insights

Both papers indicated that the simulators could be used for visually monitoring and debugging. Automatic evaluation of designed prototypes was approached in [104], by proposing the alerting system. Nevertheless, the situations that give rise to such alerts need to be traced for further manual analysis. In any case, both approaches are not straightforward for most real-world situations, especially when large-scale applications are considered. It would be interesting to investigate the scalability of their evaluation method in order to specify the type of SIS applications that their simulators support.

3.2.3 Co-simulation

The co-simulation cluster contains papers that offer large-scale co-simulation solutions for crowd modelling, *i.e.* scenarios with several agents. The evaluation methods in this cluster are scalability and internal validity analysis.

Table 3.5 shows the taxonomy specifications of this cluster. The space modelling method used in this cluster is interactive, which is cumbersome when applied to existing large scale buildings. Therefore, the scalability evaluations of the simulators are focused on the sensor and agent modelling. The work in [122] concentrates on techniques for developing a scalable simulator; whereas in [120], simulator paradigms were classified and guidelines were proposed for modelling simulators based on the user's requirements.

Papers Summary

Sanmugalingam and Coulouris [122] proposed a simulator to address the need for simulating environments for testing large-scale pervasive computing applications. The requirements they identified for the simulator are genericity (generic programming) and pluggability (pluggable statistical distributions to allow modelling different devices' behaviours and pluggable equations to model different locatables such as occupants). These two requirements offer to model different infrastructures for different types of applications. The simulator has a state transition controller with a global clock which manages entities and their relations, agents, tasks, sensor model, etc. For scalability purposes, the simulator executes each agent on separate threads. In a case study, the authors found that in an airport (529,547 m^2), the simulator can add as many as 2.5 agents per square meter on average. In this case, the average movement speed of the agents reduces to 0.2 (m/s).

Sánchez et al. [120] argued that combining and synchronizing various domains that coexist and evolve in simulation is challenging since including all the domains into a single simulation causes simulators to be slower. The authors presented a co-simulation methodology which contains four steps. For each step, the authors provided guidelines that help designers choose a suitable simulation configuration based on their requirements. The steps of the proposed methodology are 1) selection of the co-simulation paradigm, 2) particularization of the general simulation model and

simulation lifecycle, 3) selection of the appropriate coordination mechanism and 4) design of the user interface and results presentation. The authors performed three types of experiments. First, they evaluated a crowd simulation by asking an expert about different quality parameters such as usability, scalability, or the ability to customize the simulation. The second and third experiments involve measuring scalability precisely. The authors measured simulation time in the second experiment while increasing the number of agents. In the third experiment, the authors obtained the probability of successful simulation execution while increasing operation parameters such as the number of agents. The authors claim that their methodology produces scalable simulators concerning the number of simulated devices and agents. Specifically, their methodology can model 2000 agents, which requires around 10^5 (s) of simulation time on average.

Insights

The methodologies of the papers considered flexibility in defining and deploying different devices. However, only the evaluation criterion in [120] included the flexibility analysis. Their results show that designers using their simulation methodology will likely find new devices easier than baseline methodologies. Nevertheless, further analysis is required to show the strength of this observation.

The simulator's interactive approach for space modelling is cumbersome for large existing, often public, buildings. Furthermore, as noted in [120], the detailed geometry specifications of the physical space become important for specific applications such as crowd management. Therefore, model-driven approaches are needed for accurately modelling the space.

3.2.4 Emulation

The emulation cluster incorporates papers that aim to present simulation tools that are either hybrid (using a combination of real and virtual worlds or modelling physical

processes) or provide an easier transition between real and virtual worlds. Several papers attempted to present emulation-like tools so that virtual and real-world devices could communicate (for example [70]), while other papers use hybrid frameworks that introduce human-computer interaction methods which require users to interact with real-world elements in Virtual/Augmented Reality (VR/AR); for example, by using a maquette and Augmented Reality (AR) to create space models in [125]. In principle, there are two primary motivations for using AR/VR technology in SIS applications. Firstly, it allows studying the users' activities as they interact with the application through physiological measures [131] and customizing the applications accordingly. Secondly, it affords a more realistic user-interaction modality, with users moving towards and "touching" objects, and thus a more intuitive understanding of the pre-deployment applications [125]. Therefore, using VR/AR is more related to studying the end users' experience with SIS applications than analyzing the technical infrastructure behaviours, which is the focus of our literature review.

Table 3.6 shows the taxonomy specifications of this cluster. As the table shows, most of the papers [132, 23, 125] used temperature sensors. Thus, they modelled physical processes, specifically indoor temperature changes.

Papers Summary

Kang et al. [70] presented a simulator with the ability to add and modify entities (sensors and actuators) easily. They introduced the concept of a widget, which encapsulates the definition of sensors, actuators and services. The architecture of the simulation is based on the unified context-aware application model in a ubiquitous computing environment (Ubi-UCAM) [66]. Therefore, widgets can communicate with real-world services and sensors. Their results prove that the simulator is scalable concerning the number of services, i.e., rules that link sensors to actuators to simulate the sense-control process. Specifically, the authors found that the simulation time does not change significantly as the number of services increases from 1 to 6.

Table 3.6: Taxonomy specifications of the papers in the emulation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Kang et al. [70] - 2009	Interactive	ES (particle sensor)	SC	model-driven (sequential)	Residential	System design	Scalability
Stahl et al. [132] - 2011	Interactive	ES (temperature sensor), IS (RFID)	–	Interactive	Residential	System design	–
Bruneau et al. [23] - 2012	Interactive	ES (temperature sensor)	SC	Model-driven (sequential)	Residential	System design	Internal validity
Seo et al. [125] - 2016	Interactive	ES (temperature sensor)	–	Interactive	Residential	System design	Usability

Stahl et al. [132] focused on evaluating currently deployed SIS systems regarding end-user usability. The authors used the idea of *dual reality* to apply modifications to the currently deployed systems. The dual reality concept handles the mutual influence between real and virtual environments while synchronized, i.e., changes to either of the environments also change the other. After the design phase, Universal Remote Console (URC) standard and Universal Control Hub (HCH) were used to implement the Dual Reality concept. The authors argued that their system supports modelling applications like controlling local and remote places, remote caregiving, and debugging SIS systems.

Bruneau et al. [23] proposed a framework for virtually testing SIS applications. For modelling devices, they used DiaSpec, a domain-specific language for networked entities such as sensors and actuators. They also modelled a building's heat transfer and temperature changes, relative humidity and carbon dioxide density. The system can be deployed in DiaSim, a 2D simulator for DiaSpec for virtually testing different approaches in terms of energy usage management. The authors demonstrated the usage of their simulator by testing several algorithms to find the most efficient one

for regulating HVAC control systems.

Su et al. [125] proposed a hybrid-reality methodology for designing and evaluating SIS systems. The proposed methodology incorporates a first-person perspective in Virtual Reality (VR), wherein users can navigate indoor spaces and interact with virtual objects. It also has a third-person perspective in Augmented Reality (AR), wherein users can design SIS systems through an interface (*e.g.* their smartphone). The latter requires a miniature version of the intended space. The authors found that their methodology shows a promising user experience. However, they found no significant differences between AR and VR experiences from the questionnaires given to their subjects. In addition, they found that although VR is more immersive than AR, it is more stressful because users feel uncomfortable wearing a VR headset. Finally, in terms of interacting with objects, it is found that gestures in VR might be frustrating because of possible errors.

Insights

The simulators presented in [132] and [125] both offer methodologies to improve the usability of the SIS systems designing process in both real and virtual worlds. Their approaches are suitable for defining and satisfying end-user requirements in the design process because designers and end-users can effectively interact. Nevertheless, their methodology might be time-consuming and burdensome, especially for the end-users. One could also argue that some frustrations with the use of VR will be ironed out as the technology evolves and that they are not specific to SIS.

The simulators presented in [70] and [23] encapsulate smart indoor space objects and use a domain-specific language for two main reasons: smoother transition from simulation to real-world and vice-versa, and interoperability of virtual and real devices. However, none of the studies evaluated the similarity of the simulation performance compared to the real-world implementations in terms of sensors, actuators, and services.

3.2.5 Data-Driven Agent Modelling

The data-driven agent modelling cluster contains papers that aim to utilize real-world occupants' trace datasets to train generative models, mainly Markov Chains, for modelling agent behaviours. The main evaluation criterion of these papers is the external-validity of their method [37, 60, 91, 146, 41, 102], while internal-validity [110] and usability [27] are occasionally used as well.

Table 3.7 shows the taxonomy specifications of this cluster. The table shows that less attention has been paid to space modelling because few studies [27, 102, 110, 91] utilized an interactive space modelling approach, and other studies [37, 60, 146, 41] assumed that the space model is given. In addition, most of the studies focused on residential building applications since real-world occupants' dataset in such buildings is more straightforward to collect and more available than in public buildings. The only study focusing on public building applications [27] limited the indoor space to a meeting room.

Papers Summary

Burghardt et al. [27] proposed a tool for assisting designers in developing user behaviour detection algorithms. Two steps are proposed in their framework. In the first step, a GUI is presented to facilitate a Hidden Markov Model (HMM) definition, such as information about activities like their names, duration, location, and involved devices. Then the HMM parameters will be constructed based on the information provided. The second step provides a simulation environment to navigate virtual agents and perform activities manually. The corresponding agent's trace is fed to the trained HMM in the first step. The differences between the information provided in the first step and the HMM output in the second step are suitable for debugging indoor space applications.

Dahmen et al. [37] presented a simulation that uses a machine learning-based synthetic dataset generation methodology. The methodology uses real-world data

Table 3.7: Taxonomy specifications of the papers in the data-driven agent modelling cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Burghardt et al. [27]—2009	Interactive	IS (RFID)	–	Data-driven (HMM trained on real data)	Public	System design	Usability
Dehman et al. [37]—2019	N/A	IS (smart objects)	–	Data-driven (HMM trained on real data)	Residential	Dataset generation	External validity
Helal et al. [60]—2009	N/A	IS (smart objects)	–	Data-driven (HMM trained on real data)	Residential	Dataset generation	External validity
Veronese et al. [146]—2015	N/A	IS (smart objects)	–	Data-driven (a model based on agents needs, *e.g.* hunger, tiredness, etc.)	Residential	Dataset generation	External validity
Elbayoudi et al. [41] - 2015	N/A	LS (motion sensor), IS (pressure and door sensors)	–	Data-driven (HMM trained on real data)	Residential	Dataset generation	External validity
Noury et al. [102] - 2012	Interactive	LS (motion sensor)	–	Data-driven (HMM trained on real data)	Residential	Dataset generation	External validity
Papamakarios et al. [110] - 2014	Interactive	LS (Kinect camera)	–	Data-driven (HMM trained on real data)	Residential	Dataset generation	Internal validity
Mendez et al. [91] - 2009	Interactive	IS (smart objects), LS (pressure sensor)	–	Data-driven (HMM trained on real data)	Residential	Dataset generation	External validity

and generates agent traces and respective sensor values using the Hidden Markov and Regression models. First, the HMM generates activity sequences, and each activity has an HMM to generate a corresponding sensor event. Then, their methodology uses the regression models for each activity to generate a duration of sensor events. The authors defined a distance measure based on Dynamic Time Warping (DTW) as their evaluation metric. They showed that their methodology produces more realistic datasets in terms of the metric than a random model, other similar datasets, and the same data from different observation intervals.

Helal et al. [60] proposed a methodology to obtain datasets from real-world indoor

spaces. Firstly, they offered a standard definition of SIS datasets; secondly, they utilized a machine learning method to generate datasets similar to existing ones. For the first purpose, the authors used an XML-based standard for sensory dataset description language (SDDL), *i.e.* a hierarchical collection of elements and their attributes such as meta-data, dataset parameters, and sensor events. The authors then utilized a Markov Chain-based algorithm for dataset generation. They used currently available datasets to construct the algorithm's parameters. Specifically, they obtained a transition matrix that describes activity transitions, probabilistically, two probability density functions for each activity representing its duration and respective sensor value. The authors used DTW to compare the synthetic data with ground truth. Using the model is promising, but issues arise for more complex datasets, such as different sensor types.

Veronese et al. [146] proposed a simulation methodology, SHARON, designed to complement real-world datasets and simulate occupants' activities and corresponding sensor readings. SHARON contains two main layers. The first layer is an agent model that generates daily activity schedules based on a motivation-driven approach. The second layer is a sensor model that converts the activities to corresponding sensor readings. Their proposed motivation-driven approach consists of several parameters, *e.g.* the likelihood of performing actions given time, the effect of each action on each need, cause-relation link between needs and actions. They used real datasets to tune these parameters. They showed that the schedule of the generated activities is similar to real-world schedules in terms of the Earth Mover Distance metric.

Elbayoudi et al. [41] proposed a simulator for generating synthetic datasets with different agent traces. The simulator uses the combination of HMM and Direct Simulation Monte Carlo (DSMC) to simulate older adults' behaviour. Their methodology ingests real datasets and mimics human activities and corresponding sensor values. First, a set of parameters about each person's behaviours must be defined *e.g.* number of bathroom visits. Then HMM models a sequence of space (room-level) visits and

their duration. The probability of going from one room to another depends on the time of the day and the current activity. The authors modelled motion sensors as simple room-level presence detectors. The authors evaluated the simulator by profiling two elderly individuals, one with fewer movements. They showed that the simulator could simulate their corresponding datasets by visually comparing their graphical representations.

Noury et al. [102] focused on simulating human activities using their available data in pervasive spaces to generate similar datasets with slight changes. They built an HMM using collected human activities in a smart home environment. Sensors are simplified to represent the location (room) of the agent. They evaluated the simulator based on the location (room number) of the simulated and real occupant's traces based on calculating correlations of the traces. They found that their methodology accurately generates simulated data even for an extended period (one month).

Papamakarios et al. [110] proposed a dataset generation method for producing occupants' trajectories while performing ADL. Their methodology first introduces an ADL specification based on trajectories (how ADLs look in terms of trajectories). The specification has three elements. Firstly, regions of interest denote where agents are most likely present during a specific activity. Secondly, a transition matrix that represents the pairwise probability of regions of interest transitions. Finally, the generation of transition trajectories produces a path (using a variation of the Dijkstra algorithm with randomness and smoothness parameters) between two given regions of interest. The authors used synthetic and real-world data to train two machine-learning models for detecting activities. The models' accuracies are a metric to analyze if the datasets are similar. Their results show that both models perform similarly.

Mendez-Vazquez et al. [91] presented a simulation that combines Markov Chains and Poisson processes for modelling agent's behaviour and probabilistic sensor modelling. Their methodology uses real-world datasets (activity label, time, outdoor temperature and energy consumption) to build a Markov chain model. Then it uses Pois-

son distribution for each activity to generate slightly different timestamps. Finally, sensor readings generation uses probability distributions based on apriori knowledge, like the sensing area of each sensor for slightly randomized sensor readings. The authors used DTW to compare the output of their proposed method, alongside the output of a constant (simulating a single activity) simulation and a random simulation as baseline methods, with ground truth data. They showed that the dataset generated with their proposed method is more similar to the ground truth than the baseline methods.

Insights

The agent models' accuracy in the papers depends on sufficient ground-truth data; thus, it is not immediate and requires expensive real-world experiments. Furthermore, the amount of ground-truth data is related to the granularity level of the simulation methods; that is, more granular agent traces and sensor types require more ground-truth data. That is why the simulators presented in [102] and [41] have room-level agent traces, and the simulator presented in [60] models only one sensor type. Nevertheless, the large granularity of agent traces and few sensor types limit the kinds of applications that the simulators support.

Another approach to cope with the lack of data is used in the simulator presented in [27]. The authors used expert knowledge as prior knowledge to tune the parameters of their data-driven agent modelling approach. However, the process requires manual back-and-forth modification of the parameters for consistency between the designed agent model and the deployment application. Therefore, their approach might be challenging for specific applications or might dismiss certain information due to the subjectivity of the approach.

3.2.6 Dataset Generation

The dataset generation cluster incorporates papers that use simulators to produce synthetic datasets for quickly testing data-driven algorithms, such as activity recognition in indoor spaces or using the dataset to evaluate different sensor placements. The evaluation metrics used are external-validity (such as [29], internal-validity (such as [61]), usability (such as [92, 134]), and scalability (such as [79]).

Table 3.8 shows the taxonomy specifications of this cluster. The table shows that most of the papers did not consider modelling actuators. The reason is that the applications of the simulators are mostly either dataset generation for quickly designing and testing activity recognition models, especially in residential buildings (such as [78, 31]) or quickly evaluating different sensor deployment configurations (such as [150, 55]). In addition, most of the papers used an interactive approach for agent modelling because the approach principally offers more variation of agent traces than model-driven approaches if a large enough number of users interact with the simulators. Therefore, it can be seen as a suitable sampling technique because the resulting dataset is a representative subset of a larger population, which is necessary for training machine learning models. However, the process is time-consuming and requires much effort. On the other hand, the simulators that used model-driven (hierarchy-based) agent modelling [69] claimed that the method is good enough for generating both ADL traces in residential and office routines in public buildings.

Papers Summary

Armac and Retkowitz [8] presented a simulator for SIS applications for residential buildings. The goal of the simulator is to generate synthetic ADL datasets. It specifies SIS devices as reusable software components in the design process. The simulator requires designers to define accessible and inaccessible areas (obstacles) and place devices indoors. They need to import a 2D image of the intended space and define obstacles by placing square tiles of 50 pixels side length. Users can interact with vir-

Table 3.8: Taxonomy specifications of the papers in the dataset generation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Armac and Retkowitz [8] - 2007	Inter-active	LS (smart objects), ES (tempera-ture sensor)	–	Inter-active	Reside-ntial	Dataset gen-eration (for activity recogni-tion tasks)	–
Godsey et al. [53] - 2009	N/A	LS (motion sensor)	–	Inter-active	Reside-ntial	Dataset gen-eration (for activity recogni-tion tasks)	External validity
Prendinger [116] - 2009	Inter-active	IS (RFID)	–	Interactive	Reside-ntial and public	System design	–
Bouchard et al. [21] - 2010	Inter-active	IS (smart objects)	–	Interactive	Reside-ntial	Dataset gen-eration (for activity recogni-tion tasks)	–
Velasquez et al. [145] - 2011	Inter-active	IS (smart objects)	–	Model-driven (hierarchy based)	Reside-ntial	Dataset gen-eration (for activity recogni-tion tasks)	–
Merico and Bisiani [92] - 2011	Inter-active	LS (oc-cupancy sensor), ES (tempera-ture sensor)	–	Model-driven (sequen-tial)	Reside-ntial	Dataset genera-tion	Usability and scala-bility
Buchmayr et al. [26] - 2011	Inter-active	LS (motion sensor), IS (contact switch), ES (tempera-ture sensor)	–	Interactive	Reside-ntial	System design	–
Kormanyos and Pataki [73] - 2013	Inter-active	LS (motion sensor), IS (RFID, smart ob-jects)	–	Model-driven (motivation-driven)	Reside-ntial	Dataset gen-eration (for activity recogni-tion tasks)	–
Mustafa [99] - 2013	Inter-active	LS (motion sensor), IS (pressure and door sensors)	–	model-driven (sequen-tial)	Reside-ntial	System design	–
Caruso et al. [30] - 2013	Model-based	LS (motion sensor), IS (door sen-sor)	–	Model-driven (habits and distrac-tions)	Reside-ntial	Dataset gen-eration (for activity recogni-tion tasks)	Internal va-lidity

Table 3.8: Taxonomy specifications of the papers in the dataset generation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type (continued).

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Garcia-Rodríguez et al. [51] - 2013	Interactive	IS (RFID)	–	Interactive	Residential	Dataset generation (for activity recognition tasks)	–
Cardinaux et al. [29] - 2013	N/A	LS (motion sensor), IS (smart objects, door sensor)	–	Model-driven (motivation-driven)	Residential	Dataset generation (for activity recognition tasks)	External validity
Lee et al. [79] - 2013	Interactive	LS (motion sensor), IS (pressure, vibration, and door sensors), ES (temperature sensor)	–	Interactive	Residential	System design	Scalability
Su and Huang [134] - 2014	Interactive	LS (motion sensor), IS (RFID), ES(temperature sensor)	SC (light switch)	Interactive	Residential	Dataset generation (for activity recognition tasks)	Usability
Synnott et al. [137] - 2014	Interactive	LS (motion sensor), IS (pressure sensor), ES (temperature and humidity sensors)	–	Interactive	Residential	Dataset generation (for activity recognition tasks)	Internal validity
Sernani et al. [127] - 2015	Interactive	LS (motion sensor)	–	Interactive	Residential	System design	Usability
Lee et al. [78] - 2015	Interactive	LS (motion sensor), IS (smart objects)	–	Model-driven (hierarchy based)	Residential	Dataset generation (for activity recognition tasks)	External validity and usability
Vasilateanu et al. [144] - 2016	Interactive	LS (motion sensor)	–	Model-driven (motivation-driven)	Residential	Dataset generation (for sensor placement analysis)	–
Weitz et al. [150]— 2016	Interactive	LS (motion sensor), ES (temperature and humidity sensors)	–	Model-driven (probabilistic sequence of activities and duration)	Residential	Dataset generation (for sensor placement analysis)	Internal validity
Bang and Ko [14]— 2017	Interactive	LS (camera, motion sensor), IS (smart objects), ES (temperature sensor)	–	Model-driven (sequential)	Residential	Dataset generation (for activity recognition tasks)	Internal validity

Table 3.8: Taxonomy specifications of the papers in the dataset generation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type (continued).

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Alshammari et al. [4]—2017	Interactive	IS (pressure and door sensors, smart objects)	–	Interactive	Residential	Dataset generation (for activity recognition tasks)	Usability
Kamara-Esteban [69]—2017	Interactive	LS (motion sensor), IS (smart objects)	–	Model-driven (hierarchy based)	Public and residential	Dataset generation (for activity recognition tasks)	External validity
Spoladore et al. [131]—2017	Interactive	IS (smart objects)	SC	Interactive	Residential	Dataset generation	Usability
Renoux et al. [117]—2018	Interactive	LS (room sensors) IS (smart objects)	–	model-driven (hierarchy based)	Residential	Dataset generation (for activity recognition tasks)	External validity
Golestan et al. [55]—2019	Interactive	LS (motion sensor)	–	model-driven (sequencial)	Residential	Dataset generation (for sensor placement analysis)	–
Ho et al. [61]—2019	Interactive	LS (proximity sensor), IS (smart objects), ES (temperature sensor)	–	Model-driven (hierarchy based)	Residential	Dataset generation (for activity recognition tasks)	Internal validity
Casaccia et al. [31]—2020	Interactive	LS (motion sensor)	–	Model-driven (sequential)	Residential	Dataset generation (for activity recognition tasks)	Internal validity
Bicakci and Gunes [18]—2020	Interactive	LS (motion sensor), ES (temperature and humidity sensors)	–	Model-driven (sequential)	Residential	Dataset generation (for activity recognition tasks)	Internal validity (localization using synthetic dataset)
Francillette et al. [48]—2020	Interactive	IS (smart objects)	INF	Model-driven (Hierarchy based with error probabilities per action)	Residential	Dataset generation (for activity recognition tasks)	External validity

tual agents (up to two agents simultaneously) to produce agent traces by interacting with environmental objects.

Godsey et al. [53] presented a simulator using the game development libraries in

C++ (*i.e.* Gosu and Box2D). The simulator can record and replicate agent traces that users generate. The authors evaluated the simulator in terms of sensor modelling. They conducted a real-world experiment wherein subjects were asked to walk within an instrumented environment at three speeds: slow, average and fast. The authors also performed the same experiment in the simulator. They collected both real-world's and simulator sensor data for comparison. They found that the frequency of synthetic sensor readings is similar to real-world data. They also found faster simulation speeds are more likely to cause data loss than slower speeds.

Prendinger et al. [116] presented a methodology for quickly designing and testing SIS applications using Second Life [124]. The main component of their methodology is called Twin World Mediator (TWM), which provides an interface for modelling real-world sensors and devices. Second Life (SL) offers writing scripts for any object in the environment. TWM is responsible for sending/receiving back-and-forth messages to/from SL and a sensor modelling component. The quality of the methodology depends on the sensor model component accuracy. In addition, roundtrip communication and processing between TWM and SL could be a scalability challenge as the number of devices increases. Therefore, the methodology should be evaluated regarding external validity and scalability.

Bouchard et al. [21] proposed a simulator called SIMACT that allows third-party components to connect to the simulator's database to receive and store real-time sensor readings. In addition, a set of pre-recorded scenarios, *i.e.* agent traces, were also included in the simulator for data consistency. First, however, users can define their scenarios in an XML file. Then, for each agent's action, several parameters are required: duration, the states of the simulation before and after acting (for example, opening a fridge will have an after markup that describes the open state of the refrigerator door), and a parameter that describes if the virtual agent should perform the activity in the given duration (otherwise the agent is assumed to capture individuals with cognitive impairments).

Velasquez et al. [145] introduced a simulator for home care SIS applications to manage the integration of off-the-shelf devices in assisted living environments. The proposed system architecture has two main layers. The fundamental layer, the base layer, deals with all the software (*e.g.* devices API) and hardware (*e.g.* sensors). The second layer, called the meta layer, comprises a database containing all the information from the base layer, a simulation interface, and an interference engine. The interference engine observes the state of the simulation and compares them with a database of pre-defined possible conditions to detect undesirable user behaviours. The process of detecting unwanted user behaviours is illustrated in the paper as an example. The authors believe that their simulation methodology is suitable for healthcare purposes. However, defining the possible states is challenging for most SIS applications.

Merico and Bisiani [92] developed a tile-based simulator. The environment is composed of several tiles in a grid, wherein each tile could be an obstacle, a free space, or a sensor. An agent is a special tile that can overlap free space and sensor tiles. The simulator contains two other components, the DayManager and the HeatManager, that model daylight and heat evolution, respectively. The authors generated synthetic datasets with different sizes (one day's, one week's and one month's worth of data). Their results show that the simulator can produce one month's worth of data in 4533.2 seconds.

Buchmayr et al. [26] presented a simulator that models binary (on/off) sensors such as contact switches, motion and pressure sensors, and temperature sensors. Using a noise signal, the simulator models faulty sensor behaviours by scrambling the sensors' reading. The simulator requires users' interaction to generate agent traces regarding sensor readings by clicking on any sensor. However, their interactive agent modelling approach is challenging for numerous deployed sensors, especially in scenarios where several sensors must be triggered simultaneously.

Kormányos and Pataki [73] designed a simulator for ADL dataset generation. In

their simulator, the agent modelling has three layers. At the top layer, there are complex activities such as cooking; the middle layer consists of simple activities such as going to a location, and the lower layer consists of sensor data representation of the activities in the top layers. Using a motivation-driven approach, their simulator models a single occupant living in a smart indoor space. The priority of performing an activity increases based on the occupant's personality parameters (physical and basic needs such as hunger) and prior activities. After deciding which activity comes next, the agent breaks down the activity into the middle layer activities. Other activities can interrupt the current activity if their priority gets higher than the current activity. The simulator models an ideal, general, form motion sensor that reports the relative distance and angle of the sensed agent(s). How actual agent locations would be mapped to realistic motion sensors output via this general model is left unspecified.

Mustafa [99] proposed a simulation methodology for recognizing lifestyle patterns. The methodology has three components. The first component is a sensor database that stores sensor data. The second component is a knowledge base containing rule-based behavioural patterns using the sensor database. Finally, the third component is a rule-based inference engine that consumes the knowledge base patterns for the recognition task and updates the knowledge base if necessary.

Caruso et al. [30] designed a simulator based on declarative process models [115], *i.e.* a minimal set of rules which should be satisfied as a prerequisite of a software component execution. The simulator uses action theory principles (habits and distractions) for modelling agents. A user must use the model and define the habits and distractions. A planner then converts them into a detailed activity log for a virtual agent to perform sequentially. Using the same agent model in the simulation, the authors illustrated how different sensor deployment configurations result in different sensor logs. The authors used a tool for activity recognition (from the Center of Advanced Studies in Adaptive Systems (CASAS) [34]) and showed that the deployment of motion sensors and door and window sensors significantly increases the recogni-

tion accuracy. However, the definition of sensor configuration considers only the type of the sensors (sensor locations are fixed), which simplifies the sensor deployment evaluation problem.

Garcia-Rodríguez et al. [51] proposed a simulator that can be used to design and evaluate smart spaces using camera and RFID sensors. The simulator provides a view of the deployed cameras and RFID readings so that users can decide their location and rotation. The goal of the simulator is to use sensor readings to detect two types of video events, namely, *output events* such as: begins to walk, detected-man-object disappears, and *instantaneous events* such as holding, not holding, standing. Then, the events are converted into a special format so that an inference algorithm can ingest them as input and output high-level activities.

Cardinaux et al. [29] presented a simulator with motivation-driven agent modelling. They argued that modelling occupants' behaviour is the most challenging part of developing a simulator. In the simulator's agent modelling, the probability of performing an activity increases linearly from the last time of being performed. This model is useful for modelling daily recursive activities, such as cooking meals. The authors showed, specifically for the breakfast activity, that virtual agents perform the activity at a similar time of the day compared to an actual participant. They also showed that the number of kettle sensor activation (it is assumed that an accelerometer is attached to the kettle) is similar in both simulation and real-world experiments.

Lee et al. [79] presented UbiSim, a simulator that allows designers to quickly examine different sensor configurations and find the best one based on their needs. The simulator is envisioned to support a wide range of residential applications because it models LS, IS, and ES sensor types. A user is required to navigate virtual agents and perform activities. The agent traces and corresponding sensor readings are collected temporally to form a synthetic dataset. The authors evaluated the scalability of the simulator by increasing, each time, the number of sensors of a specific type and ob-

taining separate processing times. The authors report that the processing time for all sensor types remains stable with a negligible increase. More specifically, the simulation execution duration using 10 and 170 numbers of vibration-detection sensors are 17.1 (ms) and 18.8 (ms), respectively.

Su and Huang. [134] presented an easy-to-use simulator for non-technical users. Users can construct an indoor space in a 2D mode in their simulator and view it in 3D afterwards. It offers a user interface for placing objects, sensors, actuators and the defining rules that associate sensor readings to actuators. The authors studied the usability of the simulator and found that their simulator is easy to use, but it is not a professional tool for simulating complex scenarios.

Synnott et al. [137] proposed IE Sim, which can generate datasets associated with normal/hazardous scenarios. A user needs to interact with a virtual agent's simulator to perform activities. The intuition about the interactive agent modelling approach is that even a simple activity, like making coffee, can be performed in different ways. The simulator provides an object toolbox, which offers a wide range of indoor objects and sensors. Also, it is possible to create new objects using IE Sim. IE Sim gathers sensor readings throughout the simulation. The authors illustrated that the simulator's data can be used to detect three types of hazardous activities: leaving the bathroom tap running, leaving the oven on and leaving the main entry door open, as well as overlapping activities like sitting on couch and watching TV in the middle of making coffee. IE Sim utilizes interactive sensitive sensors and interactive agent modelling.

Sernani et al. [127] presented a simulator called Smart Tales, a game for non-technical users to increase their familiarity with the core concepts of SIS systems. Users are required to control virtual agents, perform activities in an intelligent indoor space, and score. The deployed sensors must not detect them. On the other hand, an engineer/designer designs each level. If users get recognized, the engineer scores. They evaluated their system using pre- and post-experiment questionnaires regarding subjects' knowledge of smart indoor spaces. They found that using the simulator is

promising for increasing users' awareness.

Lee et al. [78] proposed Persim-3D, a context-driven simulator in Unity 3D. The simulator models agent activities as sequences of actions. The simulator also models contexts, the state of simulation where certain activities can be performed. The authors divided real-world and synthetic datasets into different subsets (as different activities) to evaluate the external validity of the produced synthetic datasets. Each subset's conditional probability of each sensor event, given the previous event, is higher than a defined threshold. They compare each pair of subsets (one from the real world and one from a synthetic dataset) regarding their sensor events. They found that the simulator produces synthetic datasets 81% similar to real-world counterparts.

Vasilateanu et al. [144] presented a simulation for quickly testing sensor placements. They presented a motivation-driven agent modelling based on some attributes such as hunger and energy. The simulator uses the Graphplan algorithm for activity planning, which finds a sequence of activities to reach a specific goal (satisfying hunger, for instance). The simulator models activities in STRIPS (Stanford Research Institute Problem Solver) style [47], in which each activity has pre- and post-conditions. Finally, the A* algorithm is used for path planning. The authors argued that several characteristics of agents could be modelled using their methodology. However, the paper lacks an evaluation of several test cases to support the claim.

Weitz et al. [150] developed a simulator to generate synthetic datasets suitable for training and testing machine learning models for recognizing ADL activities. The simulator uses the discrete event simulation model, operating on three dimensions: dynamic, discrete and stochastic. The dynamic dimension deals with the representation of simulation time and activities. The discrete dimension represents state changes of specific events such as sensors in the simulation. Finally, the stochastic dimension models unpredictable events such as random occupants' activities. To model virtual agents, the authors asked participants to log the start time and duration of certain activities, such as breakfast, lunch, etc. Then the data is used to construct for each

activity a probability distribution. Using the distributions, agents perform a sequence of activities accordingly. The authors demonstrated in an example that the generated sensor readings visualizations could be interpreted as specific occupant's health conditions.

Bang and Ko [14] developed a simulator to obtain datasets for designing activity recognition algorithms. The intuition of the simulator is that it can be used to provide balanced data to learning algorithms, *i.e.* a sufficiently large amount of data for each user behaviour, regardless of how often it might occur in reality. The balanced generation is achieved via interactive-based agent modelling. The simulator extracts several features for each activity, such as location, previous actions, used object ID, time, and environmental information such as temperature, weather and day of the week. The simulator then uses the features for classification tasks. However, the features used might be difficult to describe in real-world applications. For example, the feature indicating whether an object is used, depends on the definition of what "use" of a particular object entails, which could include: placement at a location, continuous interaction, start/stop activation, etc. Also, while balanced data collection is useful, it rarely reflects what is available to a system trained by real, unbalanced data.

Alshammari et al. [4] proposed OpenSHS, a simulator for ADL dataset generation. First, a designer should use Blender 3D to design the space and the deployed devices. Then, users should control an agent with the first-person view to generate agent traces. The simulator stores sensor readings and their states according to participants' interactions throughout the simulation. The authors evaluated OpenSHS's usability using questionnaires given to designers and participants. They found their simulation methodology easy to use from both the designer's and the users' points of view.

Kamara-Esteban et al. [69] presented a simulator, called MASSHA, 'for generating synthetic datasets for residential and public buildings SIS applications. The simulator has two types of activities in its agent model: mandatory and non-mandatory. Virtual

agents prioritize mandatory activities, and if there is no such activity remaining, non-mandatory activities are selected using a roulette-wheel method based on their predefined importance parameter. The authors used the simulator in two test cases to generate their corresponding synthetic sensor readings. Then, they compared the synthetic readings with their real-world counterparts regarding frequency and duration percentage of activation during a session. Their simulator could accurately model LS sensors, particularly motion sensors. However, they found modelling IS sensors challenging because they heavily depend on the agent's actions, specifically their order and the duration of each action.

Spoladore et al. [131] proposed a virtual reality-based intelligent home simulator (SHS). The simulator's service-oriented architecture enables SIS applications modelling with loosely coupled components. The main component of the simulator is called knowledge base home, which defines an ontology about the physical status of the agents and objects. The paper proposed a test case for usability evaluation wherein users should design a kitchen area for visually impaired users. The authors used the System Usability Score (SUS), a questionnaire and the simulator's log (such as timing, errors, bugs and user comments) to measure the simulator's usability. However, the validation was still underway when the paper was published.

Renoux et al. [117] presented a simulator for generating synthetic datasets based on their intelligent home application, E-care@Home. The simulator's agent model is based on a priori knowledge that provides essential information about each activity, *i.e.* mandatory or non-mandatory, minimum and maximum duration times, earliest and latest start times, affordance objects, and prerequisites. The agent model organizes mandatory and non-mandatory activities within a day to ensure that the agent performs mandatory activities and, preferably, performs as many non-mandatory activities as possible. The authors evaluated agent traces by asking several participants to mark the visualized version of each agent trace file as real or synthetic. They found that their agent model can produce a believable activity timeline for a session.

Golestan et al. [55] proposed a simulation-based methodology for evaluating different sensor placements. The authors used the CASi simulator [33] for synthetic dataset generation, which defines space, agent, and sensor models. The sensor events are sent to an inference algorithm to localize the virtual agents performing activities in real-time. The output of the localization and the ground truth location of the virtual agents are compared using their proposed metrics to quantify the quality of information that sensors provide. The authors showed in two test cases that the evaluation algorithms offer helpful information to designers to revise the sensor placements based on context and application.

Ho et al. [61] presented a simulator called SESim, that consists of two main layers: physical and logical. The physical layer is responsible for modelling and visualizing agents, environment and objects such as sensors and furniture. The agent model is based on behaviour trees wherein leaves specify actions, and internal nodes control the flow of performing actions. The logical layer handles and keeps track of the simulation parameters such as clock, agent-objects interactions and sensor reading generation. The authors validated the simulator using three steps. In the first step, the utility of the simulator is examined. They defined a use case and showed that its requirements could be satisfied in the design process. In the second step, the quality of the synthetic dataset generated is evaluated by training and testing an artificial neural network using the dataset for activity recognition. Finally, they argued that the model's performance is reasonable compared to the related work using real-world datasets.

Casaccia et al. [31] developed a simulator for generating datasets suitable for recognizing wandering versus typical trajectories of occupants. First, they defined two models of motion sensors (attached to walls or ceiling, which have circular and cone-like sensing areas, respectively). The authors described four basic trajectories, direct, random, pacing, and lapping. Then, they used the synthetic dataset to train a binary classification algorithm to detect whether trajectories are wandering or normal. The

results show that the algorithm can accurately predict the wandering behaviour by 95%. However, the same algorithm also should be applied to real datasets to observe if the accuracy changes. In addition, it is interesting to investigate whether or not wandering-like activities such as cooking are recognized mistakenly as wandering behaviour. In these cases, using other sensor types can be helpful.

Bicakci and Gunes [18] developed a hybrid SIS simulation tool for quickly generating synthetic datasets. The simulator generates agent traces using real-world occupant's real-time traces or users' interaction with the simulation. The authors demonstrated their simulation by designing a real-world setting and its virtual twin in the simulator. The tool allows manually adding/modifying simulation parameters, such as adding an extra event or changing room temperatures. The authors showed how to utilize different algorithms, such as predicting the occupants' next location (room) to adjust the room's temperature appropriately. They found that the algorithm is accurate using the synthetic dataset. The simulator can also be used for visualizing currently deployed systems.

Francillette et al. [48] proposed a SIS simulation tool that is capable of modelling the behaviour of people with Mild Cognitive Impairment (MCI) or Alzheimer's Disease (AD). The agent modelling is based on a behaviour tree model with error probabilities for each action. They recorded video clips, each depicting an agent performing indoor activities, and showed the videos to an expert to label agents as healthy, MCI or AD. They found that their simulator can accurately emulate MCI or AD individuals if actions have different error probabilities.

Insights

The synthetic datasets generated should be externally valid regarding agent and sensor models. However, Except for [117, 78, 69, 53, 48, 29], less attention has been paid to examining the external validity of the simulators. The evaluation is challenging for agent modelling if the goal is to compare its outputs with ground-truth agent traces.

The reason is that collecting ground-truth agent traces requires video feeds and careful manual annotations of the feeds (for example, in [54]). An alternative approach is presented by Renoux et al. [117]. They studied the believability of their agent model's outputs by asking several participants to mark the visualized version (a timeline that shows the start and duration of each activity throughout a day) of each data as real or synthetic traces. Another approach is to utilize currently available datasets, such as publicly available datasets from the Center of Advanced Studies in Adaptive Systems (CASAS) [34]. However, implementing the virtual twin of the datasets' smart indoor space might be challenging because space modelling accuracy directly affects the agent modelling accuracy. Therefore, using a dataset that their space models are available via IFC for BIM is preferable. The external validity of sensor modelling is also challenging because real-world sensor readings usually contain noisy readings due to environmental factors. Several studies such as [78, 69] evaluated the aggregated sensor readings over a time window to remedy this. Nevertheless, large window sizes limit the types of applications that the simulator can support.

Less attention has been paid to modelling indoor spaces accurately. Detailed geometry specifications, such as using BIM, provide essential information about proper device locations, *e.g.* a motion sensor on the ceiling occluded by a pillar may not cover the area assumed. The BIM also represents realistic indoor object models that can be enhanced to develop interactive objects in simulators.

We found no dedicated dataset generation method for public buildings for related applications such as energy-saving or occupancy detection. However, these simulators should use a model-driven space modelling approach for two reasons. First, interactive approaches are tedious and require a lot of work to model large-scale public buildings. Second, model-driven approaches like BIM contain information regarding Heating, Ventilation, and Air Conditioning (HVAC) and relations between rooms such as shared walls or windows direction (north, south, west, east) that can all be used for temperature evolution modelling. In addition, these applications require multi-agent

Table 3.9: Taxonomy specifications of the papers in the software tools cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Silva et al. [129] - 2014	Interactive	LS (motion sensor)	SC (light switch)	Interactive	Residential	System design	Usability
Bertran et al. [17] - 2014	Interactive	LS (motion sensor), ES (smoke detector)	SC (display text on screen)	Interactive	Residential and public	System design	–

modelling, such as in [69], wherein agents perform individual *and* group activities such as working at a personal workstation or attending meetings, respectively.

3.2.7 Software Tools

The software tools cluster includes studies whose objective is to systematically assist SIS application designers in terms of the systems' easier development, testing and evolution processes. They adapt software development tools and modelling languages in designing SIS applications. The evaluation used in this cluster is the usability in [129].

Table 3.9 shows the taxonomy specifications of this cluster. The thesis focused on SIS applications that include sensors and actuators to showcase the software tools' capability for identifying devices and their relations. The papers do not consider agent models in their development process. That is why they used an interactive approach for modelling agents.

Papers Summary

Silva et al. [129] presented APEX, a 3D simulation-based prototyping framework. APEX supports examining different intelligent home designs and evaluating each in terms of user experience. The development of prototypes is supported through three layers: a simulation layer (using OpenSimulator [106]), a modelling layer (using CPN Tools [36]), and a physical layer (using external devices and real users). OpenSim-

ulator supports cloud simulations. Therefore multiple users can interact inside a simulation environment at the same time. CPN Tools can be used to model dynamic objects and sensors using its collection of predefined modules. However, the agents' behaviour must be manually defined or use the previously recorded instances. The authors evaluated APEX using a questionnaire focused on the framework's usability. Their results show that the simulator is moderately easy to learn and use.

Bertran et al. [17] used a software design approach to develop a simulator, DiaSuite, capable of simulating Sense/compute/control (SCC) applications spanning intelligent indoor spaces, telecommunications, robotics and avionics. DiaSuite offers a complete development tool for SCC applications. The methodology's steps are defining entities as taxonomy layer, defining entities relationships as application layer, using a compiler to generate corresponding implementations, and testing using a simulation interface. The authors showed that several applications from different domains, including intelligent indoor spaces, can be designed using the methodology.

Insights

The studies in this cluster have different motivations. The simulator in [129], APEX, provides a tool to ensure the development process obtains the software verification patterns introduced in [40]. APEX formalized the definition of intelligent indoor space application elements so that the definitions can be checked against the patterns for testing/debugging purposes. However, the simulator in [17], DiaSuite, offers tools to implement the applications more accessible. Therefore, DiaSuite focuses on implementation, and APEX focuses on testing/debugging aspects of SIS applications development process of the application.

Including model-driven agent modelling in the development process offers systematically designing reproducible test cases for debugging purposes. However, it introduces more complexity in the design phase, especially in large-scale applications.

Table 3.10: Taxonomy specifications of the papers in the user experience simulation cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Stahl et al. [133] - 2010	Interactive	LS (bluetooth access point), IS (RFID)	–	Interactive	Residential and public	System design	–
Abade et al. [1] - 2014	Interactive	LS (motion sensor), IS (pressure sensor)	–	Interactive	Public	System design	Usability

3.2.8 User Experience Simulation

The user experience simulation cluster includes studies that whose objective is to utilize simulation environments for quickly developing and validating user experience designs, including assistive technologies or indoor navigation assistants. The typical evaluation criterion of this cluster is usability analysis.

The taxonomy specifications of this cluster are shown in Table 3.10. Since the papers focus on user experience analysis, the agent modelling must be interactive to incorporate end-users' interaction with the application in the simulation environment. In both of the simulators in [133] and [1], designers can deploy different placement of devices and evaluate the users' interaction with the system for optimizing the usability of their SIS application.

Papers Summary

Stahl and Schwartz [133] proposed an intelligent indoor space modelling toolkit called YAMAMOTO, which assists designers in prototyping such systems in a 3D environment with the ability to place sensors and actuators. The simulator has an interactive agent (first-person view), which users can use to generate contextual (location) information. Its authors argued that many applications needing user locations could be studied using YAMAMOTO. Furthermore, the simulator is suitable for visualizing

the state of the real world for a more straightforward debugging process. Specifically, they focused on finding a suitable placement and direction for displays in large conference rooms. Since the simulator is in first-person view, it is easier for designers to observe and evaluate different display combinations based on the user's view and location.

In the paper by Abade et al. [1], the contribution is to propose a prototyping framework using APEX, to quickly collect and evaluate user interactions in pervasive systems to reach user experience goals. They defined the following main components: 1) behaviour component, which manages how services within a SIS application work; 2) physical components that allow connection of real devices such as smartphones and sensors; and 3) communication component for exchanging data between all components in the space. The authors used OpenSimulator [106] to build a 3D model of a library equipped with sensors and large displays. The displays show taken seats (the seats are equipped with pressure sensors). Then, the authors asked several users to use the simulator to walk around and find themselves an empty seat. The goal of using the simulation environment is that designers could evaluate the number and location of displays. The authors found that using virtual spaces is promising for quickly evaluating pervasive systems.

Insights

The focus of the studies is to quickly evaluate the quality of the information that SIS applications provide to users. Another aspect of analyzing such systems is evaluating the occupants' experience using the building. For instance, in a public building navigation application considering contagious diseases like COVID-19, it is interesting to evaluate how successful the occupants are in maintaining social distancing while using the application's recommendations.

3.2.9 Services Debugging

The services debugging cluster, similar to the inference debugging tools cluster, also contains papers that assist designers in developing and testing actuator behaviours. However, most of the papers in this cluster propose a methodology for systematically testing rule-based sensor-controlled actuators. Specifically, their methodologies validate the designed rules by generating different test cases. The typical evaluation criteria include usability [45] and scalability [101].

Table 3.11 shows the taxonomy of the papers in this cluster. All of the papers focused on debugging services with actuators. Therefore, they employed sensor-controlled (SC) actuators. The agent models used are varied. The simulators in [101, 111, 81, 143, 50, 45] use interactive agent modelling because it gives the flexibility to produce a variety of test cases quickly. On the other hand, the simulators in [80, 161, 126] use model-driven agent modelling, which offers reproducibility in terms of test case generation, but it might not be as flexible as interactive agent modelling.

Papers Summary

Nishikawa et al. [101] proposed UbiREAL incorporating a GUI to place devices with two viewing modes, 2D top to bottom and first-person. The path followed by the virtual agent has to be defined before each run. UbiREAL can model physical quantities such as temperature, humidity, electricity, sound level and illumination. The simulator offers a comprehensive and systematic testing/debugging approach (besides visual) to ensure the user's requirements are met. A rule-based formal model is developed for the specification of the requirements. The model uses logical expressions to describe each need by specifying the relations between sensors and actuators. First, their methodology evaluates the correctness of the conditions by assigning several values to sensors and actuators. Then, the simulation examines whether certain situations (occupants' behaviour) can satisfy the requirements. The authors showed that the simulator could be used for testing/debugging in a reasonable time. The simu-

Table 3.11: Taxonomy specifications of the papers in the services debugging cluster. SPM: Space Model, SEM: Sensor Model, ACM: Actuator Model, AGM: Agent Model, BT: Building Type.

Simulator	SPM	SEM	ACM	AGM	BT	Application	Evaluation
Nishikawa et al [101] - 2006	Interactive	ES (temperature sensor), IS (RFID)	SC	Interactive	Residential	System design	Scalability
Park et al. [111] - 2007	N/A	ES (temperature sensor), IS (RFID and door sensor)	SC	Interactive	Residential	System design	–
Lertlakkhanakul et al. [81] - 2008	Interactive	IS (smart objects)	SC	Interactive	Residential	System design	–
Van Nguyen et al. [143] - 2009	Interactive	IS (pressure sensor)	SC	Interactive	Residential	System design	–
Lei et al. [80] - 2010	Interactive	IS (RFID)	SC	Model-driven (sequential)	Residential	System design	Internal validity
Fu et al. [50] - 2011	N/A (home floorplan image)	IS (RFID, lights)	SC	Interactive	Residential	System design	–
Fernandez et al. [45] - 2011	N/A	IS (smart objects)	SC	Interactive	Residential	System design	Usability
Zhang et al. [161] - 2011	Interactive	IS (RFID), ES (temperature sensor)	SC	Model-driven (motivation-driven)	Residential	System design	–
Sernani et al. [126] - 2016	Interactive	LS (motion sensor), IS (RFID), ES (temperature sensor)	SC	Model-driven (motivation-driven)	Residential	System design	Internal validity

lator's frame rate drops only 16% and 20% when using 20 and 50 number of rules, respectively.

Park et al. [111] proposed a simulator, CASS, to help designers detect inconsistencies of a defined set of rules that incorporate a relation between sensor readings, the occupants' locations and the actuators. The authors argued that application-specific

behaviours can be described in a rule-based fashion. A configuration language deals with the space and device specifications, including SC actuators such as air conditioners, fire alarms, dehumidifiers, etc. Their simulator can be used for manually debugging the system.

Lertlakkhanakul et al. [81] focused on using VR to propose a user-centred simulation approach (V-PlaceSims) for intelligent indoor spaces design. Their approach includes several data models describing space, objects/devices and spatial context (definition of users and their activity types). The simulator offers a web-based user interaction in which users can control avatars in the environment. The web-based approach provides a collaborative environment between the designer and the users. Also, multiple users can interact with the system to form a multi-agent environment. The simulator visualizes invisible services that contain different actuation of separate devices based on users' interaction and sensor readings. The simulator is suitable for visually evaluating intelligient indoor space applications.

Van Nguyen et al. [143] proposed an ISS simulator responsible for simulating context-aware (activities) applications. It consists of two key modules: a context retriever module, which requests and receives sensor readings, and a rule-based reasoning module that manages actuators according to the current state of occupants and sensor readings. The authors showed that ISS is capable of activating/deactivating actuators based on sensor readings.

Lei et al. [80] proposed SHSim, an OSGi-based [109] simulator that offers easy configuration of SIS systems using bundles. Bundles are generic modules describing each element of the SIS system, such as sensors or code to execute. Different bundle connections represent different services. The OSGi-based feature of the simulator promises that virtual devices can be transplanted into real devices with no or minor modifications. The authors used several bundles that represent different rule-based test cases. The simulator stores the output of the test cases in a log file.

Fu et al. [50], similar to [80], proposed a simulator based on OSGi. The simulator

can be configured using a few XML configuration files, *i.e.* context XML, device XML, human XML and Environment XML, which include the definition of rule-based inferences, sensors and actuators models, and agent and space models, respectively. Devices are abstracted as OSGi bundles, which can be deployed in the OSGi service platform wherein each bundle can send/get contexts to/from other bundles. In a simple case study, the authors showed that the simulator could generate a sequence of contexts from agents, corresponding sensor readings, and actuators.

Fernandez-Llatas et al. [45] adapted the Human Centered Design (HCD) process for designing smart indoor space systems. HCD is an iterative process that has four steps: conceptualization phase, designing phase, implementation phase, and test phase. The authors argued that simulation is beneficial because it speeds up the process. Therefore, they used the VAALID [142] tool to create and execute simulation scenarios. Before running simulations, users can manage an accessibility check tool that checks if the system conflicts with the user model in terms of accessibility (*e.g.* alarm device volume level is not loud enough for a user with a specified hearing ability). The usability of the simulator is measured by asking 19 designers to work with the tool. Designers were given options to score how they liked the experience: very much, much, moderately, not much, and not at all.

Zhang et al. [161] presented a simulator based on OSGi [109]. The simulator offers modelling rule-based inference engines using a hierarchical architecture. It also allows dynamically modifying the rules and defining a method to detect rule conflicts. The methodology uses a directed graph representation for information circulation in the system. The authors showed that if a cycle exists in the graph, the rules construct an infinite loop, which can be used for debugging. Furthermore, the authors developed three scenarios based on the occupants' location. Finally, they argued that the methodology correctly detects the rules conflicts and the infinite loops.

Sernani et al. [126] argued that unpredictable situations could occur in real-world SIS system designs, which lead to a sophisticated fixing/redesigning process. There-

fore, the authors adopted Morse, *i.e.* an open-source robotics simulator based on the Blender game engine. Morse includes an agent model and several sensor models, including one for sensed temperatures as a function of distance from a heat source. The authors demonstrated their methodology in a case study and showed that the simulator could simulate hazardous scenarios like gas leak detection.

Insights

In most studies, debugging services must be carried out by visually observing the events in the simulator or manually analyzing the simulator's log file. Although these approaches are straightforward to implement, they are error-prone and time-consuming. The only paper that uses an automatic testing approach is by Nishikawa et al. [101], wherein the authors used a systematic testing approach via a rule-based formal model for specifying the relationships between sensors and actuators. First, however, the scalability of their approach needs to be analyzed to specify the type of applications the simulator could support.

A common characteristic in this group was disregarding the uncertainty in the SIS systems, e.g., sensor readings. Probabilistic sensor modelling (such as [54] in the *uncertainty simulation* cluster) could affect the designer's decisions about placement and type of sensors and their associations with actuators. Therefore, probabilistic sensor modelling could introduce additional/different and closer-to-reality test cases.

3.3 Discussion

In this review of existing literature, we defined a unified taxonomy for SIS simulation tools. We demonstrated the applicability of the taxonomy by conducting a survey methodology to identify and collect studies within the scope of the literature. We used the taxonomy specifications to cluster the studies into meaningful groups systematically. However, some cases of specific papers whose clusters where derived incorrectly. This is because of two reasons. First, some simulation tools can be used

for different purposes; therefore, a simulator could belong to multiple clusters. For instance, the simulator presented by Prendinger et al. [116] in the dataset generation cluster can also be a member of the emulation cluster because it incorporates working with real-world devices. Second, a few simulation tools can be counted as outliers. For instance, the simulator by Sernani et al. [127] is presented to increase users' knowledge about how intelligent indoor spaces work, which is not similar to any other simulator. For further investigation, fuzzy clustering approaches can be used to assign to each study a membership percentage to each cluster.

Regarding space modelling, we observe that most of the studies preferred an interactive approach. This is because the interactive method is simpler to implement. However, it is tedious and error-prone when specifying the geometry of space and objects. On the other hand, although model-driven space modelling provides very accurate geometry specifications, it is not readily available (or outdated) for many spaces or buildings.

In sensor modelling approaches, location-sensitive and interaction-sensitive types were more popular than environment-sensitive sensors. Most studies utilized motion sensor modelling as a notable location-sensitive sensor in SIS applications. Motion sensors are preferred because of being non-intrusive, inexpensive, easy to install, yet effective in tracking occupants in indoor spaces. On the other hand, in terms of interaction-sensitive sensors, RFIDs and object sensors were mainly used.

Sensor-controlled actuators dominate studies that used actuators as they are easier to implement and understand. But, complex applications are expected to emerge where actuating involves activity recognition and inference-controlled actuation.

The agent modelling approaches used in the literature were either interactive or model-driven (sequential). The interactive approach can replicate many agent trace patterns but is costly and error-prone. On the other hand, model-driven approaches offer faster and more reliable agent models. Among model-driven approaches, motivation-driven agent modelling could be challenging because it requires

adjusting several parameters. First, it needs prior knowledge to denote a priority for each activity. Then, it needs a method to modify the priorities over time. A more appealing agent modelling approach is hierarchy-based model-driven. As it is stated in [69], a hierarchy-based model-driven approach can simulate different kinds of occupants' behaviours, such as office routines and activities of daily living. Therefore, this approach appears better to satisfy SIS simulation tools requirements.

It is important to note that synthetic agent traces should be externally valid to have a concrete sensor model evaluation. The reason is that two very similar sets of sensor events could result from two different agent behaviours. First, therefore, a simulator could fail to recognize the activities, and the designer could use other sensor deployments to resolve the issue. Second, specific actions, such as cooking, could generate sensor events in a different order every time they are performed. This difference can be investigated by inspecting dissimilarities in synthetic and real-world agent behaviours.

Most papers focused on SIS applications for residential buildings. This is because real-world SIS applications are often assumed to provide services to residents of those buildings. Fortunately, developing SIS simulation tools for such buildings are more straightforward, especially regarding space modelling. However, public building applications are receiving increasing attention [6]. Therefore, SIS simulation tools should utilize methodologies that support modelling large-scale application deployments. Specifically, using model-driven space modellings, such as IFC for BIM, and co-simulation techniques could make the design process more manageable.

Concerning the application domain, most studies concentrated on generating synthetic datasets. Such datasets must have adequate fidelity compared to real-world datasets. Thus, an interesting research question is to investigate the quality of the datasets produced for different use scenarios, such as activity recognition. Real-world and synthetic datasets from the same designed experiment executed in real-world and simulation worlds should be fed to activity recognition algorithms and compared to

the outputs. Another application domain is simulating system design, where the main requirement is to speed up and ease the design process by presenting virtual twins of the intended systems.

We observed that few papers (such as [54]) examined the external validity of the simulators. This is because real-world datasets are not immediately available. SIS simulation tools should replicate real-world settings as faithfully as possible, which means that synthetic and real-world datasets should be similar. In addition, scalability analysis is essential for the simulation tools envisioned to support applications for legacy buildings because models are not readily available and because of the scale of devices and agents needed to simulate.

3.4 Conclusion

Access to a rich sensor readings dataset is an asset in research in SIS applications. Due to the unavailability of datasets or the uneven characteristics of existing ones (different sensor types, different deployment strategies, etc.) an emerging research field is to present methodologies for simulating SIS systems. We reviewed and analyzed the literature to identify a unified taxonomy for the current work. We recognized four main models involved in the simulation. Specifically, every simulation requires defining space, sensor, actuator, and agent models based on their needs. We also identified building type, application, and evaluation criteria that the designers must define based on their needs and requirements. We applied a survey methodology to identify and collect papers within the scope of our survey. The bibliometric results show that our presented taxonomy can divide the related work into meaningful clusters. In each cluster, we found similar papers that focus on either modelling specific phenomena of SIS applications or specific research questions.

Chapter 4

SIM_{sis} : A Simulator for Smart Indoor Spaces

This chapter reports on a SIS simulation methodology that supports the modelling of indoor spaces, the activities of their occupants, and the behaviours of different types of sensors. We argue that, for a simulation to help evaluate a sensor deployment configuration, it has to generate realistic event streams of individual sensors over time, as well as realistic compositions of sensor events within a time window. We have evaluated our simulator for smart indoor spaces, SIM_{sis} toolkit, in the context of an ambient-assisted living platform, Smart Condo, which supports the observation and analysis of Activity of Daily Living (ADL). Our findings indicate that SIM_{sis} produces realistic agent traces and sensor readings and has the potential to support the process of developing and deploying sensor-based applications.

This chapter focuses on R3 research question, specified in Section 1.2: A SIS simulation methodology that can simulate various applications and produce high-fidelity synthetic datasets.

4.1 Introduction

In principle, a simulator for smart indoor spaces must model the space, the sensors embedded in it and their behaviors, and the agents that occupy the space. The *space model* defines the physical world including rooms, furniture, objects, etc. The *agent*

model supports the specification of the space occupants' and the activities they perform in the space. Finally, the *sensor model* describes the types of the deployed sensors and their sensing behavior, as well as the number and placement of the actual sensors in the space.

Space, agent, and sensor models together contribute to the overall generality and external validity of the simulator. The space model should accurately capture the geometry of the space and its interior layout. Ideally it should be general enough to describe most indoor spaces, including home, office, and corporate building layouts as well as the objects typically found in them. The agent model should accurately capture typical occupant behaviors. The model generality relies on capturing the high-level activities in which the agents are likely to engage in a given space, such as Activity of Daily Living (ADL) in homes, office routines in corporate buildings, etc. Finally, the sensor model should enable the simulation of multiple types of sensors, especially those most likely to be deployed in indoor spaces. One expects from the sensor model to at least capture, with some fidelity, the fact that an agent is present within a sensor "coverage area", or that an action was caused by the agent, e.g., flipping a switch.

We argue that the external validity of the simulator behavior should be evaluated through comparative time-series analysis of the agent-behavior and sensor-event traces it produces. In effect, given a specific scenario of a number of agents acting in an indoor space embedded with a number of sensors, the simulator should produce agent activity traces similar to the actual real-world agent activity traces and sensor-event sequences similar to the actual real-world sensor-event sequences.

More specifically, there are two important aspects in examining the validity of the simulated sensor behavior: the activation sequence of the overall set of sensors and the temporal sequence of each sensor's readings. The former denotes the order of activated sensors in a session, and the latter denotes the values of each sensor reading throughout a session.

In this chapter, we describe our work on SIM_{sis} , an integrated simulator for smart indoor spaces that produces sequences of realistic synthetic data sets, using space, agent, and sensor models. Our definition of space model is based on Building Information Modelling (BIM), an industry standard for digital representation of a built facility [13], represented in the International Foundation Class (IFC) format. The IFC data model is an open specification, and an International Standard ISO 16739-1:2018, intended to describe architectural, building and construction industry data. Architectural and engineering tools use IFC to exchange data and geometry about building models between programs. This choice eliminates the need of developing special-purpose space models and enables our simulator to accommodate the complexities and idiosyncrasies of real-world buildings. Furthermore, we define a hierarchical agent-behavior model, in which virtual agents perform activities in the modeled space, towards meeting their high-level objectives. This approach enables the simulation of different occupant behaviors in different settings, such as homes or offices, where agents are likely to have different objectives and perform activities afforded by the setting. Finally, our sensor-behavior model takes into account the space geometry, the agents' activities as well as the properties of the sensor type itself. In addition to sensors, actuators can play an important role in smart indoor space applications, but they are beyond the scope of this work.

We evaluate our simulator against a real-world case study (Smart Condo™ study [94]) and demonstrate that SIM_{sis} simulations generate synthetic data similar to the data collected in the real-world deployment (our ground truth), in terms of (1) the sensor activation sequences (SAS), and (2) the temporal sensor readings (TSR).

4.2 Related Work

In this section, we hand-pick the related works from our literature review in Chapter 3 that are closer to our work.

Park et al. [111] proposed an early context-aware simulation system (CASS).

The main purpose of their simulator is to determine whether rules triggered by sensor readings and the location of simulated users are consistent, i.e., do not result in conflicting decisions. As such, their interest is in actuation and application behavior, assuming it can be described in a rule-based fashion. Their simulator does not indicate any automated means for ingesting floorplans and/or scripting of simulated user activities unfolding over time. A configuration language takes care of all the space and sensor specification, including devices that can be actuated, e.g., air conditioner, fire alarm, dehumidifier, etc. At all times, their simulator remains a closed virtual representation of a physical environment. Conceivably, the rules, once (manually) debugged for consistency, could be transferred over to an actual system. Yet, no validation in a real environment was provided.

Buchmayr et al. [25] presented a simulator using the Microsoft .NET framework. Users are able to import a floor plan image to represent an indoor space. They simulated anonymous binary (on/off) sensors such as contact switches, motion and pressure sensors, and also temperature sensors. The simulator adds a noise signal to sensors' actual signal and generates simulated signals for modeling faulty sensor behaviors. The simulator lacks an agent-behavior model and requires user interaction, *i.e.* clicking on any sensor, in order to advance its state, thus producing synthetic sensor data. Their simulator lacks an explicit space model. In addition, the simulator's alternative to agent model needs sophisticated and precise interactions and it is mainly subjective. Finally, the sensor model does not necessarily generalize in terms of faulty sensor behavior since the problem depends on wide range of parameters that are impossible/difficult to predict before actual implementation, hence this model cannot be validated with real-world ground truth.

Persim-3D [78] is a context-driven simulator in Unity3D. The space model is constructed from scratch by a user through the Unity3D user interface. Their work views each sensor as belonging to one of two categories: location-based and object sensors. The former are triggered from measurements caused by the physical presence of a

human agent inside their "sensing" area, e.g., a pressure sensor. The latter report a change in their state caused, directly or indirectly, by an agent, e.g., opening a door. Confusingly, they consider RFID readers as object sensors, because they report a "contact" event by reading an agent-carried RFID tag, while technically, an RFID reader also has a small coverage area. The agent model consist of actions, activities and contexts. Activities are modeled as sequence of actions and a context defines a state of simulation where a set of activities (with preconditions) are only allowed to be perform. In order to demonstrate how realistic the data produced by Persim-3D are, the authors divided real-world and synthetic data sets into subsets, in which the conditional probability of each sensor event, given the previous event is higher than a threshold. Each subset contains a sequence of sensor events up until a sensor event violates the threshold condition. Therefore, subsets are treated as different activities and sensor events are related and associated together. Then for each pair, one from synthetic data set and one from real-world data set, they evaluate if they have the same sensor events in the same order. They showed that the simulator is able to produce synthetic data 81% similar to real ones. The space model definition heavily depends on users and neglects geometrically important details and may be inaccurate. Finally, although the sensors' behavior strongly depends on the agent's trace, the methodology does not validate the agent model. This is important because of two reasons: first, two similar sets of sensor events could be results of two different agent behaviors, e.g., two different activities in kitchen trigger relatively the same set of motion sensors. Thus, the simulator could fail recognizing the activities, and the designer of such smart indoor spaces could use different sensor deployments to resolve the issue. Second, a specific activity, like cooking, could generate sensor events in different order. This difference can be investigated by inspecting dissimilarities in synthetic and ground-truth agent behaviors.

There are several studies (like [85, 108]) based on IE Sim [136]. As with Persim-3D, a user constructs the space model within the simulator, a step that inherently

endangers the accuracy of the space model. The simulator models door contact, PIR, and pressure sensors. IE Sim requires an operator to control virtual characters and perform activities by interacting with the environment.

Lundstrom et al. [85] used IE Sim to simulate ADLs. They showed that the number of PIR sensor readings over an interval follows Poisson distribution. Also Ortiz-Barrios et al. [108] statistically studied the feasibility of using IE Sim in order to generate realistic data sets. They found that since IE Sim needs a human operator, the software fails to accurately model agents in terms of activity duration. In the second study [108], the authors reported that the number of sensor events per activity is significantly similar to real-world data (with confidence level of 95% and p=0.141). If they separate the events based on the sensor type, *i.e.* door sensor and pressure sensor, the similarity in particular is not significant for pressure sensors. These studies examine as a validity criterion the number of sensor events per activity, and ignore the temporal ordering of these events.

Similarly, Renoux et al. [117] presented a simulator for generating ADL data sets based on their smart home application, E-care@Home. The space model should be defined by users given a floor plan. Sensors are associated to indoor objects, like a couch or oven, and their states change when a simulated human interacts with the objects, and room sensors are ambient sensors, *i.e.* motion and temperature sensors. Their agent model is based on a priori knowledge that provides important information about each activity, *i.e.* mandatory or optional, minimum and maximum duration time, earliest and latest start time, affordance objects, and prerequisites. Overall, agents organize mandatory and optional activities within a day in order to make sure mandatory activities will be performed besides as many as optional activities. In terms of space model, indoor space definition needs a sophisticated effort in order to be accurate. In addition, there is no specific definition for indoor objects. Therefore the space model representation makes the simulation less practical and less accurate to be used for any intended application. The agent model is evaluated by asking a

number of participants if each sequence of activities comes from a real or artificial agent. They found that their agent model can produce "believable" activity timeline for a session. Although, their evaluation is limited and could be subjective, their agent model mimics human behaviors accurately. However, since the agent model needs a priori knowledge, its accuracy depends mostly on expert knowledge, which could be costly in time and effort. The sensor model is also compared in terms of percentage of activation over each day, which is not adequate, because most of the smart home applications involve time-series data analysis for localization and activity recognition.

OpenSHS [4] is another smart indoor space simulator for ADL data set generation which can be used by researchers in the field of internet of things and machine learning. A designer is required to use Blender 3D to design indoor spaces. Then, participants interact with the space to generate agent trace. OpenSHS supports pressure and door sensors, lock devices, appliance switches, and light controllers, and stores their readings and states according to participants interaction. The authors evaluated OpenSHS in terms of usability analysis using questionnaires given to both designers and participants, and they found the results promising. However, the synthetic data sets are required to be validated in terms of agents trances, sensors readings, and devices states. Yet again, the definition of space and agent models are burdensome and time consuming and are subject to users error.

In [69], MASSHA, an agent-based simulator was presented for generating synthetic ADL data sets. A space model is defined by a user given a set of objects and building elements. The agent model in MASSHA is carried out by a hierarchical model where activities consist of a sequence of actions. Agents prioritize mandatory activities, and if there is no such activity, other activities are selected using a roulette-wheel approach based on their importance. They were able to model sensors in terms of frequency and duration percentage of activation during a session. However, similar to previous attempts, the accuracy of the space model depends directly on high-effort invested by users. Objects and building elements also limit the simulator's practical

usage. The agent model enables modeling single or multi agent scenarios such as smart home or office building applications; but, it lacks validation of the model with real-world ground-truth data. If we adopt and modify the terminology used in Persim-3D to location-sensitive (LS) (for their location-based) and interaction-sensitive (IS) (for their object) sensors, what MASSHA demonstrated using their ground-truth data sets was the power of LS over IS sensors. However, the temporal granularity of MASSHA, is big for fine-grained ADL.

Masciadri et al. [88] utilized a simulator called SHARON, in order to, first, complement real-world data sets, and second, to simulate inhabitants' activities and corresponding sensor readings. SHARON has two main layers (lacks a definition of space model): a top layer (agent model), which generates daily activity schedules based on a motivation-driven approach, and a bottom layer (sensor model), which converts the activities to corresponding sensor readings. They showed that the schedule of the generated activities is similar to real-world schedules in terms of the Earth Mover Distance metric. However, it requires real-world training sets; hence, agent model accuracy depends on having sufficient ground-truth data; thus it is not immediate, and requires real-world experiments. Additionally, the distribution of the synthetic sensor reading is compared against real-world ground truth given three specific activities. The comparison suggests that for "cleaning" and "lunch" activities, the sequences of activation were random, but for "lunch" activity the overall procedure was the same. The "shower" activity, however, had almost the same order of actions to its ground-truth peer. This comparison shows that although each activity has a set of predefined actions, they can be performed in different order, hence it is difficult to validate the behavior of simulated agents and sensors.

Based on the related work and the taxonomy defined in Chapter 3, *space models* are generally defined by using the *interactive* approach. In addition to efforts needed to design such space as realistically as possible, the simulator software does not necessarily offer capabilities to model any intended indoor space. Moreover, this approach

makes the simulator less realistic because it may not be able to accurately take into account the complete geometry specifications.

Agent models are defined using the *model-driven* approach. Virtual characters interact with objects based on some behavioral policies, like motivation-driven [88] and hierarchy-based [117, 69]. The hierarchy-based behavior can be adjusted to fit the intended context, e.g., performing ADLs, or office routines. Validation of agent models are carried out by comparing the activity distributions in simulated versus real data [88], and comparing simulator generated timeline of activities versus human generated peers [117]. The comparison by [88] is not temporal, which is necessary for smart indoor space applications. For example, we are required to anticipate the time and order of activities in order to decide about energy saving policies of a building. The comparison by Renoux et al. [117] is temporal; however, not only it could be subjective, but also it is limited to activities performed in specific time and duration. Instead, it is important to compare the temporal trace of agents based on location and activity against the real-world ground truth.

Sensor models are shown to be accurate when modeling LS sensors. In the research most similar to ours [69], the simulator models LS sensors for one (out of two) data sets, which has regular daily behaviors of an office space. It was found that the simulator produces hourly activation of sensors similar to ground truth. Nevertheless, this granularity level of analysis is not adequate for many smart indoor space applications. For instance, the simulation of an aging-in-place application fails to meet its requirements as aggregating synthetic data in one-hour intervals does not provide enough contextual information for caregivers. Our simulator only needs seven minutes data aggregation to produce realistic sensor readings.

The other sensor type that previous works mostly modeled are IS sensors. However, this type of sensor is heavily tailored to actions within every activity. Simulation of this sensor type can replicate real-world environments only when enough, and in the right sequence, actions are performed within every activity. For various typical

indoor activities, this may be unlikely or burdensome in practice. This limitation is reported in [78, 69]. More specifically, Kamara-Esteban et al. [69] found that their simulator does not produce hourly activation of sensors similar to one (out of two) of their real-world data set (single user data set), where IS sensors were deployed. The reasons were: (1) they were not able to match the real-world short-term (fine-grained) annotations in their experiments, (2) wrongly annotated activities and actions within each activity, (3) subjects not following consistent behavioral patterns during real-world experiments, and more importantly, (4) dependence of the IS sensors to sequence of actions within activities.

4.3 Simulation Methodology, Models, and Validity Metrics

Figure 4.1 illustrates the architecture of the SIM_{sis} toolkit. It consists of four main components: (a) human-activity simulation, (b) sensor-event simulation, (c) validity assessment of agent traces, and (d) validity assessment of sensor events. The first two components implement the simulation functionalities of the toolkit, while the latter two implement three metrics designed to evaluate the validity of the simulation and the synthetic data produced.

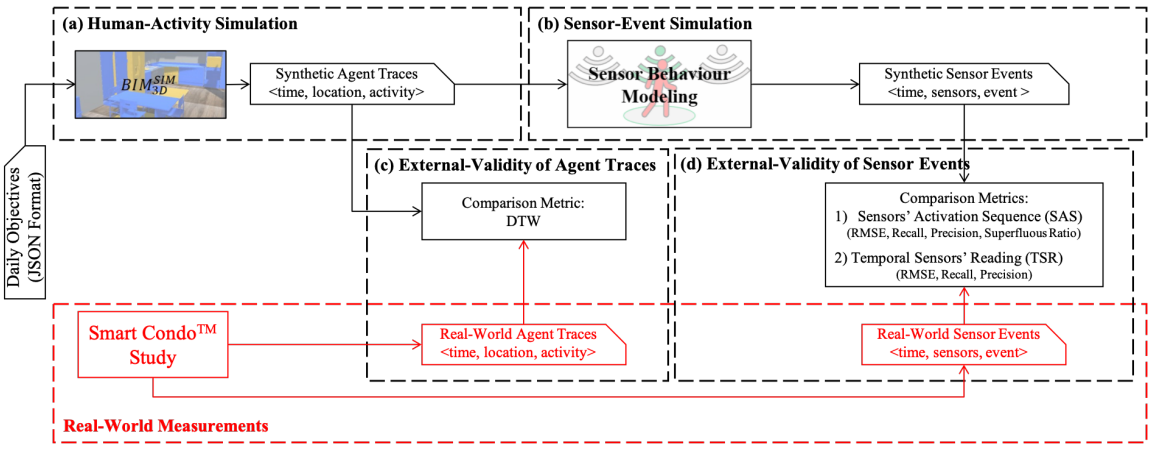


Figure 4.1: The software architecture of the SIM_{SIS} toolkit. Real-world measurements module is shown in red color.

BIM_{3D}^{Sim} [162] renders a 3D model of the simulated space based on its IFC model, extended with special-purpose object annotations specifying the user interactions that these objects afford. Virtual agents perform an activity script that meets their daily objectives, resulting in synthetic agent traces. Given these traces as input, the sensor-behavior modeling component generates synthetic sensor events. To evaluate the external validity of the synthetic agent races and corresponding sensor events, the SIM_{sis} toolkit supports three metrics: dynamic time warping (DTW) for comparing synthetic agent traces against real-world agent traces, and two, sensor-sctivation sequences (SAS) and temporal sensor readings (TSR), for evaluating synthetic sensor events against corresponding real-world sensor events.

To evaluate our simulation methodology (components A and B), we compare the synthetic agent traces and sensor events it produces against a real-world study in the Smart Condo™ study [94]. In this study, participants were given a scripted sequence of typical activities of daily living, and they were asked to perform them in the order listed in their script (more details about this study is provided in Experimental Evaluation section). The Smart Condo™ space was instrumented with motion and beacon sensors, and their readings constitute the ground truth for our empirical evaluation of SIM_{sis} toolkit in this chapter. The ground truth regarding the participants' movements and activities was established by manually annotating video recordings of the study, in 3-second intervals.

In principle, there are two sources of uncertainty in establishing the ground truth in such applications. First, *timestamp ambiguity*, which stems from inaccurate timestamps of sensor events since various data sources have different clocks which are not necessarily synchronized. Furthermore, sensor events may be lost or received by the application out of order due to problems with the network infrastructure. The second source of uncertainty (*behavioral ambiguity*) lies in the behavior of participants, because they might perform activities slightly (and sometimes noticeably) different. Therefore, the first type of uncertainty captures impairments of the infrastructure,

while the second reflects the non-uniformity or subjectivity in the execution of tasks across human populations.

4.3.1 Human-Activity Simulation

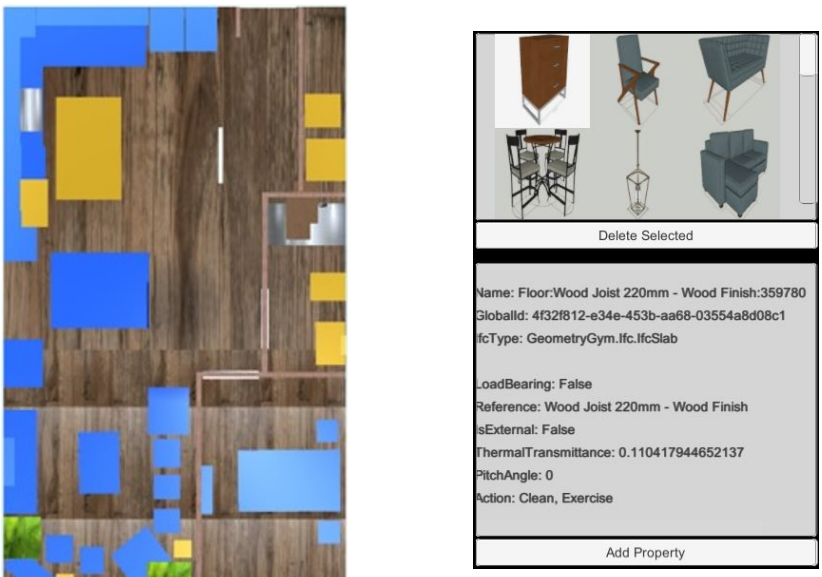
This module uses our BIM_{3D}^{Sim} [162], to simulate the occupants' activities in indoor spaces. This component involves two main features. First, it enables the simulation designer to review the BIM elements and specify the interactions they afford. Second, based on a user-configurable model of the simulated agents' daily objectives and the desired time period, it visually simulates and produces a trace of all agents' activities as they pursue their tasks. In the following, each feature is explained in detail.

Space Modelling: The BIM Editor

In order to run a simulation, a BIM file of the intended indoor environment (in IFC format) is needed. We have implemented in Unity 3D a BIM Editor (Fig. 4.2) that visualizes the input IFC file and allows designers to modify and add a layer of information with the types of interactions that the model elements afford. For instance, if an object is defined as "sittable", agents may choose this element to sit on, when their task becomes to sit. The edited BIM can then be exported as a new enriched IFC file. This is the type of BIM that BIM_{3D}^{Sim} expects, namely a BIM with objects whose properties list the actions that the simulated agents can perform with them.

Agent Modelling: Task Planner

This module takes as input a user-configurable set of *daily objectives* for each agent and outputs a *plan*, i.e., a sequence of time-stamped activities, for each agent. The agent's daily objectives represent the behavioural model of each agent, in terms of the activities they are capable of (and should be) performing during the day, as well as the frequency, duration, pre- and post-conditions of these activities. The agent's daily-objectives specification is non-deterministic. The agent's plan is a deterministic



(a) a rendered IFC file. (b) a panel for adding/deleting objects, or adding properties to the model objects.

Figure 4.2: Screenshots of the BIM Editor used in SIM_{sis} toolkit.

time-stamped and coordinate-aware sequence of activities that meets the constraints implied by the daily objectives. The daily-objectives specification is stored in a JSON file; a sample excerpt describing sleeping and eating behavior is shown below.

```
{
  "Actions": {
    "Sleep": {
      "name": "Sleep",
      "duration": 480,
      "probability": 100,
      "occurrence": 1,
      "requires": [],
      "post": [],
      "times": [
        [ 0, 8 ]
      ],
      "aliases": []

    },

     "Eat": {
      "name": "Eat",
      "duration": 30,
      "probability": 100,
      "occurrence": 3,
      "requires": [ "Cook" ],
      "post": [ "Wash" ],
      "times": [
        [ 8, 9 ],
        [ 12, 13 ],
        [ 18, 19 ]
      ],
      "aliases": []
    },
```

```
  }
}
```

The properties of the actions are explained below:

- *Duration:* How long the activity will last in simulation time, indicated in minutes. A value of 480 would translate to 8 hours of simulation time.
- *Probability:* The program takes in a list of actions and randomly sorts the list every time a new action needs to be selected. The actions are then selected sequentially. This selection method ensures the equal probability of all actions being selected, which may not be desirable for agent behaviour. To remedy this, the probability we define is the chances of performing the selected action, allowing for more variation in behaviour. Probability is indicated from a 0 - 100. If the action is exclusively a precondition, its probability is 0, as it will be forcibly acted upon when needed, and no other time.
- *Occurrence:* The maximum number of times this action can occur per day.
- *Requires:* Actions that must be performed before this action.
- *Post:* Actions that must be performed after this action.
- *Times:* Time constraints on when this action can be performed. The range of times and values are indicated using a 24-hour clock. (Ex. [13, 16])
- *Aliases:* Alternate names for the action the agent is taking to make them seem more lifelike. Prevents the creation of redundant actions, as "Eating lunch" and "Eating dinner" would effectively be the same thing.

Note that each activity has to be associated with at least one (and possibly more) object(s) in the IFC file, which affords this activity. Within each simulation scenario, multiple behavioral profiles may be specified and each simulated agent is associated with one of them.

As mentioned above, the task planner takes as input the daily objectives JSON file and provides a plan for each agent ($< time,\ location,\ activity >$). This plan describes what activities the agent should be doing together with the time and location for each. Our simulation is defined by the space, the objects in the space, the agents, and their activities. The agent is an entity that lives and acts in the space and has its own behavioural model. The agent's behaviour can be as complex as desired and depends on spatial restrictions such as physical barriers. The agent interacts with the objects in the space, and it is these objects that enable the agent to perform its tasks. These objects also have their own constraints regarding the agent's interactions.

An agent trace of length N, $\mathcal{A}$ is denoted as a sequence of the agent's locations ($\mathcal{L}_t$) and activities ($\mathcal{K}_t$) at timestamp t, $1 \leq t \leq N$, shown in Equation (4.2). Every two subsequent timestamps are separated by a time period τ, where τ is the frequency with which sensors emit their observations.

$$\mathcal{A} = \{(\mathcal{A}_1), (\mathcal{A}_2), ..., (\mathcal{A}_N)\} \tag{4.1}$$

$$= \{(\mathcal{L}_1, \mathcal{K}_1), (\mathcal{L}_2, \mathcal{K}_2), ..., (\mathcal{L}_N, \mathcal{K}_N)\} \tag{4.2}$$

4.3.2 Sensor-Event Simulation

The sensor behavior modeling component includes a specification of the sensor configuration in the space. We model two popular sensor types (according to Chapter 3), *i.e.* LS and IS. The sensor behavior modeling component reads the synthetic agent traces as input and generates a sequence of sensor state-events (fired/unfired) for every sensor in the modeled space.

Modelling Location Sensitive (LS) Sensors: Each LS sensor is defined as tuples s_{LS}=$\{\texttt{ID}, x, y, C\}$, where `ID` is the unique sensor identifier, $\{x, y\}$ is the location of the sensor in the space, and C is the sensitivity of the sensor. In order for an LS sensor to fire at a particular timestamp (t), the agent's location at this time ($\mathcal{L}_t$) has to be within the sensor's effective coverage area. The effective coverage, C, can be

defined as $C{=}C_0 \cap G$ where C_0 is the coverage area if no geometry, e.g., occlusion and boundaries, were introduced by the space, and G is the part of the geometry within which the corresponding sensor is placed, *e.g.* boundaries imposed by the geometry of the space. Depending on where a sensor is placed, the same C_0 can result in different C. Moreover, depending on the particular sensor technology the geometry of the space can play a more (or less) significant role. For example, PIR sensors are limited by occlusion, while BLE beacons are much less so their signal can pass through most residential walls and furnishings, conceptually making G equal to the entire 2D plane. In this chapter, we conflate into C the impact of the unhindered coverage, C_0, and the geometry-specific impact G.

The synthetic sensor events capture a non-idealized sensor behavior whereby, the $\hat{e}_t^{\texttt{ID}}{=}1$ and $\hat{e}_t^{\texttt{ID}}{=}0$ (the synthetic sensor $s.\texttt{ID}$ "fired" and "unfired", respectively at a particular timestamp t) is related probabilistically to the agent's movement to a location within the sensor's coverage area, $C^{\texttt{ID}}$. Specifically, we define an asymmetric error for LS sensors to fire as follows:

$$Pr(\hat{e}_t^{\texttt{ID}}{=}1 \mid \mathcal{L}_t \in C^{\texttt{ID}}){=}\beta(\mathcal{L}_t, C^{\texttt{ID}}, \{x,y\}^{\texttt{ID}}) \tag{4.3}$$

$$Pr(\hat{e}_t^{\texttt{ID}}{=}0 \mid \mathcal{L}_t \in C^{\texttt{ID}}){=}1 - \beta(\mathcal{L}_t, C^{\texttt{ID}}, \{x,y\}^{\texttt{ID}}) \tag{4.4}$$

where $0 \leq \beta(\mathcal{A}_t, C^{\texttt{ID}}, \{x,y\}^{\texttt{ID}}) \leq 1$ captures the probability that an agent's location $\mathcal{A}_t$ at the particular timestamp t will be detected as such by the $\texttt{ID}$ sensor, at location $\{x,y\}^{\texttt{ID}}$, with an effective coverage $C^{\texttt{ID}}$. As an example, also used in the evaluation section and observed in several related works [2, 3, 128], $\beta(\mathcal{A}_t, C^{\texttt{ID}}, \{x,y\}^{\texttt{ID}})$ can be defined as a bivariate normal distribution where the probability depends only on the distance between $\{x,y\}^{\texttt{ID}}$ and $\mathcal{A}_t$ when $\mathcal{A}_t$ is inside $C^{\texttt{ID}}$, and is zero otherwise (outside of $C^{\texttt{ID}}$). The intuition of this assumption is that the closer the agent is located to the "center" of $C^{\texttt{ID}}$, defined by $\{x,y\}^{\texttt{ID}}$, the more likely it is that the sensor fires.

In this study, we model two off-the-shelf LS sensor types, *i.e.* binary infrared motion sensors and beacon sensors. The binary narrow-beam motion sensors are

attached to the ceiling and fire if an agent moves within the coverage area underneath. The beacon sensors are attached to walls or objects and use the received signal strength to fire only when a transmitter is in proximity (signal stronger than -70 dBm, which translates roughly to a distance of 1 m in our setup). Similar to Mohebbi et al. [94], our simulated sensor deployment includes 14 and 31 instances of binary motion sensors and beacon sensors respectively, distributed inside of the space model with a dimension equal to Smart Condo™ (10.5 m $\times$ 6.6 m). Sensor configuration (type, ID, location, coverage area description, and room) reflects the configuration deployed in the Smart Condo™ study.

Modelling Interactive Sensitive (IS) Sensors: Each IS sensor is defined as tuples s_{IS}=$\{\texttt{ID}, x, y, C, \mathcal{Q}\}$, where $\mathcal{Q}$ is a physical quantity that an IS is sensitive to. In order for an IS sensor to fire at a particular timestamp, similar to LS sensors, the agent's location at this time has to be within the sensor's effective coverage area as well as the agent's activity at this time has to produce a specific physical quantity that the sensor is sensitive to.

Similarly, we define an asymmetric error for IS sensors to fire as follows. Notice that $f(.)$ is a function that takes in activity K_t and outputs the physical quantity that the activity produces. For example, $f(K_t = \texttt{Sit})$ produces "pressure" which can be recognized by a pressure sensor that is close enough to the agent.

$$Pr(\hat{e}_t^{\texttt{ID}}{=}1 \mid \mathcal{L}_t \in C^{\texttt{ID}}, f(\mathcal{K}_t) = \mathcal{Q}){=}\beta(\mathcal{L}_t, C^{\texttt{ID}}, \{x, y\}^{\texttt{ID}}) \tag{4.5}$$

$$Pr(\hat{e}_t^{\texttt{ID}}{=}0 \mid \mathcal{L}_t \in C^{\texttt{ID}}, f(\mathcal{K}_t) = \mathcal{Q}){=}1 - \beta(\mathcal{L}_t, C^{\texttt{ID}}, \{x, y\}^{\texttt{ID}}) \tag{4.6}$$

In this study, we model three off-the-shelf IS sensor types, *i.e.* pressure sensors, electricity sensors, and accelerometer sensors, that are sensitive to pressure, electricity, and motion quantities, respectively. The sensors can be attached to any object in the indoor space. Our IS sensor model ignores the agent's location for electricity sensor as its operation does not rely on the agent's location. Nevertheless, we cannot evaluate

IS sensors due to unavailability of ground truth IS sensor readings in [94].

4.3.3 Validity of the Agent Traces

This SIM_{sis} component compares ground-truth agent traces (in this chapter, collected through manual annotation of videos of our real-world case study) against synthetic agent traces, using a metric based on a variant of dynamic time warping (DTW) [98] method. Based on Equation (4.2), we represent the real-world and the synthetic agent traces as $\mathcal{A}$ and $\hat{\mathcal{A}}$, respectively. Furthermore, we adopt the Euclidean distance as the basic distance metric between two corresponding real-world and synthetic agent locations at a particular timestamp, $\mathcal{A}_t$, and $\hat{\mathcal{A}}_t$.

DTW temporally aligns $\mathcal{A}$ and $\hat{\mathcal{A}}$ elements in order to minimize the aligning cost, producing a so-called *optimal warping path* (C_{DTW}), under certain conditions, *i.e.* boundary, monotonicity, and step size. An *accumulated cost matrix* shows the alignment cost between all the location pairs, and the optimal warping path is a path that connects pair $(\mathcal{A}_1, \hat{\mathcal{A}}_1)$ to pair $(\mathcal{A}_n, \hat{\mathcal{A}}_n)$, vertically, horizontally, or diagonally. Given $C_{DTW}(\mathcal{A},\ \hat{\mathcal{A}})$ as the cost of the optimal warping path found by DTW, we calculate the similarity percentage metric between $\mathcal{A}$ and $\hat{\mathcal{A}}$ as follows:

$$S(\mathcal{A},\ \hat{\mathcal{A}}) = \frac{C^{\texttt{MAX}}_{\texttt{DTW}} - C_{\texttt{DTW}}(\mathcal{A},\ \hat{\mathcal{A}})}{C^{\texttt{MAX}}_{\texttt{DTW}}} \times 100 \tag{4.7}$$

where $C^{\texttt{MAX}}_{\texttt{DTW}}$ is the maximum for $C_{\texttt{DTW}}(\mathcal{A},\ \hat{\mathcal{A}})$, occurs when real-world and synthetic traces have the maximum distance in every data point ($C^{\texttt{MAX}}_{\texttt{DTW}} = \arg\max_{i,j}(C_{\texttt{DTW}}(\mathcal{A}_i,\ \hat{\mathcal{A}}_j))$).

4.3.4 Validity of Sensor Events

SIM_{sis} compares the synthetic sensor events against their real-world counterparts using two metrics: (1) sensor activation sequence (SAS), and (2) temporal sensor readings (TSR). The former quantifies the degree to which the simulator maintains the order of fired/unfired events as compared to the real world; the latter reflects how accurately the sensor model simulates each sensor's behavior throughout each simulation session.

Sensor Activation Sequence (SAS)

The sensor activation sequence captures essential information about how indoor activities are being performed throughout a period of time, depending on the order of activities, and the amount of time spent on each activity. For example, in healthcare applications, caregivers can apply appropriate interventions if they observe irregularities to ADL, e.g., out of order activities, or spending too much time on a simple activity. Hence, the sequence and duration of activities need to be accurately reflected by a simulation.

Our toolkit (Figure 4.1) offers two metrics for evaluating the validity of the synthetic sensor events produced: the SAS metric examines whether sensor events emitted by the simulated sensors are in the same order as the events emitted by their real-world counterparts, assuming that the real-world and simulated deployment configurations are the same, including M sensors. We define two matrices, E (Equation (4.8)), and $\hat{E}$ (Equation (4.9)), as the representations of the real-world sensor events and synthetic sensor events, respectively, at N timestamps (columns are added with τ rate).

$$E_{M\times N}=\begin{bmatrix} e_1^1 & e_2^1 & \dots \\ \vdots & \ddots & \\ e_1^M & & e_N^M \end{bmatrix},\ e_n^m \in \{0,1\},\ n\in\{1,2,...,N\}, m\in\{1,2,...,M\} \tag{4.8}$$

$$\hat{E}_{M\times N}=\begin{bmatrix} \hat{e}_1^1 & \hat{e}_2^1 & \dots \\ \vdots & \ddots & \\ \hat{e}_1^M & & \hat{e}_N^M \end{bmatrix},\ \hat{e}_n^m \in \{0,1\},\ n\in\{1,2,...,N\}, m\in\{1,2,...,M\} \tag{4.9}$$

where every e_n^m and $\hat{e}_n^m$ indicate the event emitted at a particular timestamp n by the real-world sensor with $s.\texttt{ID}{=}m$ and its synthetic counterpart, respectively.

We compare each column in matrix E, $Col_n(E),\ n \in \{1,2,...,N\}$ against the corresponding column in matrix $\hat{E}$, $Col_n(\hat{E}),\ n \in \{1,2,...,N\}$ to quantify the SAS similarity, by observing the degree to which the sensor events in the two columns, *i.e.* elements with a value greater than zero ($\{i \mid e_n^i > 0\}$ and $\{i \mid \hat{e}_n^i > 0\}$) match. We

denote $Col_n(E)$=**0** and $Col_n(\hat{E})$=**0** if there is no sensor event value greater than zero in column n in E and $\hat{E}$, respectively, which is the case if no sensor fired at timestamp n. Due to the inherent uncertainty in the timestamps of the sensor events, we use a windowing approach in our comparison.

Algorithm 2 describes the SAS algorithm. Given a column corresponding to timestamp n, for each event q in $Col_n(E)$, the algorithm determines the most similar event, $\hat{q}$, that occurred in columns of $\hat{E}$ in the window time-frame. The similarity measure in this algorithm for two events, *i.e.* two sensor IDs i and j, is based on Euclidean distance of sensor locations, d_{ij}=$||\{x,y\}^i - \{x,y\}^j||$, times a parameter, ρ_{ij}, which is the length of the shortest path between the rooms where the two sensors are located, in a graph representation of the indoor space. In the graph, every room is represented by a node, and there is an undirected edge between two nodes if and only if the rooms are adjacent. We denote θ as a special event for each column n if $Col_n(E)$=**0** or $Col_n(\hat{E})$=**0**. Accordingly, we assume the following exceptions:

$$d_{i\theta}=d_{\theta j}=d_{\texttt{MAX}} \tag{4.10}$$

$$\rho_{i\theta}=\rho_{\theta j}=\rho_{\texttt{MAX}} \tag{4.11}$$

$$d_{\theta\theta}=\rho_{\theta\theta}=0 \tag{4.12}$$

where $d_{\texttt{MAX}}$ is the maximum distance that two sensors could have in the space, *i.e.* furthest corners of the space, and $\rho_{\texttt{MAX}}$ is the diameter of graph G, *i.e.* longest shortest path in graph G.

Algorithm 2 compares each reading from matrix E against their match in matrix $\hat{E}$ in terms of root mean squared error (RMSE), recall, precision, and superfluous ratio (σ_W), at different window sizes. Given a window size W, the superfluous ratio indicates, in average, the number of synthetic events that the algorithm did not map to any real events in each window time-frame proportion to the number of events in the window (Equation (4.13)). The metric is normalized, a value equal to zero shows that there is no excess synthetic events left, while any greater value shows that there

Algorithm 2 (SAS) Finds a matching list for real-world sensor activation sequence (SAS) regarding window size and calculates metrics.

Inputs: **(1)** $E_{M\times N}$: Real-world sensor events (elements e_{ij}), **(2)** $\hat{E}_{M\times N}$: Synthetic sensor events (elements $\hat{e}_{ij}$), **(3)** W : Window size, **(4)** ρ : All-pairs shortest path between sensors' rooms
Output: **(1)** $\overline{\sigma}_W$ Superfluous ratio for W (average across N), **(2)** `RMSE`: Root Mean Square Error (average across N), **(3)** `Recall`: Recall score (average across N), **(4)** `Precision`: Precision score (average across N)

$\mathcal{X}_t = \{\} \qquad \forall t \in \{1, \cdots, T\}$
for n = 0 : N **do**
  $\mathcal{C}$=$\hat{\mathcal{C}}$=$\mathcal{M}$={}
  $\mathcal{S}_{\texttt{Precision}}$=$\mathcal{S}_{\texttt{Recall}}$=0
  if $Col_n(E)$=**0 then**
    $\mathcal{C}$=$\{\theta\}$
  else
    $\mathcal{C}$=$\{i \mid e_{in} > 0\}$
  end if
  $\hat{\mathcal{C}}$=$\{i \mid \hat{e}_{ij},\ j \in \{n - [\frac{W}{2}], ..., n + [\frac{W}{2}]\}\}$
  for k=$n - [\frac{W}{2}] : n + [\frac{W}{2}]$ **do**
    if $Col_k(\hat{E}) == \mathbf{0}$ **then**
      $\hat{\mathcal{C}}$=$\hat{\mathcal{C}} \cup \{\theta\}$
    end if
  end for
  for $q \in \mathcal{C}$ **do**
    $\hat{q}$= $\text{argmin}_{j\in\hat{\mathcal{C}}}(d_{qj} \times \rho_{qj})$ // note the exceptions made in Equations (4.10)–(4.12)
    $\mathcal{M}$=$\mathcal{M} \cup \{(q,\ \hat{q})\}$
  end for
  $\hat{\mathcal{C}}_u$=$\{i \mid i \in \hat{\mathcal{C}}, (j,\ i) \notin \mathcal{M}, j \in \mathcal{C}\}$
  σ_W=$\sigma_W + \dfrac{|\hat{\mathcal{C}}_u|}{|\hat{\mathcal{C}}|}$
  `RMSE`=`RMSE` $+ \sum_{(i,\ j)\in\mathcal{M}} d_{ij}^2$
  $\mathcal{S}_{\texttt{RMSE}}$=$\mathcal{S}_{\texttt{RMSE}} + |\mathcal{M}|$
  for $i \in \{1,\ 2,\ 3,\ ...,\ M,\ \theta\}$ **do**
    if $i \in \mathcal{C}$ **then**
      $\mathcal{S}_{\texttt{Recall}}$=$\mathcal{S}_{\texttt{Recall}} + 1$,
      $\mathcal{S}_{\texttt{Precision}}$=$\mathcal{S}_{\texttt{Precision}} + 1$
    else if $i \in \hat{\mathcal{C}}$ **then**
      $\mathcal{S}_{\texttt{Recall}}$=$\mathcal{S}_{\texttt{Recall}} + 1$
    end if
    `TP`=$\{i \mid (i,\ i) \in \mathcal{M}\}$,
    `FP`=$\{g \mid (g,\ i) \in \mathcal{M},\ g \neq i\}$,
    `FN`=$\{g \mid (i,\ g) \in \mathcal{M},\ g \neq i\}$
    `Recall`=`Recall` $+ \dfrac{|\texttt{TP}|}{|\texttt{TP}| + |\texttt{FP}|}$,
    `Precision`=`Precision` $+ \dfrac{|\texttt{TP}|}{|\texttt{TP}| + |\texttt{FN}|}$
    `Recall`=$\dfrac{\texttt{Recall}}{\mathcal{S}_{\texttt{Recall}}}$, `Precision`=$\dfrac{\texttt{Precision}}{\mathcal{S}_{\texttt{Precision}}}$
  end for
  $\overline{\sigma}_W$=$\frac{\sigma_W}{N}$, `RMSE`=$\sqrt{\frac{\texttt{RMSE}}{\mathcal{S}_{\texttt{RMSE}}}}$, `Recall`=$\frac{\texttt{Recall}}{N}$, `Precision`=$\frac{\texttt{Precision}}{N}$
end for

were excess number of such events. The intuition behind this definition is motivated by the case where all the synthetic sensors fire all the time; in this case, the SAS algorithm would always find, for any given sensor event in the real-world matrix E, a matching event from matrix $\hat{E}$. Although this case results in an RMSE equal to 0 and precision and recall both equal to 1, it also exhibits the maximum superfluous ratio ($\approx$1), implying that that synthetic sensor events in each column do not provide much information about agent traces.

$$\overline{\sigma}_W = \frac{1}{N} \sum_{n=1}^{N} \frac{|\hat{\mathcal{C}}_u|}{|\hat{\mathcal{C}}|} \tag{4.13}$$

Algorithm 3 (TSR) Finds a matching list for real-world temporal sensor readings (TSR) regarding a given window size and calculates metrics.

Inputs: **(1)** $E_{M\times N}$: Real-world sensor events (elements e_{ij}), **(2)** $\hat{E}_{M\times N}$: Synthetic sensor events (elements $\hat{e}_{ij}$), **(3)** W : Window size,
Output: **(1)** `RMSE`: A set of Root Mean Square Error for sensors, **(2)** `Recall`: A set of recall score for sensors, **(3)** `Precision`: A set of precision score for sensors,

$\mathcal{M}=\{\}$
for $m=0 : M$ **do**
  $\mathcal{C}=\{e_{ij} \mid i=m\}$
  $i=0$
  for $r \in \mathcal{C}$ **do**
    $\hat{\mathcal{C}}=\{\hat{e}_{ij} \mid i=m, j \in \{i-[\frac{W}{2}], ..., i+[\frac{W}{2}]\}\}$
    $\hat{r}=\text{argmin}_{j\in\hat{\mathcal{C}}}(|r-j|)$
    $\mathcal{M}=\mathcal{M} \cup \{(r,\hat{r})\}$
    $i=i+1$
  end for
  $\texttt{TP}=\{i \mid i>1, (i,\ i) \in \mathcal{M}\}$,
  $\texttt{FP}=\{g \mid (g,\ i) \in \mathcal{M},\ g>i\}$,
  $\texttt{FN}=\{g \mid (i,\ g) \in \mathcal{M},\ g<i\}$
  $\texttt{RMSE}=\texttt{RMSE} \cup \{\sqrt{\frac{\sum_{(i,\ j)\in\mathcal{M}} |i-j|^2}{|\mathcal{M}|}}\}$
  $\texttt{Recall}=\texttt{Recall} \cup \{\frac{|\texttt{TP}|}{|\texttt{TP}|+|\texttt{FP}|}\}$,
  $\texttt{Precision}=\texttt{Precision} \cup \{\frac{|\texttt{TP}|}{|\texttt{TP}|+|\texttt{FN}|}\}$
end for

Temporal Sensor Readings (TSR)

This metric evaluates the performance of our simulator, for each individual sensor. The $\texttt{ID}_{th}$ row of matrices E and $\hat{E}$ represent the sequences of events emitted by the ID real-world sensor and its simulated counterpart. We compare each row from matrix E against its counterpart from matrix $\hat{E}$ to measure the similarity of real-world and synthetic sensor readings over time. Similarly to the SAS algorithm, we use a windowing approach to mitigate the inherent uncertainty of the phenomenon.

Algorithm 3 illustrates the process of finding a matching list for m-th row in matrix E, R_m, from m-th row in matrix $\hat{E}$, $\hat{R}_m$. The TSR algorithm compares each row of matrix E against its matching list in terms of root mean squared error (RMSE), recall, and precision, given different window sizes.

4.4 Experimental Evaluation

In order to evaluate the SIM_{sis} simulator, we use the data set captured in [94] in our Smart Condo™ $\sim$70 m^2 apartment. Smart Condo™ is a one-bedroom apartment unit equipped with several sensors, including motion sensors, and Bluetooth Low Energy (BLE) beacons. The condo is designated for health-related studies from different disciplines such as: medicine, rehabilitation, and computer science. In the study conducted by Mohebbi et al. [94], participants, either alone or in pairs, performed a sequence of activities of daily living (daily objectives in Figure 4.1). Table 4.1 shows two scripted sequences of activities of daily living and their estimate time of completion; script one was given to solo participants, and paired participants each were given one of the scripts. It is worth mentioning that paired participants were asked to perform overlapped activities together.

Throughout each session, sensor data from 31 BLE beacons and 14 motion sensors are captured and stored. In addition, the location of the ground truth was produced by manual annotation of video footage from cameras in the condo. Table 4.2 sum-

Table 4.1: Scripted sequence of activities of daily living performed by participants in [94] followed by their estimated time of completion.

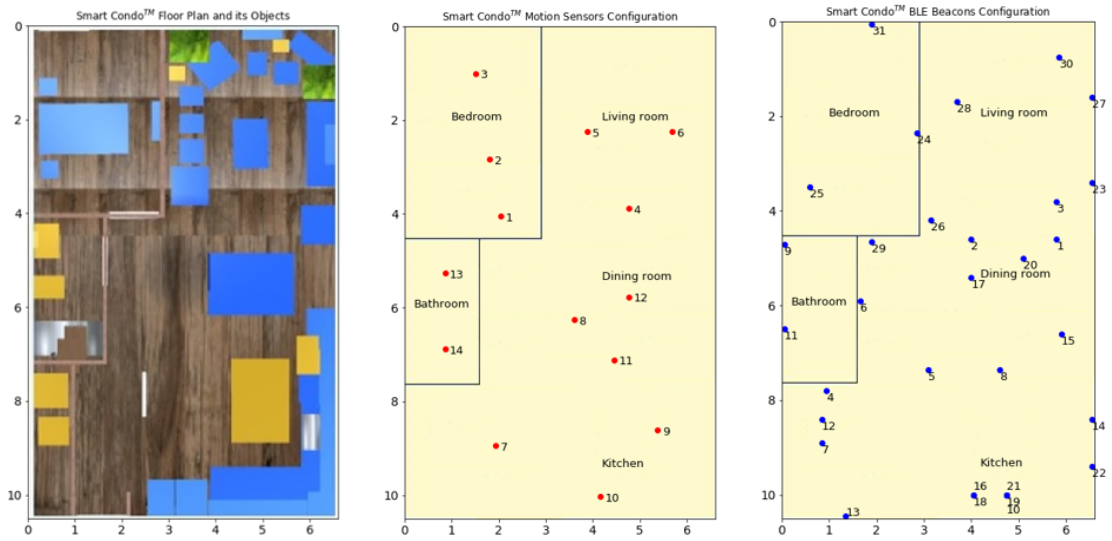
Scripts	**Activities**
Script 1	1) Exercise (30 m) 2) Use Toilet (1 m) 3) Change Cloths (1 m 30 s) 4) Take Bath (2 m) 5) Wash Hands (30 s) 6) Fill Kettle (15 s) 7) Make Tea (3 m 30 s) 8) Cook Eggs (5 m 30 s) 9) Setup Table (10 s) 10) Eat Meal (7 m) 11) Take Medicine (30 s) 12) Wash and Rinse Dishes (3 m) 13) Drain Water and Wipe Cabinets (1 m) 14) Broom Kitchen (1 m) 15) Broom Dining Room (1 m 30 s) 16) Do Laundry (2 m) 17) Iron Shirt (10 m) 18) Work with Tablet (30 m) 19) Watch TV (5 m)
Script 2	1) Use Toilet (1 m) 2) Change Cloths (1 m 30 s) 3) Take Bath (2 m) 4) Wash Hands (30 s) 5) Work with Tablet (30 m) 6) Fill Kettle (15 s) 7) Make Tea (3 m 30 s) 8) Cook Eggs (5 m 30 s) 9) Setup Table (10 s) 10) Eat Meal (7 m) 11) Take Medicine (30 s) 12) Wash and Rinse Dishes (3 m) 13) Drain Water and Wipe Cabinets (1 m) 14) Broom Kitchen (1 m) 15) Broom Dining Room (1 m 30 s) 16) Exercise (30 m) 17) Do Laundry (2 m) 18) Iron Shirt (10 m) 19) Watch TV (5 m)

Table 4.2: SIM_{sis} evaluation testbed.

Date	**#Participants**	**#DataPoints**	**Duration**	τ
28 June—1st	1	2042	01:41:00	3 (sec)
28 June—2nd	1	2229	01:51:21	3 (sec)
4 July—1st	2	2936	02:26:42	3 (sec)
4 July—2nd	2	2381	01:58:57	3 (sec)

marizes our testbed used in this chapter.

Recalling our methodology (Figure 4.3), we produced a 3D model of the Smart Condo™ including its objects in BIM Editor. Figure 4.3 shows the 3D representation of Smart Condo™ floor plan and its objects in BIM_{3D}^{Sim} (Figure 4.3a), the location of the motion sensors (Figure 4.3b) and the BLE beacons (Figure 4.3c) in our testbed. The indoor space has five rooms, *i.e.* kitchen, dining room, living room, bedroom, and bathroom. The location of sensors from both types, alongside their coverage area are stored in our sensor model. We test our methodology for the four sessions shown in Table 4.2 to produce synthetic sensor events. We ran simulations for each session of the dataset 10 times borrowing the region of similarity idea [69], *i.e.* given any



(a) 3D model of the Smart Condo™ in BIM_{3D}^{Sim}, and its objects.

(b) Motion sensor configuration in our testbed; each with an ID beside it

(c) BLE beacon configuration in our testbed; each with an ID beside it.

Figure 4.3: Our testbed for SIM_{sis} evaluation and motion sensors and Bluetooth Low Energy (BLE) beacons configurations.

destination point, virtual agents can randomly select a point inside a circle with some radius around the destination. We set the radius to 1 m and report the average result.

The real-world sensor events have many outliers, *i.e.* false readings, that need to be removed. This is due to the fact that motion and beacon sensors are sensitive to environmental parameters such as light, noise from appliances, interference from wireless networks, etc. Therefore, we detect and remove the outliers before our evaluation.

Outlier Removal: We utilize real-world agents trace, $\mathcal{A}$, in order to detect and remove outlier data points, from real-world sensor events. The objective was to compute a matrix, M, used as a mask, such that we can obtain "cleaned" sensor events by calculating Hadamard product [62] in Equation (4.14).

$$E'_{M\times N}{=}E_{M\times N} \circ M_{M\times N} \tag{4.14}$$

where E' is the matrix representation of the cleaned real-world sensor events. To achieve this, we calculate pairwise Euclidean distance of the real-world agents trace, $\mathcal{A}$ and

sensors location, S (*i.e.* S=$\{\{x,y\}^i | i \in \{1,2,3,...,M\}\}$) in Equation (4.15), and then subtract the radius of circles circumscribing sensors' coverage area (denoted by vector R) from the result in Equation (4.16).

$$D=|S^T - L| \tag{4.15}$$

$$B=diag(RR^T) \; . \overrightarrow{1} - D \tag{4.16}$$

Based on matrix B, we develop matrix M as follows:

$$M=\begin{cases} 0 & \text{if } B_{ij} < 0 \\ 1 & \text{if } B_{ij} >= 0 \end{cases} \tag{4.17}$$

Every column in matrix M is a "gate" letting sensors pass their readings to the corresponding column in matrix E'. However, this process is imperfect due to *uncertainty type 1*. Therefore, they are usually slightly different from each other. This phenomenon raises issues in aligning the agent traces with respective sensor events. To mitigate this issue, we use a windowing approach to obtain matrix M' as follows:

$$M'=\begin{cases} 0 & \text{if } M_{i(j-\lambda:j+\epsilon)}=0 \\ 1 & \text{if } M_{i(j-\lambda:j+\epsilon)}=1 \end{cases} \tag{4.18}$$

where λ and ϵ are our (asymmetric) window sizes (different from the one we defined for Algorithms 2 and 3) from left (prior) and right (future) sides, respectively. That is, λ is responsible to keep a history of sensor readings, whereas ϵ considers near future sensor readings. The λ and ϵ values should be assigned depending on τ. Finally the "cleaned" real-world sensor events can be represented as Equation (4.19). Our SAS and TSR algorithms use E' instead of E in our experimental evaluation.

$$E'_{M\times N}=E_{M\times N} \circ M'_{M\times N} \tag{4.19}$$

4.4.1 Example

We demonstrate our methodology with a simple example. Consider a simple world consists of seven discrete "cells" (all the cells are part of a room) and five motion sensors (Figure 4.4).

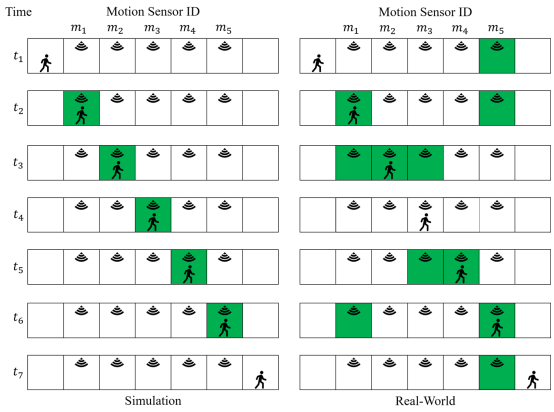


Figure 4.4: A simple environment as an example for our methodology. The green color shows which motion sensor gets activated in each step.

An agent starts walking from the left-most cell to the right-most cell, one cell at any time unit. The coverage area of each motion sensor is the whole cell that it is located on (so the radius of circles circumscribing sensors' coverage area is zero). The E and $\hat{E}$ matrices, and the vectors in Equation (4.2), sensors location S, and R are as follows:

$$\mathcal{A}=\{1,2,3,4,5,6,7\},\ S=\{2,3,4,5,6\},\ R=\{0,0,0,0,0\} \tag{4.20}$$

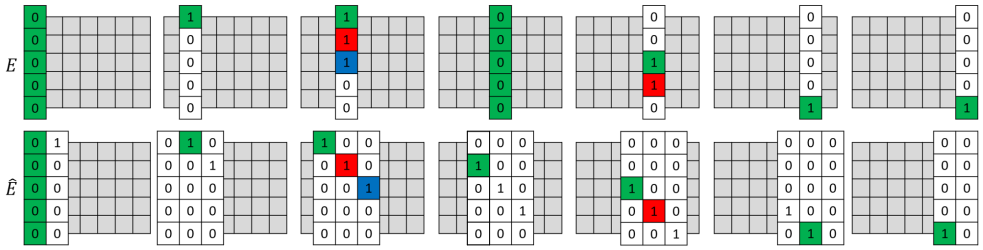
$$E_{5\times 7}=\begin{bmatrix}0&1&1&0&0&1&0\\0&0&1&0&0&0&0\\0&0&1&0&1&0&0\\0&0&0&0&1&0&0\\1&1&0&0&0&1&1\end{bmatrix}\quad \hat{E}_{5\times 7}=\begin{bmatrix}0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{bmatrix} \tag{4.21}$$

Using Equations (4.14)–(4.19), the outlier removal filters outliers in matrix E:

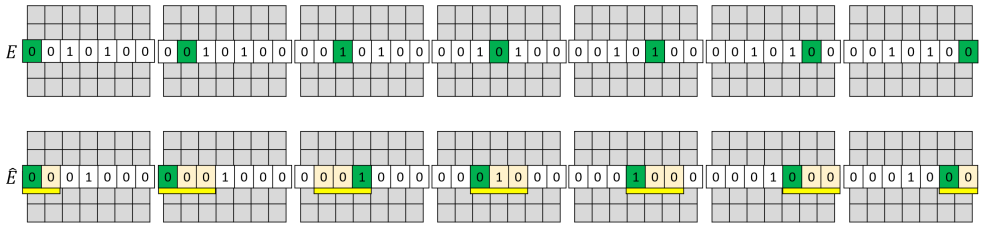
$$B_{5\times 7}=\begin{bmatrix}-1&0&-1&-2&-3&-4&-5\\-2&-1&0&-1&-2&-3&-4\\-3&-2&-1&0&-1&-2&-3\\-4&-3&-2&-1&0&-1&-2\\-5&-4&-3&-2&-1&0&-1\end{bmatrix}\quad M_{5\times 7}=\begin{bmatrix}0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{bmatrix} \tag{4.22}$$

$$M'_{5\times 7}=\begin{bmatrix}1&1&1&0&0&0&0\\0&1&1&1&0&0&0\\0&0&1&1&1&0&0\\0&0&0&1&1&1&0\\0&0&0&0&1&1&1\end{bmatrix},\ \lambda=1,\ \epsilon=1\quad E'_{5\times 7}=\begin{bmatrix}0&1&1&0&0&0&0\\0&0&1&0&0&0&0\\0&0&1&0&1&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&1\end{bmatrix} \tag{4.23}$$

Algorithms 2 and 3 compute sequence and reading matchings using columns and rows of matrices E' and $\hat{E}$, respectively. Figure 4.5 shows finding a matching for SAS



(a) Algorithm 2 execution example for SAS. Colors show the algorithm's matching. Note that $Col_1(E) = \theta$ is matched with $Col_1(\hat{E}) = \theta$. However, $Col_4(E) = \theta$ is inevitably matched with sensor m_2 from $Col_3(\hat{E})$. Colors show the matching pairs in each iteration of the algorithm.



(b) Algorithm 3 execution example for TSR. The yellow ribbon shows the window time frame and green shows the algorithm's matching.

Figure 4.5: Algorithms 2 and 3 execution examples.

with window size equal to 2, and finding a matching for sensor m_3's TSR for window size equal to 2 (The λ and ϵ are both equal to 1).

4.4.2 Agent Traces Validation Results

First, we compare synthetic agent traces against their real-world counterparts using the dynamic time warping (DTW) method. Figure 4.6 shows the DTW cost matrices for each pair of synthetic and real-world agent traces. Each element of the cost matrix, $(i,\ j)$ (which is equal to $C_{DTW}(\mathcal{A}_i,\ \hat{\mathcal{A}}_j)$), shows the accumulated cost of an optimal warping path starting at lower left corner, $(\mathcal{A}_1, \hat{\mathcal{A}}_1)$, and ending at $(i,\ j)$. Figure 4.6 shows the optimal warping path (black solid path) in each matrix for $(\mathcal{A}_N, \hat{\mathcal{A}}_N)$. Therefore, the optimal warping path shown in each matrix is actually equal to $C_{DTW}(\mathcal{A},\ \hat{\mathcal{A}})$ defined in Section 4.3.3. Based on the DTW algorithm, the closer the path to a diagonal line, the more similar two traces are, and therefore the higher the quality of the simulation. Table 4.3 shows the similarity measure ($S(\mathcal{A}, \hat{\mathcal{A}})$) results

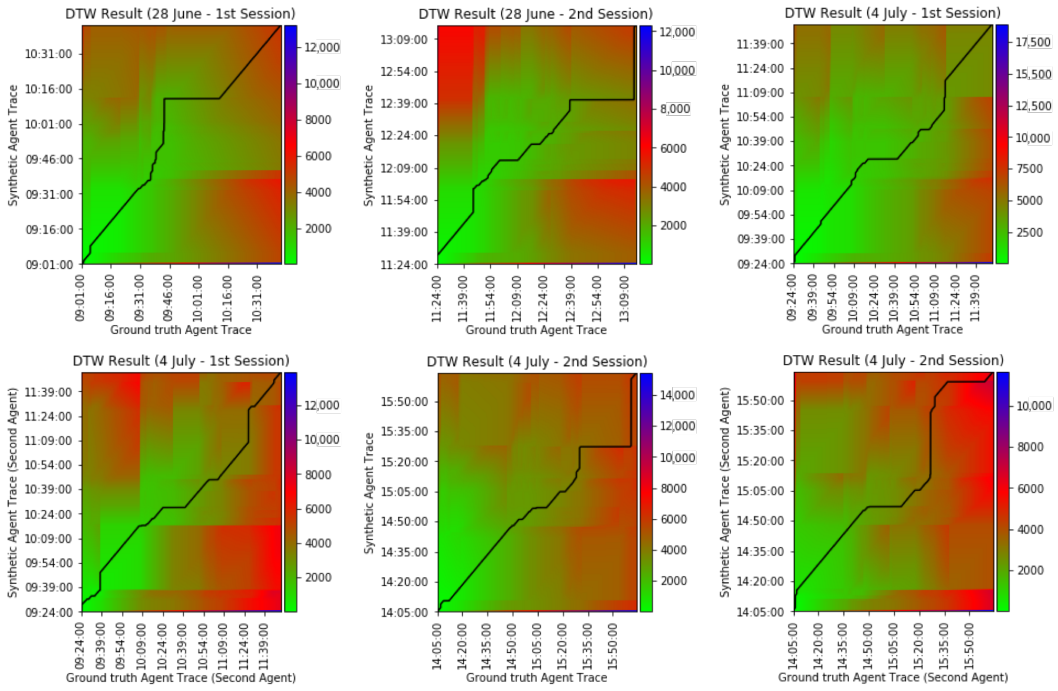


Figure 4.6: Dynamic time warping (DTW) results of synthetic agent traces compared to real-world agent traces (ground truth).

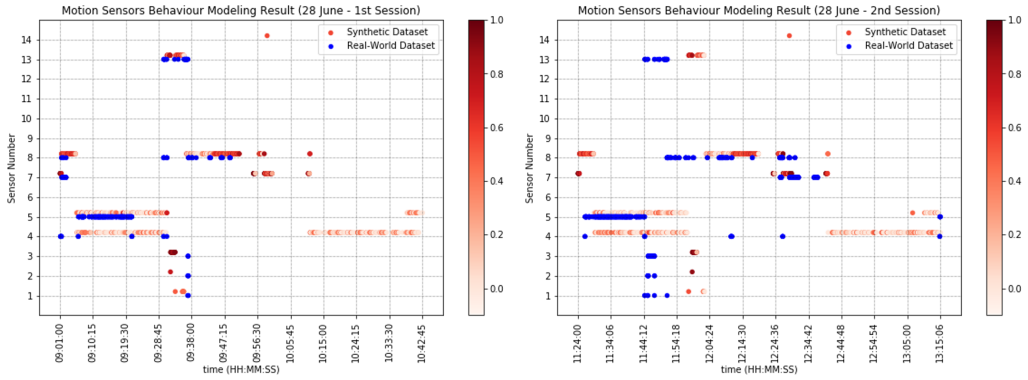
Table 4.3: Similarity measure obtained from DTW for real-world agent traces and synthetic agent traces. The average cost of the optimal warping path.

$S(\mathcal{A}, \hat{\mathcal{A}})$	**28 June—1st**	**28 June—2nd**	**4 July—1st (agent 1)**	**4 July—1st (agent 2)**	**4 July—2nd (agent 1)**	**4 July—2nd (agent 2)**
	78.77%	85.61%	86.54%	86.90%	79.87%	78.56%
	Total similarity: $\mu = 82.70\%$, $\sigma = 13.57\%$					

for our testbed. In total, our agent model is able to replicate real-world agent traces from our testbed with the accuracy of $\mu = 82.70\%$ ($\sigma = 13.57\%$). Considering the fact that in the worst and best case scenarios, virtual agent would be 12.4 m and 0 m apart from the real agent, respectively. Our results indicate that, on average, the difference between virtual agent and real agent location was 1.7 m.

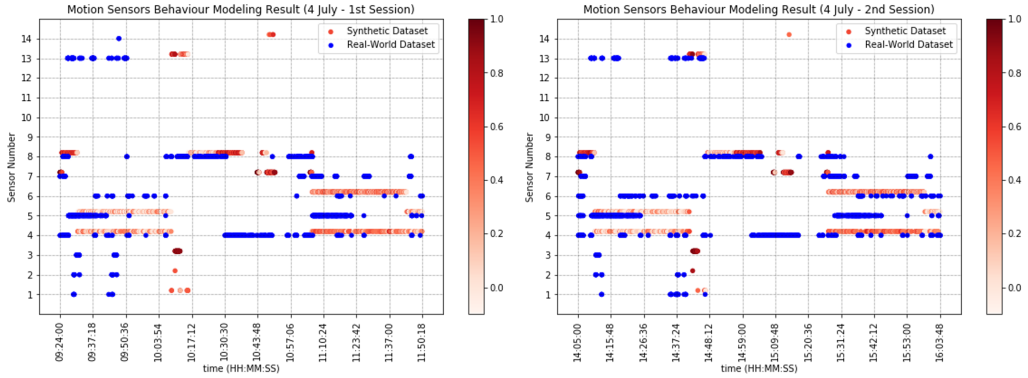
4.4.3 Sensor Events Validation Results

Figures 4.7 and 4.8 compare real-world and synthetic motion sensor events and beacon events, shown with blue and red dots, respectively. More red dots indicate that



(a) 28 June—1st session results

(b) 28 June—2st session results



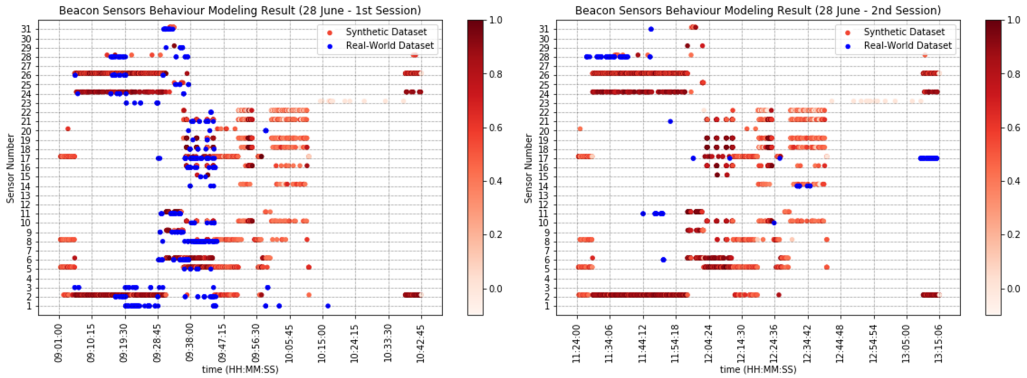
(c) 4 July—1st session results

(d) 4 July—2nd session results

Figure 4.7: Ground-truth sensor events (blue dots) versus synthetic (red dots) for motion sensors only. Red dots are plotted slightly higher that the line of the particular sensor to avoid occlusion.

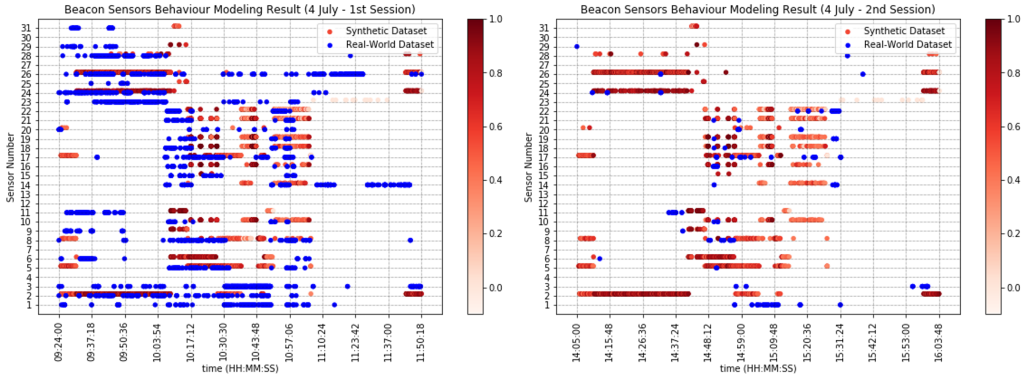
synthetic sensors fired more throughout the 10 times of simulation trials. Ideally, blue and red dots should align perfectly for each sensor. Nevertheless, there are differences in the synthetic sensor events and real-world sensor events in both sensor types, due to *timestamp ambiguity*, *i.e.* the synthetic sensors do not fire at the same time as their real-world counterparts, but also due to the agents' *behavioral ambiguity*, *i.e.* synthetic agents, even when they execute the activity script of their real-world counterparts, they may do so differently.

Take as an example motion sensor 8 from Figure 4.7. The sensor was placed above a table where all of the kitchen appliances were placed. The sensor readings throughout four sessions are slightly different from each other, indicating that there were stochastic behaviors in real-world agents, *e.g.* grabbing cookware and utensils from



(a) 28 June—1st session results

(b) 28 June—2st session results



(c) 4 July—1st session results

(d) 4 July—2nd session results

Figure 4.8: Ground-truth sensor events (blue dots) versus synthetic events (red dots) for for beacons only. Red dots are plotted slightly higher that the line of the particular sensor to avoid occlusion.

cabinet at the same time or separately, or toasting bread while at the same time making scrambled eggs. Moreover, in the real world, motion sensors are sensitive to motions within their coverage area (which is not considered in our sensor model), which implies that if a participant walks within the coverage area underneath a motion sensor and stays still, the motion sensor only triggers when the participant was walking and does not trigger when the participant was staying still (take as an example sensor 8 from Figure 4.7a). Furthermore, beacons do not always send signal strength proportional to their distance with a receiver; for example, beacons could send ≥ -70 dBm even when the receiver is farther than 1 m of proximity, or vice versa, the sensors could send < -70 dbm when the receiver is within 1 m of proximity.

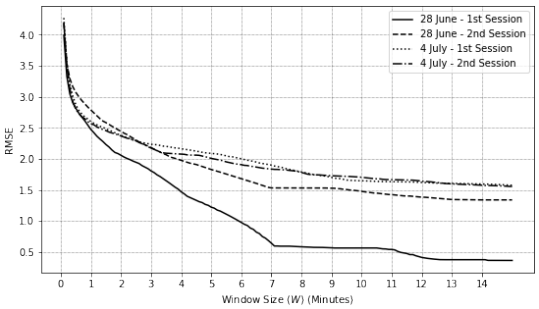
Sensors' Activation Sequence (SAS) Validity

For motion and beacon sensors, $e \in \{0, 1\}$, and $\hat{e} \in \{0, 1\}$ are elements of matrices E' and $\hat{E}$, respectively. By using these two matrices, Algorithm 2 can be executed for various window sizes. For each sensor type, the algorithm calculates a matching list $\mathcal{M}$ and accordingly, calculates RMSE, recall, precision, and superfluous ratio. We repeat this procedure for all of our data sets and various window sizes. The results of these metrics shown in Figure 4.9. In addition, we perform, separately, the same analysis for beacon sensors shown in Figure 4.10.

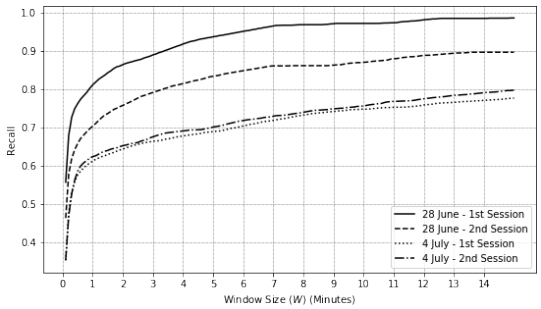
Based on Figures 4.9 and 4.10, as we increase the window size, RMSE, recall, and precision improve. It is safe to say that for large enough window sizes, these metrics converge because Algorithm 2 has plenty of options to choose from in order to create a matching list for a given sensors' activation sequence. However, larger window sizes results in higher superfluous ratio. For each window size, the precision and recall scores are calculated for each label, *i.e.* sensor number, and calculated their average score. Calculating precision for larger window sizes increases the chance of finding more false positives, which results in scores lower than smaller window sizes.

Temporal Sensors' Reading (TSR) Validity

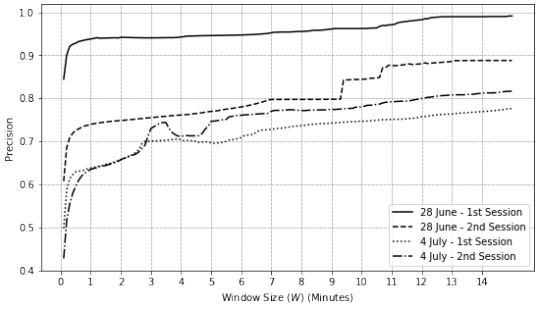
Our methodology uses Algorithm 3 with input matrices E' and $\hat{E}$ for beacon and motion sensors with different window sizes. For any calculated matching list, RMSE, precision, and recall metrics are obtained for both sensor types, shown in Figures 4.12 and 4.13, respectively, for our testbed. The figures show the better performance of the sensor model for larger window sizes. Notice that for both motion and beacon sensors is that for large enough window sizes, sensors can be categorized into two groups in terms of their slope in RMSE, precision, and recall. The first category, *i.e.* the sensors with steeper behaviors than the other category, and eventually convergence, are the sensors that the sensor model accurately simulates. Likewise, the second category is the sensors that our sensor model fails to simulate.



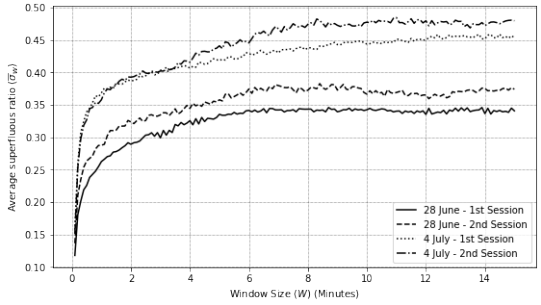
(a) RMSE value



(b) recall value

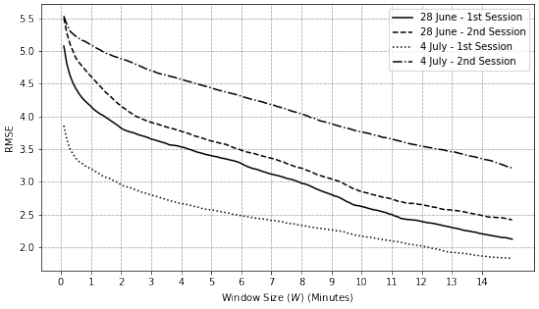


(c) precision value

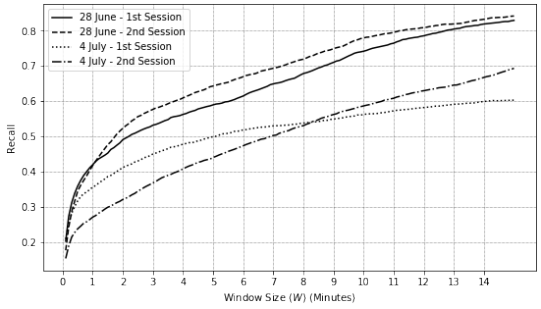


(d) superfluous ratio

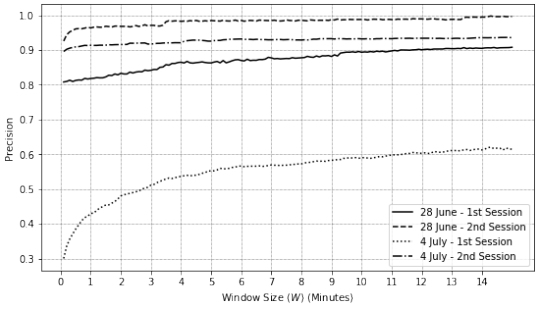
Figure 4.9: SAS analysis for motion sensors for different window sizes.



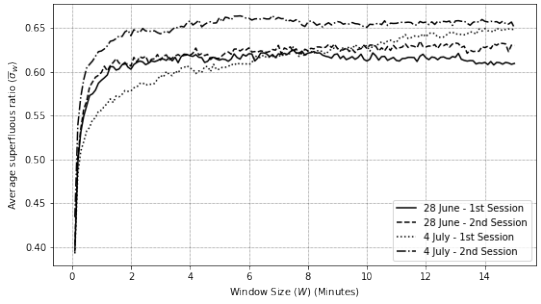
(a) RMSE value



(b) recall value



(c) precision value



(d) superfluous ratio

Figure 4.10: SAS analysis for beacon sensors for different window sizes.

4.4.4 Discussion

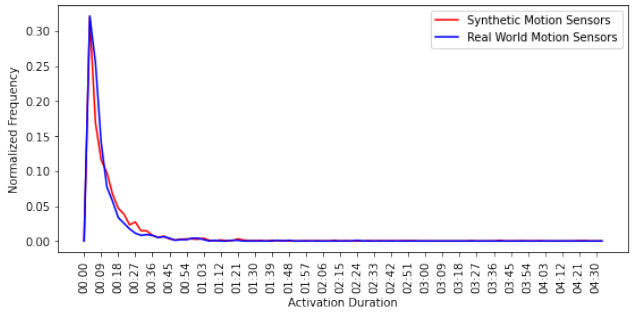
Our methodology models space, agents, and ambient sensor behavior that produces synthetic data set similar to real-world counterparts. The space model is a BIM model, a standard format produced by several architectural tools that accurately models the geometry of any intended indoor space. Our BIM_{3D}^{Sim} gets a BIM file, renders it to a 3D model. Users are able to add/define affordance properties for any object inside the model. Detailed geometry specifications of the model enables modeling agents and sensors with high fine degree of granularity.

The agent model is capable of generating synthetic agent traces based on a scripted sequence of activities, associated with daily objectives. Based on the results obtained from DTW, we observe that the model replicated real-world agent traces. In order to analyze the accuracy of the model, we obtain a "baseline model", wherein for each entry in real-world agent traces, we generate a random location. We obtain 20 "random agent traces" for each of the participants in our testbed (120 in total). On average, the similarity measure ($S(\mathcal{A}, \hat{\mathcal{A}})$) between real-world agent traces and random agent traces is $\mu = 73.62\%$ ($\sigma = 47.68\%$). We assume the accuracy from both baseline model and our agent model come from normal distributions, and calculate paired t-test. By conventional criteria, the difference of the two distributions is considered to be statistically significant ($p_{value} = 0.032$). However, there are flaws in interpreting the scripted activities into traces (*behavioral ambiguity*). There is a semantic gap in deciding how to perform some activities, even though following the daily objectives, e.g., one could debate how/where to perform specific activities like changing clothes, using a broom, or cooking. This usually happens to abstract activities, as compared to straightforward activities like use toilet, or take shower, which are activities unlikely to be performed differently every time, which would also cause different duration for performing those activities. That is why we see sometimes a lot of differences in Figure 4.6. These differences then adversely affect our sensor

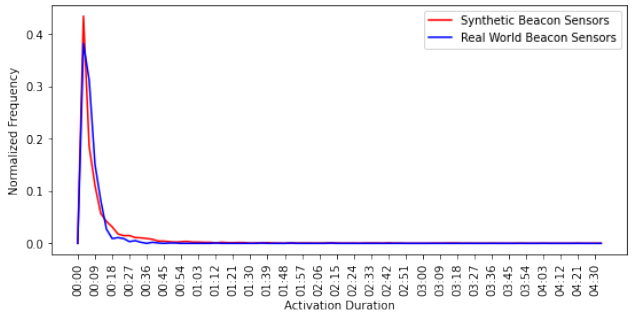
behavior modeling. Activities that the agent model fails to accurately model are: make tea, cooking, cleanup dining room, cleanup kitchen, iron shirts. On the other hand, activities that the agent model accurately models are: exercise, use toilet, take shower, washing hands, eat at dining table, doing laundry, watch TV, work with tablet. This analysis can be noted from Figure 4.6, 28 June—1st session, where at the beginning the participant performed exercise activity inside a marked area in living room. This is a straightforward activity because the participant stood on a spot and followed simple instructions shown in television. However, as it can be seen from all the figures of Figure 4.6, there are differences in the middle part of the figures. This is when participants were asked to cook. The cooking activity involved several actions, e.g., grab a pan, grab eggs, setup dining table, etc, which not only could introduce doubtfulness to participants, but also these activities could be performed differently every time by the participants. On the other hand, virtual agents followed the scripts.

The sensor model replicates real-world sensors accurately for large enough (in minutes scale) timestamp granularity. This means that the time intervals in which synthetic and real-world sensors were active should have similar distributions. Figure 4.11 shows the normalized frequency of activation duration for both synthetic and real-world motion and beacon sensors. We can observe that the most frequent activation duration in both sensor types is 6 s; and the frequency decreases for larger activation durations. It is worth mentioning that real-world motion and beacon sensors decay faster as activation duration increases; this is because real-world sensors might not get triggered constantly for a long time.

Our sensor model generates synthetic sensor events based on the synthetic agent traces. The validity of sensor model depends on the validity of the agent model, meaning that we can only validate sensor behaviors when synthetic agent traces perfectly match real-world agent traces. An obvious example for this can be seen from Figure 4.6—(28 June—2nd session), at 11:44:00 from x axis, where the path moved



(a) synthetic and real-world motion sensors



(b) synthetic and real-world beacon sensors

Figure 4.11: Normalized frequency of sensors activation duration.

vertically from 11:49:00 to 11:59:00 from y axis. This is due to the fact that real-world agent went to bathroom and used the toilet at 11:44:00, whereas the synthetic agent did the same at 11:59:00, so DTW algorithm warped the time in order to match the "use the toilet" activity in both traces. That is why we can see from Figure 4.7 that motion sensor number 13, which is placed on the top of toilet activated after some delay. We can see from Figure 4.6 that there are differences, sometimes huge, in synthetic and real-world traces (*behavioral ambiguity*). Based on this reason and *timestamp ambiguity*, regardless of the huge differences, and our Smart Condo™ application, we could choose window size equal to 7 min if we wish to mitigate the issues to some extent. Therefore, we report in Tables 4.4 and 4.5 the performance of our sensor model in terms of SAS and in Table 4.6 the performance of the model in terms of TSR.

Table 4.4 shows that we had 1.44 m average error in activation sequence for mo-

Table 4.4: SAS analysis for motion sensors with $W = 7$ min.

Motion Sensors Activation Sequence			
RMSE	Precision	Recall	Superfluous ratio
1.44 $\sigma = 0.25$	0.81 $\sigma = 0.007$	0.82 $\sigma = 0.009$	0.40 $\sigma = 0.002$

Table 4.5: SAS analysis for beacon sensors with $W = 7$ min.

Beacon Sensors Activation Sequence			
RMSE	**Precision**	**Recall**	**Superfluous ratio**
3.18 $\sigma = 0.38$	0.83 $\sigma = 0.02$	0.60 $\sigma = 0.006$	0.63 $\sigma = 0.0002$

tion sensors, which is small in our application. In addition, the average superfluous ratio is 0.4. However, for beacon sensors (Table 4.5), the error is as high as 3.18 m, with average superfluous ratio equal to 0.63. This behavior, *i.e.* high RMSE value and relatively low average superfluous ratio, means that beacon sensors had moderately different activation sequences. The recall score for beacon sensors also confirm this analysis. This might be due to two reasons: the noisy behavior of beacon sensors, and/or the consequence of the -70 dBm threshold to 1 m distance.

In terms of TSR, Table 4.6 shows that in average we had 0.31 and 0.32 sensor reading error (RMSE values for TSR) for motion and beacon sensors respectively (since they are binary sensors, the maximum is 1.0 and the minimum is 0.0 and baseline is 0.5). We should mention that the sensor model failed to model several beacon sensors due to their erratic behavior, as a result, the utilization of beacon sensors is discarded in this thesis. For example, in Figure **??** (28 June—1st Session) in the Appendix, there are several beacon sensors with rapid decrease in RMSE value, but others stay above 0.2 even with large window sizes. For this reason, we remove, as outliers, sensor behaviors with RMSE > 0.8 in Table 4.6. Precision and recall scores for both sensor types show high performance of our sensor model in modeling

Table 4.6: TSR analysis for $W = 7$ min.

	Temporal Motion Sensor Readings			**Temporal Beacon Sensor Readings**			
Session	**RMSE**	**Precision**	**Recall**	**RMSE**	**Precision**	**Recall**	**Sensor Outliers Removed**
28 June—1st	0.28 σ=0.14	0.77 σ=0.09	0.77 σ=0.09	0.23 σ=0.09	0.84 σ=0.04	0.84 σ=0.04	B1 B4 B9 B13 B17 B18 B20 B21 B22 B25
28 June—2nd	0.25 σ=0.11	0.81 σ=0.07	0.81 σ=0.07	0.39 σ=0.09	0.74 σ=0.05	0.74 σ=0.05	B5 B13 B20
4 July—1st	0.36 σ=0.10	0.75 σ=0.07	0.75 σ=0.07	0.35 σ=0.06	0.80 σ=0.03	0.80 σ=0.03	B0 B22 B26
4 July—2nd	0.32 σ=0.08	0.80 σ=0.04	0.80 σ=0.04	0.30 σ=0.07	0.82 σ=0.03	0.82 σ=0.03	B0 B4 B5 B7 B9 B15 B16 B17 B20 B21 B23 B25 B27
Average	0.31 σ=0.11	0.78 σ=0.07	0.78 σ=0.07	0.32 σ=0.08	0.80 σ=0.03	0.80 σ=0.03	

sensors in terms of TSR.

Our analysis shows if we assume that synthetic agent traces match real-world agent traces perfectly (not having *behavioral ambiguity*), then our sensor model is externally valid. However, we assumed that coverage area of sensors are circular. Although this representation is a generalized definition for ambient sensors, and IS sensors, e.g., pressure sensors, sensor model also should be capable of modeling other sensor types like temperature sensors, CO_2 sensors, or smart objects like wearable technologies. Then the agent model could be further extended in order to generate corresponding traces to cover more smart indoor space applications, e.g., fall detection, interaction scenarios for energy consumption analysis, etc.

We intend to resolve the *behavioral ambiguity* in the future, *i.e.* fill the semantic gap in interpretation of performing activities from the perspectives of real and artificial agents. One possible solution is to simply model agents with different characteristics, *e.g.*movement and ability in performing activities. A more sophisticated

approach is to train a generative model in order to produce realistic agent traces, while allowing re-sequencing of actions to some extent.

We also intend to simulate actuators in SIM_{sis} toolkit, in order to support simulating wider range of smart indoor space applications.

4.5 Conclusions

Smart homes and buildings are a very active topic of research, with a variety of applications, from ambient-assisted living, to telecare, to occupancy analysis for energy management, relying on sensor data to provide comfort and safety to the people living and working in them. Key to the effectiveness of these applications is the proper configuration of the sensors embedded in the space, but finding a satisfactory configuration is labor-intensive, costly, and time consuming.

In this chapter, we described the SIM_{sis} toolkit, a simulator for indoor smart spaces, that makes the following important contributions to the state-of-the-art. First, it incorporates a high-quality model of the space, relying on BIM in the IFC format, the de-facto representation standard of building information models. IFC enables the accurate specification of the space 3D geometry and the furnishings and objects in it. SIM_{sis} augments the IFC building model with specifications of the affordances of the objects in the space, so that virtual agents, given a set of objectives, move through the space and interact with the objects in it to accomplish their goals. Second, it includes a sensor-modeling-and-simulation component that realistically models sensors based on their type, location, and coverage area and simulates their event firing considering an increasing level of noise in the periphery of their coverage area.

We argue that an informative comparison between a simulation and the corresponding real-world activity should involve three dimensions of analysis.

1. Virtual agents should behave similarly to their real-world counterparts. SIM_{sis} adopts dynamic time warping (DTW) as a measure of how close the sequence of the

virtual-agents basic actions are to those of the real agents.

2. As the agents move and act within the space, the simulated sensors deployed in the space should behave (fire or not fire) as their real-world peers. The sensor activation sequence (SAS) metric was conceived for this purpose.

3. Finally, the sensor events emitted by a simulated sensor and its real-world counterpart over time should be similar. The temporal sensors' reading (TSR) metric captures this type of similarity.

We have evaluated the validity of SIM_{sis} simulations by comparing the synthetic traces it produced when configured with a model of the space, agent and sensors of a real-world study we conducted in the Smart Condo™. Our results demonstrate that SIM_{sis} accurately simulates agents' basic activities, *i.e.* moving, sitting, and standing close to objects to interact with them, but is not aware of abstract activities, *i.e.* cooking or sweeping the floor. The sensor-simulation component performed well in replicating the behavior of motion sensors but needs to be improved with respect to simulating beacons.

Our results demonstrate the potential of our SIM_{sis} toolkit, and the simulation methodology it supports, for generating realistic agent and sensor-event traces to support the development of SIS applications in smart buildings.

4.6 Appendix

Figure 4.12 and Figure 4.13 show TSR analysis of motion and beacon sensors, respectively. One can see that for at most 7 minutes window size, almost all of the motion sensors' metrics reaches their best value. Nevertheless, several beacon sensors require larger, sometimes impractically large (*e.g.* B27 in 28 June - 2nd Session), window sizes.

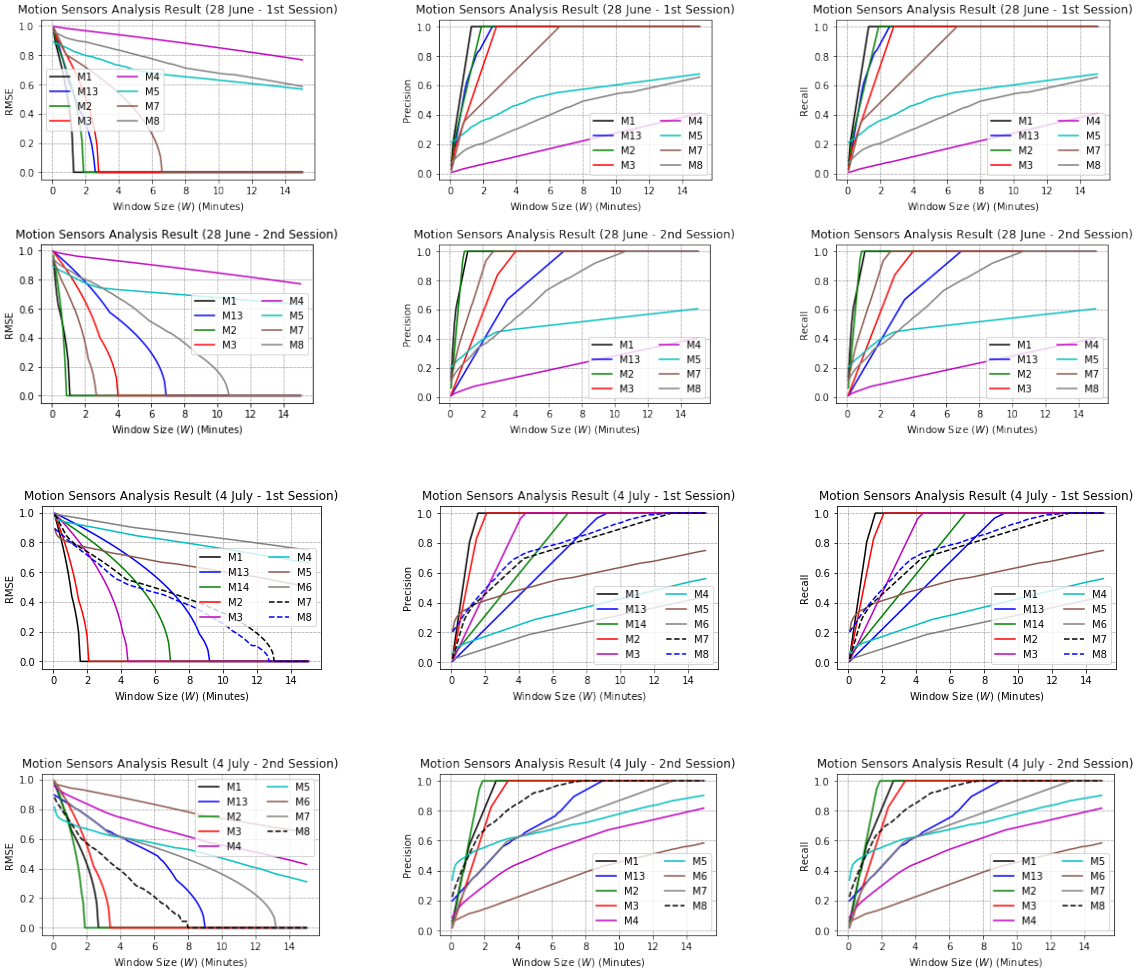


Figure 4.12: TSR analysis: (left column) RMSE value, (middle column) precision score, and (right column) recall score of motion sensors (shown in different colors) for different window sizes in each session.

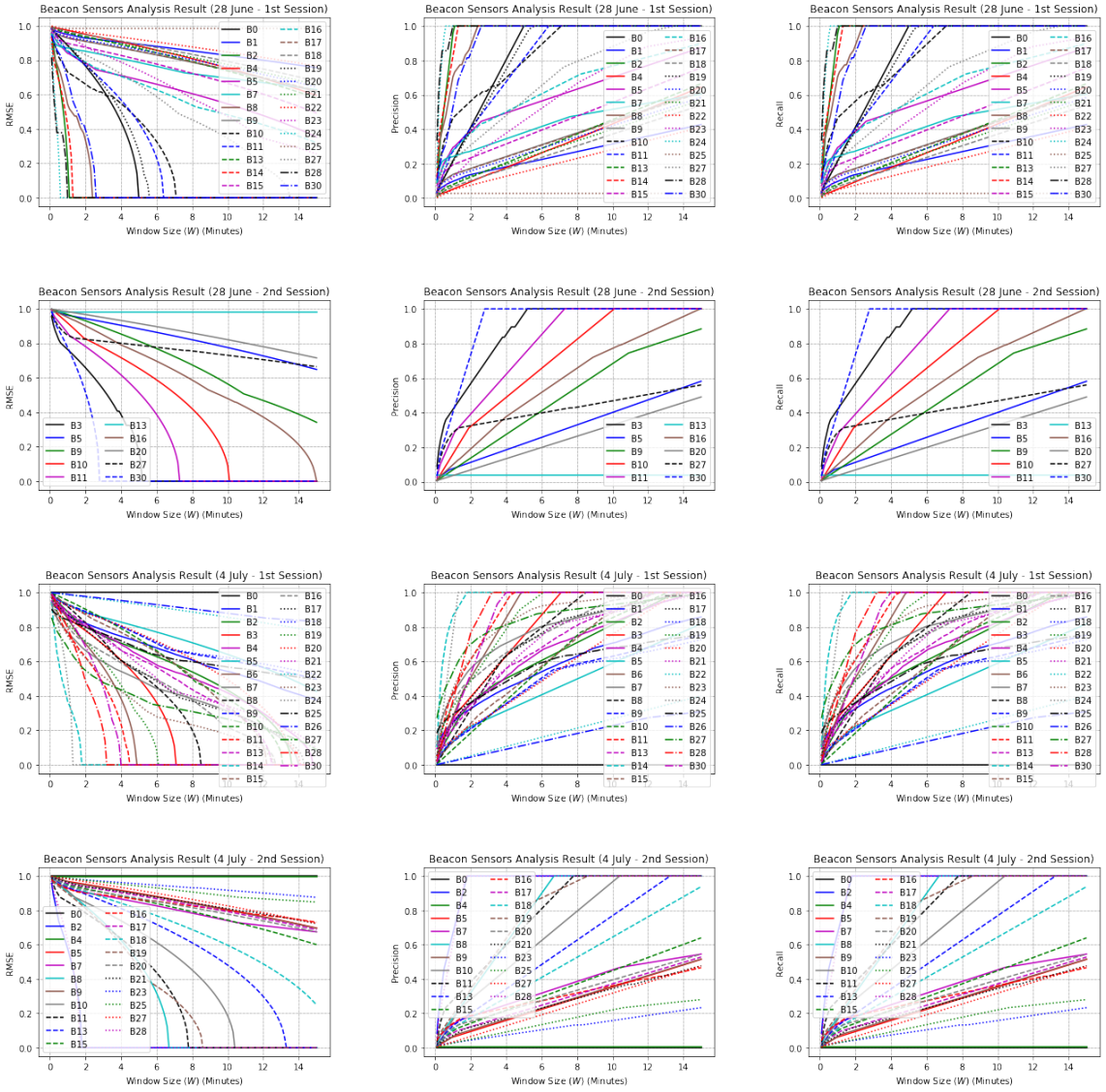


Figure 4.13: TSR analysis: (left column) RMSE value, (middle column) precision score, and (right column) recall score of beacon sensors (shown in different colors) for different window sizes in each session.

Chapter 5

Simulation-based Sensor Configuration Evaluation

In previous chapters, we showed that data-driven methods for evaluating sensor configurations could assist SIS applications' designers to make informed decisions. These methods analyze the performance of SIS applications in terms of their ability to estimate high-level contextual information, such as occupants' locations or activities.

On the other hand, we argued that the acquisition of the datasets is limited to their availability, flexibility and quality, motivating the development of a realistic SIS simulation methodology in Chapter 4.

Consequently, this chapter proposes a simulation-based component for evaluating different sensor configurations. Our component uses SIM_{sis} to generate synthetic datasets and fits a classifier to detect occupants' activities as contextual information. This chapter proposes using Bayesian optimization capable of finding a high-quality sensor configuration maximizing the ability to detect indoor activities. We show in two testcases that our proposed approach offers sensor configurations that are significantly better than the alternative methods in the literature.

This chapter focuses on R4 research question, specified in Section 1.2: A novel simulation-driven sensor configuration optimization component for SIS applications, considering the quality of detecting contextual information as the objective function.

5.1 Introduction

Intelligent indoor environments are equipped with wired or wireless sensors, *e.g.* cameras and motion sensors, to monitor activities, and physical and mental health of occupants. The configuration of these sensors (number, type, location, etc.), could affect the performance of applications that rely on the sensor data [114, 154]. For example, an optimized sensor configuration makes possible accurate activity recognition using the smallest number of sensors. But, due to the large size of the design space and the high cost of evaluating each configuration under normal occupancy conditions, finding the best sensor configuration via exhaustive search is prohibitive in practical applications.

Many efforts have been made to date to reduce the time complexity of finding a sensor configuration that includes a small number of sensors and satisfies various requirements. Specifically, several studies have focused on maximizing the area covered by the sensors [44, 58]. The main limitations of these approaches are that they assume all areas of the building are equally important and do not incorporate the activity recognition accuracy to identify the best sensor configuration. In the real world, some building spaces are rarely occupied, implying that covering the *active* regions is more important for activity recognition.

Another approach is to maximize sensor coverage while prioritizing the areas of interest in the building. To this end, several studies have taken advantage of contextual information about the building and its floor plan to obtain a set of points that might represent the location of occupants, and placed sensors such that these points are maximally covered [16, 76, 147]. But, it is generally assumed that all these points have the same importance, which could result in the misidentification of critical, yet rare human activities, such as bathing, in an aged-care monitoring application. Building upon this approach, the optimal configuration of omnidirectional motion sensors that are attached to the ceiling is found in [139] by maximizing the accuracy

of an activity recognition model while minimizing the number of deployed sensors, given the occupants' activities and trajectories. This approach requires collecting real data about activities and movement trajectories, which is expensive and challenging. Moreover, the evaluation of sensor configurations will be inaccurate when some activities are not present in the real-world dataset. We believe this problem can be addressed by generating realistic, synthetic traces. To simulate occupant activities in an arbitrary indoor environment and store the corresponding sensor readings, we need a high-fidelity simulation component that models the indoor environment, occupant presence and actions, and sensor readings.

The most notable techniques adopted in the related work to configure sensors in an indoor environment are greedy and evolutionary algorithms, in particular the genetic algorithm (GA) [139, 152, 158, 159, 149, 84]. In other domains, similar greedy and heuristic search-based methods have been proposed to find the optimal sensor configuration [64, 156, 155, 74]. Despite being powerful, these methods might not perform well in certain classes of problems because they merely rely on local information the samples provide from a function being optimized [96]. In contrast, the estimation of distribution algorithms (EDAs), *e.g.*Bayesian optimization (BO), use local information to acquire global information about the function, *i.e.* a probabilistic model (surrogate model) of the function. When this model accurately represents the function, it helps the optimizer do a more effective exploration/exploitation. The application of EDAs, notably BO, to optimizing the configuration of sensors in an indoor environment with respect to the performance of an activity recognition model is unexplored.

Figure 5.1 shows an example wherein GA and BO are applied to find the optimum value (maximum) of a function $f(x)$. As the figure shows, the GA utilizes the current observations to identify the next query point (2.9), which is probably close to the best-observed one (3). In contrast, the BO utilizes a probabilistic model of the function to identify the next query point (1.8), which is close to the optimum point (2). This

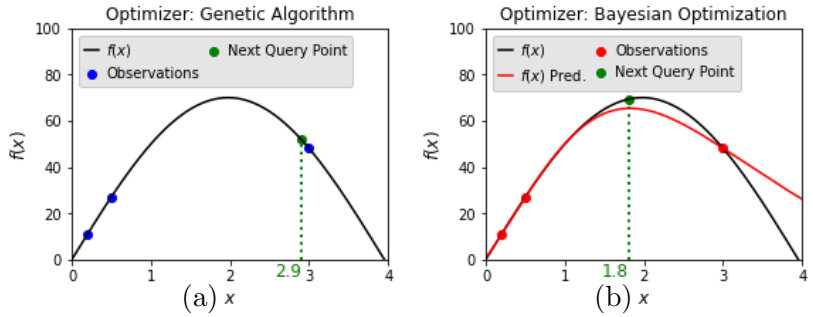


Figure 5.1: Comparison of (a) search-based methods, *e.g.* Genetic Algorithm, versus (b) Estimation of Distribution Algorithms (EDAs), *e.g.* Bayesian Optimization, in finding the maximum value of a function $f(x)$.

example shows that, when deploying a reasonable exploration/exploitation strategy, a probabilistic model of the function can better guide the optimizer.

This chapter presents a component based on Bayesian optimization and building simulation for efficient identification of a motion sensor configuration that better supports accurate activity recognition in an aged-care facility. To generate synthetic data representing different activities, we use a high-fidelity simulator developed in [54]. We show empirically that BO outperforms GA and greedy in two test buildings, which implies that EDAs are more appropriate than heuristic search-based methods in this particular problem. More specifically, our result indicates that the surrogate model of the objective function contains useful information in the context of intelligent indoor environments. Our contributions are threefold:

1. We cast the problem of configuring motion sensors in an indoor environment as a Bayesian optimization problem,

2. We propose a simulation-based assessment and optimization component that allows for using synthetic activities and movement trajectories, instead of real-world traces that are difficult to collect, and

3. We show the efficacy of the proposed methodology in finding different sensor configurations in two apartment buildings

5.2 Related Work

5.2.1 Manual Approach

A handful of studies propose quick intelligent indoor environments prototyping components, via simulation, for manual sensor configuration optimization [55, 150, 160, 31]. The main goal of these studies is to offer useful evaluation metrics based on the applications that these components support. The combination of the metrics and simulation environments provides inexpensive, repeatable and rapid prototyping of SIS applications. An example of these components is the work presented by Zhan and Haddadi [160], a component to evaluate sensor configuration for accurately identifying individuals in multi-occupancy scenarios. The component generates several synthetic occupants' location traces using real-world sensor readings. Then, the traces are used to simulate temporal sensor readings data of a specific sensor configuration. Each sensor reading is assigned to the occupant's ID whose location is closer to the location of that sensor than other occupants. The authors identified critical metrics for sensor configuration optimization, *i.e.* identification annotation accuracy, individual sensor's effect on the accuracy, and a tradeoff between detection area and the sensitivity of each sensor.

Nevertheless, decisions around sensor configuration depend on the SIS designers, which are time-consuming, burdensome and subjective. Instead, it is shown in [139, 147], that an optimization algorithm is faster, effortless, and can automatically suggest sensor configurations that are significantly more accurate than manual configurations. This chapter proposes applying an optimization algorithm for finding high-quality sensor configurations.

5.2.2 Area Coverage Approach

Fanti *et al.* [44] developed an integer linear programming (ILP) algorithm for maximizing area coverage of an indoor environment while minimizing the number of sensors

used. The authors demonstrated that their method guarantees full sensing coverage while maintaining a minimum number of sensors.

Gungor *et al.* [58] also utilized ILP for maximizing area coverage while considering sensor failures and minimizing the number of sensors used. Their algorithm produces sensor configurations that each point in the projected 2D grid on the indoor environment layout gets covered by at least one sensor; hence more robustness to sensor failures.

The area coverage approaches assume that all points in the grid have the same level of importance. Therefore, in their analysis, covering unimportant areas contributes equally to essential areas. In contrast, our methodology is based on a targeted coverage approach in which it uses contextual information to prioritize specific areas.

5.2.3 Targeted Coverage Approach

Wu *et al.* [152] proposed a sensor placement optimization method using a multi-objective genetic algorithm (NSGA-II) considering critical regions (as target points with higher priority for coverage) of the indoor environment. The optimization aims to maximize the target points coverage ratio while minimizing beacon sensors used. The authors validated their method by estimating the location of the target points and found that their algorithm produces sensor configurations that can accurately localize the critical regions. However, indicating critical regions requires expert knowledge, which is burdensome, subjective and might not capture essential contextual information, especially for designing new ageing in place settings because individuals' conditions might differ.

Laidi *et al.* [76] proposed an algorithm based on the mean-shift clustering that gets as input a set of target points (as training data) and denotes cluster means as possible sensor placeholders. Then a greedy algorithm decides whether or not to place a motion sensor in the placeholders. They showed that the outcome sensor configuration covers unseen target locations effectively. Nevertheless, the proposed method requires

collecting real-world occupancy data, *i.e.* their location, which is impractical and time-consuming for a given indoor environment. Moreover, the process requires intrusive sensors, *e.g.* cameras, and manual annotation of the collected data for the intended indoor environment.

Vlasenko *et al.* [147] developed a greedy algorithm that consumes synthetic occupancy data (their locations) and generates a motion sensor configuration that results in more accurate indoor localization of the occupants compared to random and manual designs. Their methodology uses synthetic data rather than real-world occupancy data. The data generation process requires specifying pairs of coordinates as start and destination locations. Then, it generates for each pair a path based on the A* algorithm. However, the simulated data only simulates the walking activity. In contrast, real-world occupants' data contain traces that occupants stay for some time in a specific location to perform an activity. Moreover, although localizing occupants is important for several intelligent indoor environment applications, its error tolerance is usually high (around a meter error up to a room-level error are negligible). Therefore, the more crucial metric for sensor configuration evaluation is the quality of an activity recognition task. Studies show that accurately recognizing occupants' activities is critical in most intelligent indoor environment applications, such as Activity of Daily Living (ADL) recognition [75], and space usage understanding [123]. Besides, understanding occupants' activities automatically provides information about their indoor location.

The most related research to ours is by Thomas *et al.* [139] which proposes a methodology to optimize a motion sensor placement in detecting activities while minimizing the number of sensors used. Their method uses a real-world dataset to generate synthetic occupant locations, including temporal sensor readings and occupant activities. Their methodology uses a Genetic Algorithm (GA) to propose different sensor configurations. Then, for each configuration, first, a sensor simulator uses the synthetic occupant's locations and generates synthetic sensor readings; Then,

an activity recognition model is trained and tested using the synthetic sensor readings and real-world occupant's activities. They showed that the GA algorithm could find a sensor configuration significantly more accurately than random, uniform and manual placements and the Hill-Climbing algorithm. Nevertheless, The synthetic dataset generation is driven by real-world sensor readings, and it assumes that the real sensors are error-free.

This chapter proposes a component that uses a realistic simulation methodology that generates synthetic occupants' temporal trace, *i.e.* location and activity, and its corresponding temporal motion sensors reading of an inputted sensor configuration, without needing real-world data. Moreover, the component uses BO as an example of EDAs to prove that the EDAs' probabilistic model accurately estimates the quality of different sensor configurations regarding an activity recognition's performance. This chapter aims to show that the probabilistic model provides global information that guides the optimizer towards significantly better sensor configurations than GA and greedy methods.

5.3 Methodology

We now present our simulation-driven motion sensor configuration component (depicted in Figure 5.2), which uses Bayesian optimization to find a configuration that maximizes the recognition accuracy for *activities of daily living* (ADL). We treat the simulation of the intelligent indoor environment as a black-box function that receives a candidate sensor configuration and outputs a stochastic, noisy observation. The proposed component consists of three main modules: 1) the simulator of smart indoor spaces (SIM_{sis}) developed in [54], 2) an activity classifier, and 3) a sensor configuration optimizer. The SIM_{sis} gets as input ADL plans of N occupants, the space model of the indoor environment, and a candidate sensor configuration. Given this input, it produces a synthetic dataset that consists of sensor readings (*i.e.* time-series generated by the sensors) and corresponding activities for each of the N occupants.

The activity classifier module trains a recognition model for ADL using part of this dataset, and evaluates its performance on the test sets. Finally, the sensor configuration optimization module iteratively uses the model performance to propose the next candidate sensor configuration that optimizes it.

5.3.1 Simulator of Smart Indoor Spaces

The SIM_{sis} is a simulator of smart indoor spaces introduced in [54]. It has been shown that SIM_{sis} can realistically model occupant's behavior and motion sensor readings. It gets as input a space model, daily plans of N occupants, and a candidate sensor configuration. The space model is the specifications of an indoor environment using Industry Foundation Classes (IFC) for Building Information Modeling (BIM) [13]. The IFC data model is an ISO International Standard, and an open architectural and building data description specification. The space model also includes object affordances, *e.g.*a couch affords the ability to be sat on. The daily plan of occupants models regular ADL, such as sleeping and cooking. Each sample from the model generates a slightly different daily plan in terms of ADL order and duration. Finally, the candidate sensor configuration describes the number and locations of motion sensors. The SIM_{sis} generates a synthetic dataset consisting of sensor reading and their corresponding activity for each of the N occupants in the following Equation 5.1:

$$\mathcal{D} = \{\{R_i^t, A_i^t\}_{t=1}^{T}\}_{i=1}^{N} \tag{5.1}$$

where R_i^t and A_i^t denote the i-th sensors reading (a vector of size D where D is the number of sensors used in the environment) and occupant's activity, respectively, at time t. An ideal motion sensor configuration[1] should result in a dataset where each activity has a distinct enough pattern in the space of sensor readings so that an activity recognition model can easily recognize it.

[1]We assume the motion sensors are omnidirectional and attached to the ceiling, so the sensor configuration reduces to the location of each sensor in a 2D space.

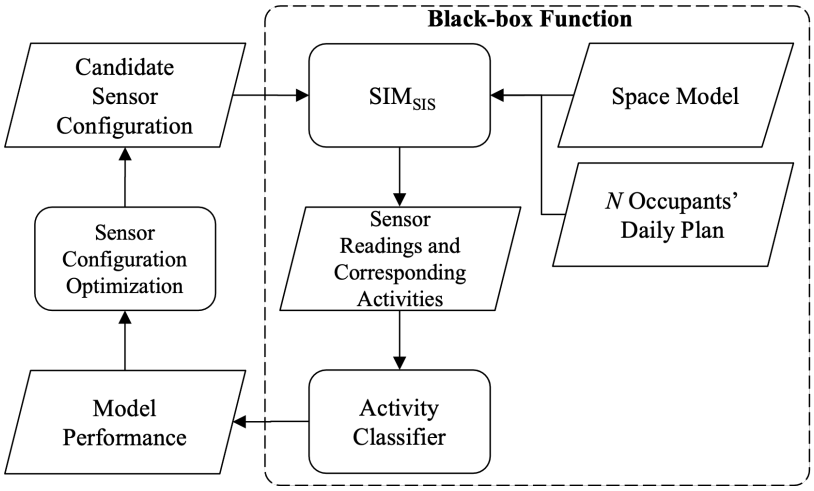


Figure 5.2: Sensor configuration optimization for accurate activity recognition via simulation.

5.3.2 Activity Classifier

The activity classifier gets as input the dataset $\mathcal{D}$ and applies the leave-one-out cross-validation method. Specifically, it considers the data that pertains to one occupant as the test set and the data of the remaining occupants as the train set. This process is repeated for every occupant to obtain different test sets, and the average performance is reported. We borrow the activity recognition system from the 'Center of Advanced Studies In Adaptive Systems' (CASAS) [32], which uses a Random Forest machine learning model (this recognition system is used by [139] too). For each test set, the activity classifier calculates the macro-averaged F1-score of the ADL and then outputs the arithmetic mean of the F1-scores (given below) as the model performance:

$$F^1 = \frac{1}{N}\Sigma_{i=1}^{N}\frac{1}{M}\Sigma_{j=1}^{M}\frac{tp_j}{tp_j + \frac{1}{2}(fp_j + fn_j)} \tag{5.2}$$

Here N is the number of occupants, M is the number of activities, and tp_j, fp_j, and fn_j are true positive, false positive and false negative of the j-th activity, respectively. Notice that F^1 is sensitive to both false negative and false positive. Therefore, this is a good performance measure, because both of these factors are important in the activity recognition task, and in particular for elderly monitoring systems where failing to alert the caregiver or issuing a false alarm could have dire consequences. Additionally, there

are rare and short, but precarious activities, such as taking a shower, that should be deemed as important as other activities when evaluating the activity recognition model. That is why we calculate the macro-averaged F1-score over the M activities in Equation 5.2.

5.3.3 Optimization Algorithm

The sensor configuration optimization module receives an observation, which is the performance of the activity recognition model expressed in (5.2), and iteratively proposes a candidate sensor configuration to optimize this performance measure. This optimization can be performed using the Bayesian approach (our proposed method described below), the genetic algorithm which is a popular evolutionary algorithm adopted in the literature [139], or a widely used greedy algorithm [74].

Problem Setup

The sensor configuration optimization module requires a representation of possible sensor locations to suggest the location of sensors effectively. The representation should maintain a specific granularity level to ensure the effectiveness of the optimization process. This is because high granularity increases the size of search space and its complexity, while using low granularity might result in finding a suboptimal sensor configuration. Most related work, including [139], creates a 2D grid (graph) based on the given floor plan, wherein each node indicates a possible sensor location in the environment. Similarly, we define a matrix $E_{H,W}$ to compactly represent the possible sensor locations:

$$E_{H,W} = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,W} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,W} \\ \vdots & \vdots & \ddots & \vdots \\ e_{H,1} & e_{H,2} & \cdots & e_{H,W} \end{bmatrix} \tag{5.3}$$

wherein each element $e_{i,j} \in \{0, 1\}$, with $0 < i \leq H$ and $0 < j \leq W$, indicates whether there is a sensor at location (i, j). Note that $H = \lceil h/\epsilon \rceil - 1$ and $W = \lceil w/\epsilon \rceil - 1$ where h and

w are the height and width of the indoor environment, respectively, and ϵ is the granularity in the 2D space, *i.e.* the spacing between consecutive rows and columns. Higher ϵ values indicate lower granularity, hence lower computation overhead.

Baseline 1: Genetic Algorithm

Our first baseline is the genetic algorithm (GA) used in [139] to find an optimal motion sensor configuration that enables accurate activity recognition. In this setting, each of the GA's chromosomes is a vectorized version of an instance of $E_{H,W}$. The GA starts with a population of 10 random sensor configurations and feeds each configuration to the black-box function shown in Figure 5.2 to obtain a fitness value. The fitness value of each chromosome is penalized by the number of sensors deployed (number of 1s in the chromosome) to prevent the algorithm from placing too many sensors (a budget constraint). The best 10% of the chromosomes survive to be the next generation. The GA then performs crossover and mutation to populate the next generation. First, the GA randomly chooses a pair of chromosomes from the best 20% of the population in the crossover function. Then, it generates two distinct random indices, excluding the first and last indices, to slice each chromosome into three sections and swap their middle sections, generating two new offspring. Finally, it mutates each newly generated offspring by randomly choosing a possible sensor location and flipping its value with a 0.005 probability.

Baseline 2: Greedy Algorithm

The greedy algorithm is a widely used heuristic search-based method for optimizing a black-box function [74, 153, 147]. We apply natural greedy sensor configuration choices, wherein the algorithm iteratively makes the best decision to optimize the entire problem. It starts with a configuration that contains no sensors, and iteratively adds a sensor to this configuration by finding the one-sensor configuration that yields the highest F1 score among the $W \times H$ possible configurations. The process continues

until it places the given number of sensors or the maximum number of black-box function queries is reached. Note the greedy algorithm may not find the global optimum as it is an iterative method and in each iteration it overlooks that black-box function observations are inherently noisy.

Bayesian Optimization

The Bayesian Optimization (BO) is a sequential search strategy for optimizing a black-box function. In our component, the goal of the Bayesian optimization is to maximize the black-box function f (depicted in Figure 5.2):

$$\max_{c \in S} f(c) \tag{5.4}$$

where S is the D-dimensional space of sensor configurations representing all possible configurations with D sensors, and $c = \{\{x \times \epsilon, y \times \epsilon\}_{d=1}^{D} \mid 0 \leq x \leq W,\ 0 \leq y \leq H\}$ represents a sensor configuration from this space. The BO method assumes that observations of the function f (Equation 5.2) are the results of a stochastic process independent across different observations. There are two sources of noise in our black-box function: 1) SIM_{sis} : although being high fidelity, it is a simplified model of the actual indoor environment. Thus, it does not necessarily capture all dynamics and uncertainties that exist in the real world; 2) the activity classifier: the random forest fits several decision tree classifiers on different randomly generated sub-samples of the given dataset. Thus, each execution might yield a slightly different performance on the dataset. BO approximates f using a probabilistic surrogate model, denoted $\hat{f}$, given a set of observations Z. The surrogate model is defined as:

$$\hat{f}(\hat{F^1}|c) = p(f|Z) \tag{5.5}$$

Equation 5.5 shows that given an input configuration c, $\hat{f}$ estimates F^1 (Equation 5.2), denoted as $\hat{F^1}$. We use the Probabilistic Random Forest (PRF) as the BO's surrogate model. At each iteration n, the surrogate model is indeed a prior over f given

observations $Z_{1:n}=\{(c_i, F_i^1)_{i=1}^n\}$ (accumulated observations from the first iteration to the n-th iteration).

The surrogate model, $\hat{f}$, is used to construct an acquisition function $\alpha(c; \hat{f})$, which decides the next candidate sensor configuration to evaluate in iteration $n+1$, denoted as c_{n+1}. We use the state-of-the-art expected improvement (EI) acquisition function [49]:

$$\alpha(c; \hat{f}) = \int_{F^{1*}}^{\infty} \max(\hat{F}^1 - F^{1*}, 0) \hat{f}(\hat{F}^1|c)\, d\hat{F}^1 \tag{5.6}$$

where F^{1*} is the best observed F^1 so far, *i.e.* $F^{1*}=\max\{F_1^1, ..., F_n^1\}$. The acquisition function can be used to find potentially better sensor configurations. By maximizing (5.6), we find:

$$c_{n+1} = \arg\max_c \alpha(c; \hat{f}) \tag{5.7}$$

which has the largest expected improvement. We remark that there are several alternatives to EI, such as probability of improvement, entropy search, and upper confidence bound [49]. However, comparing the performance of these acquisition functions is outside the scope of this thesis, which primarily focuses on investigating the efficacy of the surrogate model for optimizing the sensor configuration in an indoor intelligent environment.

After evaluating c_{n+1}, the corresponding observation, $z_{n+1}=(c_{n+1}, F_{n+1}^1)$, is appended to the past observations: $Z_{1:n+1}=\{Z_{1:n}, z_{n+1}\}$. Next, $Z_{1:n+1}$ is utilized to update the surrogate model $\hat{f}$, resulting in the posterior surrogate model which better approximates f. The posterior surrogate model is then used as a prior for obtaining c_{n+2}. The process continues for 1000 iterations and the best sensor configuration found is reported. Algorithm 4 summarizes the BO algorithm used in this chapter. We set the initial configuration (c_1) randomly and implement Algorithm 4 using the BO package from [82].

Algorithm 4 Bayesian Optimization Algorithm

Inputs: Number of sensors D, Granularity ϵ, Environment's width w, Environment's height h, and black-box function f

Output: An optimal sensor configuration c^*

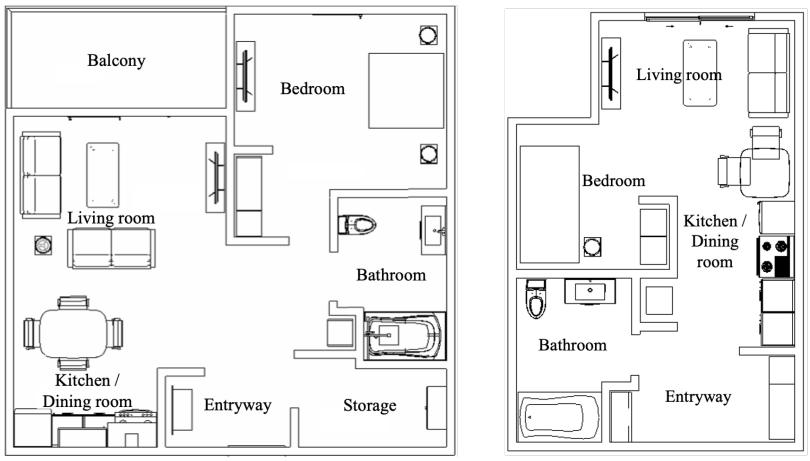
Initialize $c_1 = \{\{x \sim U([0, \lfloor \frac{w}{\epsilon} \rfloor]), y \sim U([0, \lfloor \frac{h}{\epsilon} \rfloor])\}_{d=0}^{D}\}$.
Initialize $F^{1*} = 0$
Let $Z_{1:1} = \{(c_1, f(c_1))\}$.
Let $\hat{f}(\hat{F^1}|c) = p(f|Z_{1:1})$.
for i = 2 : n **do**
    $c_i = \arg\max_c \alpha(c; \hat{f})$
    $F_i^1 = f(c_i)$
    $Z_{1:i} = \{Z_{1:i-1}, (c_i, F_i^1)\}$
    Update $\hat{f} : \hat{f}(\hat{F^1}|c) = p(f|Z_{1:i})$
    $(c^*, F^{1*}) = argmax_{F^1}\{Z_{1:i}\}$
end for
return c^*

5.4 Algorithm Evaluation

5.4.1 Case Study

We use two testbeds for our case study: an 8×8 (m^2) fully-furnished one-bedroom suite (denoted as T1) and a 8×5.2 (m^2) fully-furnished studio suite (denoted as T2) from the Lifestyle Options Retirement Communities [83]. The suites are designated for older adults needing independent living, assisted living, or memory care. We choose these suites as our testbeds because of their fundamental differences, such as layout and size, which could lead to various traces. The T2 causes different movement trajectories in some areas, such as the kitchen, because the entryway and bathroom are accessible only from the kitchen. The T1 causes slightly scattered movement trajectories and has no traces on the balcony. Therefore, they present unique challenges. Since there is no IFC file available for our case study, we use Autodesk Revit® [12] and BIMobject repository [19] to develop a digital twin of the indoor spaces in IFC format. Figure 5.3 shows the space models of our case study.

The case study includes five (simulated) occupants performing various ADLs separately (hence N=5 in Equation 5.1). Table 5.1 shows an ordered list of 23 detailed activities the occupants perform in 196 minutes in total (in Equation 5.1, A is the list of detailed activities and T=$\frac{196\times60}{3}$ because the data gets collected every 3 second).



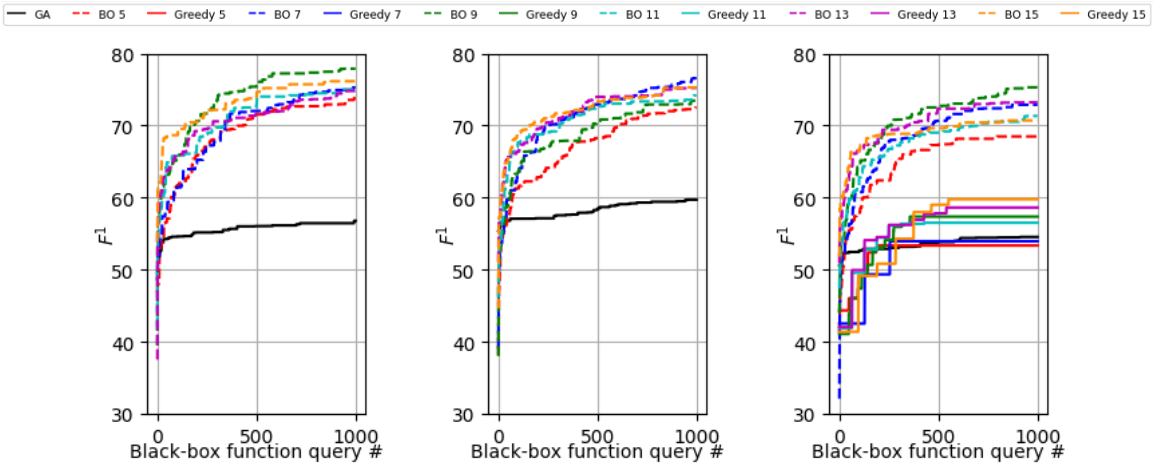
(a) Testbed 1 (T1). (b) Testbed 2 (T2).

Figure 5.3: The floor plan/space model of the two apartments: (left) an 8×8 (m^2) one-bedroom suite; (right) a 8×5.2 (m^2) studio.

Performing ADLs and few detailed activities (denoted by the same superscript in Table 5.1) can be swapped, giving slightly different activity plans.

5.4.2 Results

We compare motion sensor configurations found by using BO, GA, and greedy in our case study. We run each algorithm 5 times for each value of $\epsilon \in \{0.25m, 0.5m, 1.0m\}$. Given the range of motion sensors in the simulator (a circle with radius of 1 meter), these ϵ values are reasonable because they allow to potentially have some overlap between the areas covered by multiple sensors and maintain a clear line of sight despite the obstacles that exist in the environment. For greedy and BO, we repeat the process after setting the total number of sensors to 5, 7, 9, 11, 13, and 15. We cannot control the number of sensors placed by GA. Each algorithm can query the black-box function at most 1000 times. Tables 5.2 and 5.3 show the performance (Equation 5.2) of our method and the baselines in T1 and T2, respectively. In both T1 and T2, the greedy algorithm mostly exhausts the 1000 black-box function queries for ϵ=0.25 and 0.5. For example, in T1 when ϵ=0.5, 1115 function queries are needed to place 5 sensors, with this breakdown: 225, 224, 223, 222, 221 queries for putting



(a) $\epsilon = 0.25$ (T1) (b) $\epsilon = 0.5$ (T1) (c) $\epsilon = 1$ (T1)

(d) $\epsilon = 0.25$ (T2) (e) $\epsilon = 0.5$ (T2) (f) $\epsilon = 1$ (T2)

Figure 5.4: Average of the best performance of GA, greedy, and BO with different no. sensors across black-box function queries.

the 1st, 2nd, 3rd, 4th, and 5th sensor, respectively.

It can be readily seen that regardless of the number of sensors, BO significantly outperforms GA and greedy for all ϵ values[2], except the pairs that have the same superscript in Table 5.3. Notice that the best configuration found by BO for each ϵ value (printed in bold), significantly outperforms our baselines. Moreover, there are no significant changes in the performance of BO using different numbers of sensors. It achieves a good performance with as few as 7 sensors in T1.

[2]For each ϵ value, a two-tailed t-test is used to decide if there exists a significant difference (p<0.05) between the performance means of GA, greedy and BO for different amounts of sensors.

Table 5.1: Occupants' ADLs in our case study. ADLs and a few detailed activities with the same superscript (a or b) can be shuffled.

ADL	**Sequence of Detailed Activities (minutes)**
Bathing	Undress (5), Take a shower (15), Dress (6)
Hygiene	Use toilet (3), Wash hands (3)
Dining routine	Make tea (10), Grab ingredients (2), Fry eggs (10), Toast breads (5), Grab utensils (1), Eat (10), Take medicine (2), Wipe dining table^a (5), Wash dishes^a (3), Clean kitchen^a (5)
Brooming	Grab the broom from storage (2), Broom (7), Return the broom (2)
Others	Sit and work with tablet^b (30), Exercise^b (30), Watch TV^b (15), Iron^b (5), Sleep^b (20)

Figure 5.4 shows the average of the best obtained performance (among the 5 runs) after each black-box function query by GA, greedy, and BO, for different numbers of sensors and ϵ values in T1 and T2. It can be seen that after about 300 queries, BO's performance using any number of sensors is better than GA and greedy even after 1000 queries. The only exceptions are the greedy algorithm with 9, 11, and 15 sensors in T2 with ϵ=1, where the performance is slightly better than BO with 5 and 7 sensors. This finding suggests that BO explores the search space more effectively. To corroborate this finding, we compare the F^1 of each function query (a measure of the query quality) from GA, greedy, and BO, and then examine sensor locations of the best sensor configurations found by these methods. Figure 5.5 shows the best optimization runs (out of the 5 runs) by GA, greedy with 15 sensors for ϵ=1, and BO with 9, 7, and 9 sensors for ϵ=0.25, ϵ=0.5, and ϵ=1 respectively in T1 (Figure 5.5a–5.5c). In case of T2, the figure shows the same result for GA, greedy with 7, and 11 sensors for ϵ=0.5 and ϵ=1 respectively, and BO with 11, 11, and 13 sensors for ϵ=0.25, ϵ=0.5, ϵ=1 respectively (Figure 5.5d–5.5f). Every observation after BO's query to the function is plotted as a dot and the best obtained performance of each of the three methods is plotted using dashed and solid line segments. Observe that for any ϵ value except for

Table 5.2: The performance of GA, Greedy, and BO in terms of the macro-averaged F1-score (Equation 5.2) in T1. A dash is used when a sensor configuration is not found using 1000 queries.

		Avg. F^1 (± one standard deviation)		
Method	**Sensors #**	ϵ=0.25(m)	ϵ=0.5(m)	ϵ=1.0(m)
Baseline 1 GA	N/A	56.7±1.0 average no. sensors:20.4	59.7±0.4 average no. sensors:13.2	54.5±1.3 average no. sensors:11.4
Baseline 2 Greedy	5	—	—	53.3±6.9
	7	—	—	53.9±0.9
	9	—	—	57.3±3.1
	11	—	—	56.5±0.8
	13	—	—	58.6±2.8
	15	—	—	59.7±3.0
BO	5	73.8±0.8	72.4±1.0	68.4±4.4
	7	75.2±0.9	**76.5±1.9**	72.8±1.2
	9	**77.8±1.5**	73.4±1.6	**75.3±1.1**
	11	75.0±1.8	74.1±1.5	71.3±1.3
	13	74.8±1.9	75.2±0.9	73.2±1.9
	15	76.1±0.9	75.3±1.4	70.7±0.9

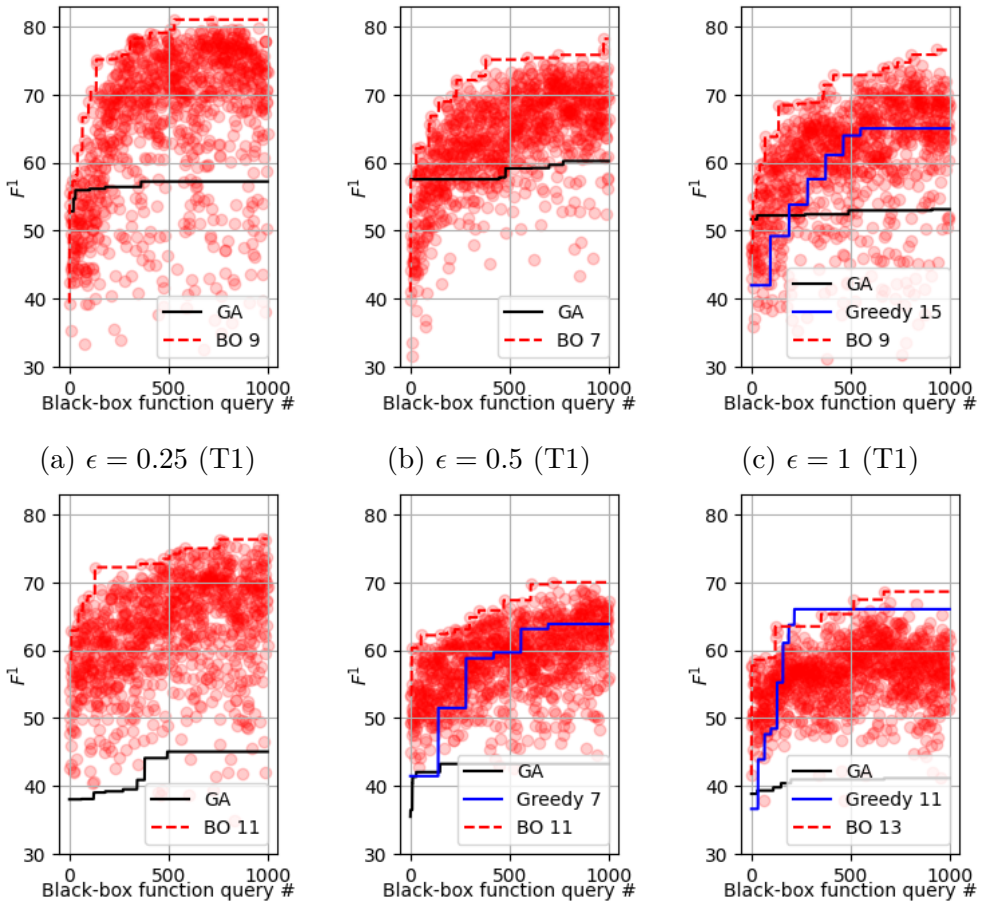
$\epsilon = 1$ in T2 (Figure 5.5f), most of the BO's observations yield better performance than the best observations of GA and greedy. Specifically, in Figures 5.5a to 5.5c, 77.9%, 79.4%, 81.8%, 98.8%, 99.8% and 99.8% of BO's observations are higher than GA's best performance. Similarly, in Figures 5.5c, 5.5e, and 5.5f, 62.5%, 51.8%, and 14.2% of BO's observations are better than greedy's best performance. The performance of greedy is acceptable in Figure 5.5f because the search space is small. Thus, the chance of reaching the global optimum point is higher as there are fewer options to explore.

Figure 5.6 shows sensor locations for the best sensor configuration found for every ϵ value in the two testbeds. The heatmap overlaid on the floor plan shows the occupant

Table 5.3: The performance of GA, Greedy, and BO in terms of the macro-averaged F1-score (Equation 5.2) in T2. There is no significant difference between results that have the same superscript. A dash is used when a sensor configuration is not found using 1000 queries.

		Avg. F^1 (± one standard deviation)		
Method	**Sensors #**	ϵ=0.25(m)	ϵ=0.5(m)	ϵ=1.0(m)
Baseline 1 GA	N/A	43.3±0.9 average no. sensors:17.4	42.1±1.4 average no. sensors:10.6	42.5±0.7 average no. sensors:10.0
Baseline 2 Greedy	5	—	59.0±2.0	56.5±3.8
	7	—	59.8±2.7	59.5±1.7^{a}
	9	—	—	62.3±1.7^{b,d}
	11	—	—	63.7±1.7^{e,f}
	13	—	—	56.5±1.2
	15	—	—	61.9±0.5^{c}
BO	5	65.7±0.9	64.9±0.6	61.0±0.8^{a,b,c}
	7	67.0±0.8	66.6±0.6	63.4±0.4^{d,e}
	9	69.1±0.5	67.8±1.0	64.5±0.7^{f}
	11	**73.7±2.1**	**68.4±0.9**	65.9±1.1
	13	68.4±0.6	67.7±1.0	**67.4±1.1**
	15	68.4±1.4	67.3±0.8	67.4±0.9

locations (yellow indicates higher occupancy than red and white; white indicates little to no occupancy). Notice that all methods place sensors in highly occupied locations, such as the kitchen and dining room. In the other areas, BO tends to put motion sensors in more appropriate locations. In particular, in T1, BO always places one or more sensors above the toilet and/or bathroom sink, and pays more attention to the infrequent activities that happen in the storage room than the two baselines. Interestingly, both GA and greedy place sensors in areas such as the balcony and entryway, where no activities were performed. In T2, BO is the only method that places at least one sensor in the living room for all ϵ values.



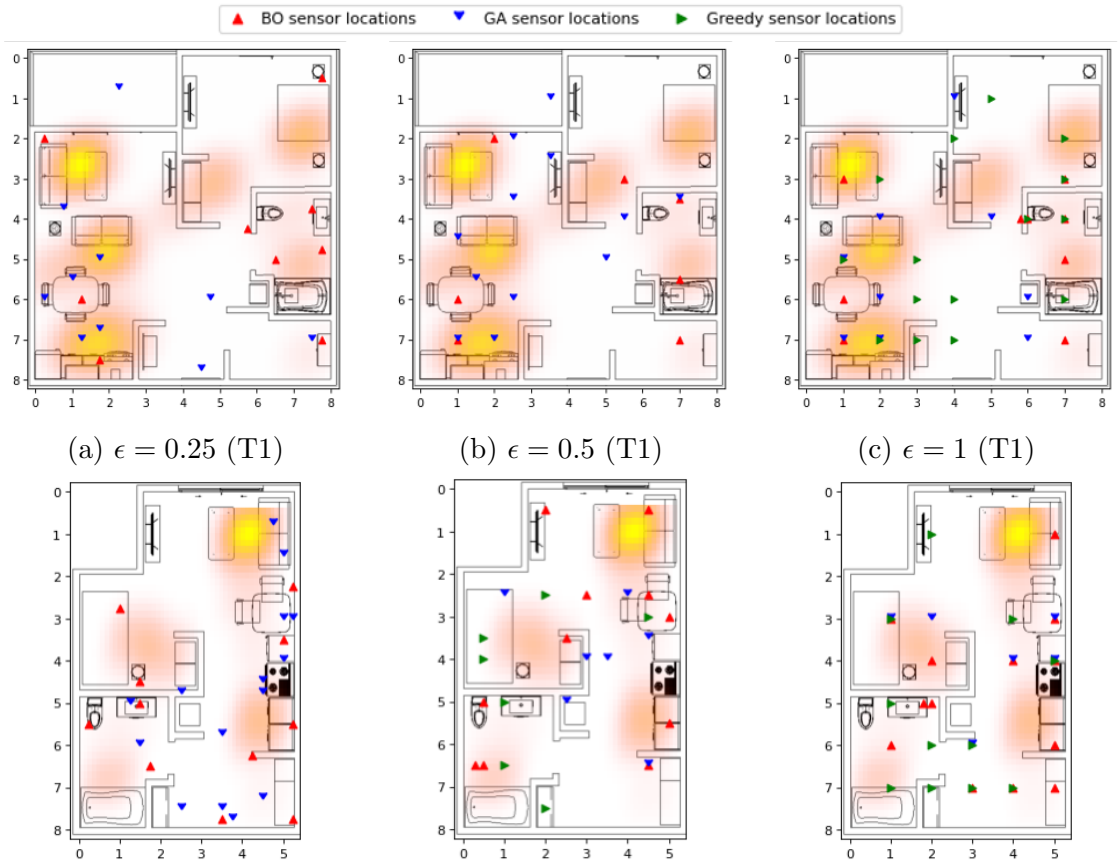
(a) $\epsilon = 0.25$ (T1) (b) $\epsilon = 0.5$ (T1) (c) $\epsilon = 1$ (T1)

(d) $\epsilon = 0.25$ (T2) (e) $\epsilon = 0.5$ (T2) (f) $\epsilon = 1$ (T2)

Figure 5.5: Best performed GA, greedy and BO with different no. sensors on T1 and T2 across black-box function queries.

5.5 Scalability and Modularity Tests

Scalability is a critical factor in the design of any component or system. It refers to the ability of a system to handle an increasing workload without compromising performance, quality, or user experience. For our sensor configuration optimization component, scalability is crucial as it determines the range of applications the component can support. Scalability should be explored in terms of different sensor types, the number of activities, the number of occupants, and the size of the indoor environment. Our component's primary application is for automatic ageing-in-place systems, which involve monitoring the daily living activities of an older individual in



(a) $\epsilon = 0.25$ (T1) (b) $\epsilon = 0.5$ (T1) (c) $\epsilon = 1$ (T1)

(d) $\epsilon = 0.25$ (T2) (e) $\epsilon = 0.5$ (T2) (f) $\epsilon = 1$ (T2)

Figure 5.6: Motion sensor configurations found by GA, greedy, and BO for different ϵ values in T1 and T2. The coincident sensors in the BO sensor configuration are shown with a small offset.

a residential indoor environment. Therefore, the scalability of our component mainly concerns the various sensor types and their quantity.

A component is modular, if it is built with separate, self-contained modules that can be easily combined, modified, or replaced without affecting the performance of other modules and the overall performance of the component.

In this section, we conduct a thorough investigation into the scalability and modularity of our component. Specifically, we evaluate the quality of sensor configurations given different sensor types and an increasing number of sensors. We also demonstrate that our component is modular by interchanging its important modules.

We use the $8{\times}8(m^2)$ fully-furnished one-bedroom suite (T1 in Figure 5.3) as well

Table 5.4: The performance of our component with BO in terms of the macro-averaged F1-Score (Equation 5.2) in T1 using 7 motion sensors and varying LS sensors (type and number).

IS Sensor #	Avg. F^1 (± one standard deviation)		
	IS Sensors Type: Pressure Sensor	**IS Sensors Types:** Pressure Sensor, Electricity Sensor	**IS Sensors Types:** Pressure Sensor, Electricity Sensor, Accelerometer
1	74.9±0.5	73.2±1.2	76.9±1.2
2	74.8±1.4	72.5±2.6	78.5±1.2
3	74.3±1.3	74.0±0.7	79.1±1.9
4	73.5±1.8	73.1±2.6	79.0±2.1
5	74.0±1.2	74.2±1.8	81.7±0.7
6	74.9±1.3	73.2±1.3	80.2±2.3
7	74.6±1.7	76.3±1.6	81.5±1.1
8	73.4±2.0	74.7±0.8	82.8±1.3
9	77.3±1.2	74.0±1.4	82.2±0.9
10	74.3±0.9	75.8±1.1	82.7±1.7

as including five (simulated) occupants performing ADLs (Table 5.1) defined in this chapter.

5.5.1 Scalability Test

We run the optimization for different configurations of IS sensor types and IS sensor numbers combined with 7 motion sensors as LS sensors and set $\epsilon= = 0.5$. Specifically, we evaluate the scalability by incrementally allowing three IS sensor types, *i.e.* pressure sensor, electricity sensor, and accelerometer, to consider by the optimizer in the sensor configuration optimization process. Finally, we repeat the experiment after setting the total number of IS sensors to $1, 2, 3, ..., 10$.

Table 5.4 shows the performance of our component with BO in terms of macro-averaged F1-Score in T1. Regarding the sensor's scalability, it can be seen that our

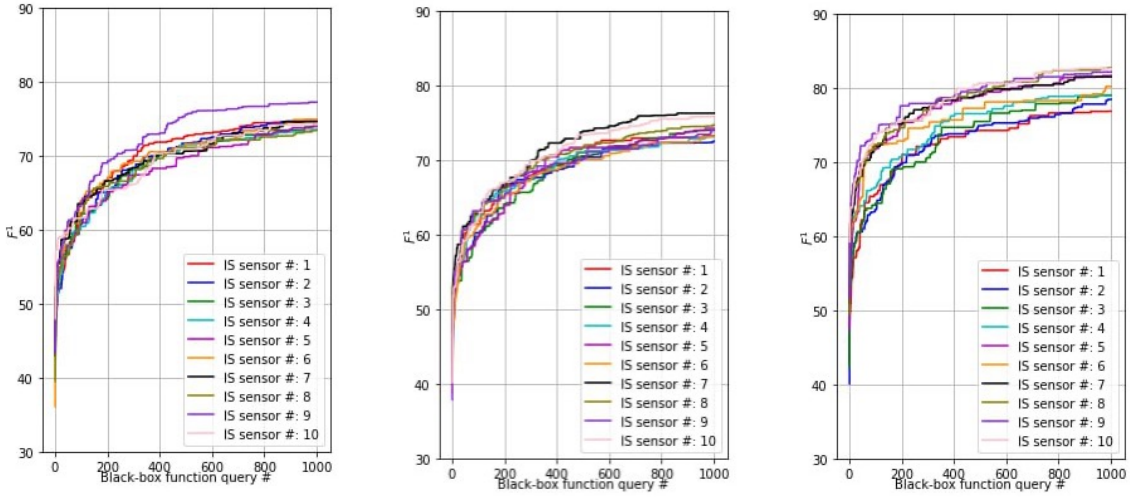
component finds accurate sensor configurations even given high-dimensional sensor configuration search space, *e.g.* 7 motion sensors (LS sensors) and 10 pressure sensors (as IS sensors). Furthermore, for more IS sensor types, our component is not only scalable but also finds more accurate sensor configurations, such as F1-Score of 82.7±1.7 with 7 motion sensors (as LS sensors) and 10 pressure/electricity/accelerometer sensors (as IS sensors).

Figure 5.7 shows the average of the best-obtained performance (among the 5 runs) after each black-box function query by BO when considering pressure sensor only 5.7a, pressure sensor and electricity sensor 5.7b, and pressure sensor, electricity sensor, and accelerometer 5.7c. It can be seen that our component converges with any given IS sensor number/type; hence, scalable. Interestingly, the component's performance does not significantly change when considering pressure and electricity sensors in comparison with pressure sensors only. However, when considering the accelerometer, too, the performance increase substantially for any IS sensor number, implying that accelerometer data is vital for activity recognition.

5.5.2 Modularity Test

We wish to emphasize that our component is not designed for a particular choice of activity classifier. To show this, we have repeated the optimization process in T1 for ϵ=1 using two different activity classifiers: gradient boosting and K-nearest neighbour (KNN) with k=5. The GA, greedy with 15 sensors, and BO with 9 sensors respectively yield the F^1 score of 56.1±2.1, 58.2±4.1 and 73.8±0.9 when using the gradient boosting classifier, and the F^1 score of 49.3±1.6, 51.6±2.2 and 60.8±0.8 when using KNN. In both cases, the superior performance of BO remains statistically significant.

Our component is not limited to a specific simulation methodology. To demonstrate this, we have repeated the optimization process in T1 for ϵ=1 using the simulation methodology from [139]. The GA, greedy with 15 sensors, and BO with 9 sensors



(a) The performance of our component considering pressure sensor.

(b) The performance of our component considering pressure sensor, and electricity sensor.

(c) The performance of our component considering pressure sensor, electricity sensor, and accelerometer.

Figure 5.7: Best performed BO with 7 motion sensors and different no. and types of IS sensors on T1 across black-box function queries with ϵ=0.5.

respectively yield the F^1 score of 53.9±0.5, 60.5±1.1 and 72.9±0.6. This shows that BO remains effective with a different simulation methodology.

5.6 Discussion

Our experiments confirm that the proposed component based on BO outperforms the conventional methods for finding the best motion sensor configuration, resulting in significantly more accurate activity recognition task. It acquires a surrogate model of the given black-box function and updates it with new observations, allowing for more effective sampling as the number of observations increases. Each query takes about 2 minutes, making exhaustive search impractical.

We argue that the surrogate model of the function is a sufficiently accurate representation of the function. Further studies can be conducted using this surrogate model to address other research questions, such as maximizing the robustness of sensor configuration (some attention has been paid recently [57]). The robustness is

defined as the average performance of the configurations that are relatively similar. Hence, it can be interpreted as robustness to minor installation errors. To quantify this, one can analyze the gradient of the surrogate model in the vicinity of a particular configuration. The gradient is also useful for incorporating the sensors' precision and accuracy. Another direction is to apply transfer learning to the surrogate model to use its knowledge in other environments.

To understand how changing the hyperparameters of GA affects the exploration/-exploitation trade-off, we considered different choices in one case, *i.e.* ϵ=1 in T1. We found that GA's performance shows little sensitivity to the choice of hyperparameters, and even in the best case it improves negligibly compared to the result presented in Table 5.2.

5.7 Conclusion

Our experiments confirm that the proposed component based on BO outperforms the conventional methods for finding the best motion sensor configuration, resulting in significantly more accurate activity recognition tasks. It acquires a surrogate model of the given black-box function. Then, it updates it with new observations, allowing for more effective sampling as the number of observations increases. Each query takes about 2 minutes, making exhaustive search impractical.

We argue that the surrogate model of the function is a sufficiently accurate representation. Further studies can be conducted using this surrogate model to address other research questions, such as maximizing the robustness of sensor configuration (some attention has been paid recently [57]). The robustness is defined as the average performance of relatively similar configurations. Hence, it can be interpreted as robustness to minor installation errors. To quantify this, one can analyze the gradient of the surrogate model in the vicinity of a particular configuration. The gradient is also useful for incorporating the sensors' precision and accuracy. Another direction is to apply transfer learning to the surrogate model to use its knowledge in other

environments. To understand how changing the hyperparameters of GA affects the exploration/exploitation tradeoff, we considered different choices in one case, *i.e.* ϵ=1 in T1. We found that GA's performance shows little sensitivity to the choice of hyper-parameters, and even in the best case, it improves negligibly compared to the result presented in Table 5.2.

The main limitation of BO is that it might place coincident sensors, which can be prevented by incorporating some constraints, which is a future work direction. The results show that the optimal configuration of one type of sensor is not deemed intrusive and is commonly used for activity detection in ageing-in-place settings. We plan to extend our method to configure other sensor types *e.g.*pressure sensors.

Chapter 6

Conclusion and Future Work

This thesis culminates with drawing the central conclusions from the previous chapters. To this end, we first recap the key contributions made towards *building a simulation-driven sensor configuration evaluation framework*, our primary goal in this thesis. Then, we take stock of the findings and contributions to discuss potential future research directions.

6.1 Summary of Contributions

The present thesis set out to alleviate the problem of optimizing sensor configuration for Smart Indoor Spaces (SIS) applications, capable of detecting contextual information needed for the applications. We argued that achieving this goal demands the following four key requirements, stated in Section 1.2:

- (R1) the ability to derive *contextual information* from raw sensor readings in SIS applications,
- (R2) evaluating different sensor configurations using synthetic datasets,
- (R3) a SIS simulation methodology capable of modelling various scenarios, and
- (R4) a simulation-driven optimization framework for finding optimal sensor configurations.

This section discusses the contributions of my research and their connections to the stated requirements. To begin with, we prototyped two proof of concepts in Chapter 2. The first prototype featured two data-driven models, namely particle filters and dynamic neural networks, employed for predicting the number of occupants as contextual information. The methods utilized raw occupancy-indicative sensor readings as input and outputted the predictive number of occupants in separate rooms of two buildings, serving as our testbeds. The underlying assumption was that the target contextual information always presents distinct enough patterns in the space of sensor readings so that a prediction model can easily recognize it. The first prototype demonstrated that high-quality sensor readings contain specific contextual information and that data-driven models can recognize them accurately, fulfilling the first requirement (R1).

In the second prototype, we proposed utilizing the predicted contextual information to assist SIS designers effortlessly evaluating different sensor configuration deployments. The prototype leveraged an off-the-shelf SIS simulation methodology instead of expensive-to-obtain real-world datasets. The simulator allows obtaining datasets needed by data-driven models effortlessly and quickly, which are reproducible and readily available for any floor plan. This prototype showed that the accuracy of a localization algorithm could be employed to analyze the performance of an underlying sensor configuration in different areas of the indoor space, fulfilling the second requirement (R2).

In Chapter 3, we proceeded to work on a crucial component of our framework, namely, a simulation methodology. First, we proposed a unified taxonomy of simulation methodologies for SIS applications and conducted a systematic review of the available methodologies. Then, we applied a hierarchical clustering method based on our proposed taxonomy to divide the methodologies into meaningful groups. We demonstrated that the available methodologies lack the generalization feature and high-fidelity in modeling occupants and sensors required for effective simulation of

SIS applications.

In Chapter 4, we proposed a high-fidelity simulation methodology capable of modelling any indoor space, thus generalizable. Our methodology, *i.e.* SIM_{sis} toolkit, takes in as input any intended indoor space representation in Industry Foundation Classes (IFC) for Building Information Modeling (BIM) format [1]. It models occupants' behaviour using a hierarchy-based model, *i.e.* a hierarchy of activities and a sequence of actions for each activity, capable of modelling a wide range of indoor activities in residential and commercial applications. Furthermore, the SIM_{sis} can simulate several off-the-shelf sensors widely used in SIS applications, *i.e.*motion sensors, pressure sensors, electricity sensors, and accelerometers. Finally, considering the dataset of a real-world SIS application, the chapter showed that the replication of the application in SIM_{sis} (its digital twin) generates high-fidelity synthetic datasets in terms of both occupants and sensor behaviours. Therefore, Chapter 3 and Chapter 4 fulfill the third requirement (R3).

In Chapter 5, a simulation-driven framework was proposed for the evaluation of different sensor configurations using the proposed simulation methodology. We argued that most of the SIS applications require knowledge about occupants' activities as contextual information. Therefore, the sensor configuration deployment should generate distinct sensors' reading patterns for each indoor activity to obtain an accurate activity recognition algorithm. This structure can be viewed as an optimization problem, wherein an optimizer sequentially proposes candidate sensor configurations to maximize the accuracy of an activity classifier as its objective function. The Bayesian Optimization (BO) approach was proposed as the optimizer of this framework. We demonstrated that BO is able to find sensor configurations that result, on average, 16% more accurate activity classifier in comparison to state-of-the-art methods, such as Genetic Algorithm (GA) and greedy. We experimentally showed that BO outper-

[1]The IFC data model is an ISO International Standard, and an open architectural and building data description specification.

forms the previous methods because, in each iteration, it trains a surrogate model of the objective function using its available local information of previous observations, as opposed to the related work methods, which solely rely on the local information. The surrogate model serves as a global information which effectively guides the optimizer towards better sensor configurations. This framework satisfies the forth requirement (R4). Furthermore, we showed that the proposed framework is scalable in terms of the number and type of sensors. Finally, our experiments have demonstrated that our framework's performance remains consistent when its components are altered, highlighting its modularity.

6.2 Future Work

We strongly believe there is still a long way to go to accomplish a genuine sensor configuration evaluation framework, which leaves ample avenues for future work. In this section, we discuss the open problems based on the insights we gained doing this thesis.

6.2.1 Simulation Methodology

Sensor Model

Although high-fidelity, the SIM_{sis} methodology is only evaluated using a limited real-world dataset, where only LS sensor types were deployed. Much sophisticated sensor modelling and a broader study are required to enhance the ability of the simulation methodology to model both LS and IS sensor types realistically.

Space Model

The SIM_{sis} uses IFC representation of indoor spaces for quickly building a digital twin of an intended indoor space. Unfortunately, the IFC models of currently built buildings are not often available. Thus, designing such models might be time-consuming and require specific skills in professional software tools like Revit. Currently, deep

learning approaches are used to construct an IFC model of an indoor space using digital images [157]. Automatically generating IFC models is a promising future direction as it promises more generalization of SIS simulation methodologies.

Agent Model

The agent model in SIM_{sis} simulates a single occupant living in a residential building. The agent model can be improved to support multi-agent scenarios where agents can collaborate to accomplish activities. This will enhance the range of supported applications to commercial buildings such as offices. However, it requires a sophisticated evaluation experiment.

6.2.2 Sensor Configuration Evaluation

The proposed Bayesian optimization can be studied in more detail to incorporate evaluating sensor configurations from different perspectives. This thesis considered the SIS simulation and its contextual information representation a stochastic system and modelled it as a black-box function $f(x)$. Consequently, black-box optimization disregards available knowledge about $f(x)$ that might be useful for sensor configuration evaluation. The knowledge can be used to evaluate sensor configurations from different perspectives, such as robustness, *i.e.*analyzing the performance of the real-world sensor configuration deployment, considering minor installation errors. In contrast to black-box optimization, grey-box optimization, such as [9], models $f(x)$ as a combination of data and some theoretical structures. It has been shown in [9] that the knowledge from the theoretical structure of $f(x)$ supports the Bayesian optimization method to explore/exploit the search space more effectively.

Moreover, applying transfer learning techniques is also promising. As [63] and [148] show, datasets of similar tasks can train an acquisition function first and transfer its knowledge to a similar optimization task can significantly increase the Bayesian optimization performance. In sensor configuration evaluation, instead of learning

the model from scratch, the idea is to use the model's knowledge for specific areas, *e.g.*kitchen and living room, of an indoor environment from previous tasks. In our application, the model obtains the knowledge of the occupant's trace in each room. Therefore, it replicates highly experienced designers that learn several basic "rules" (also supported by [139]) through experience that certain areas require a specific sensor configuration; for instance, placing pressure sensors on dining room chairs ensures capturing useful data for dining activities.

The sensor configuration deployment requires maintenance from different aspects, such as resource management based on occupants' behaviour. Resource management becomes important because it can significantly save energy consumption and has been studied rigorously [71]. In the context of a sensor configuration for SIS applications, a sequential decision-making strategy, such as a Reinforcement Learning (RL) agent, can be employed to decide, based on the state of the occupants, which sensor to hibernate/turn on.

Bibliography

[1] T. Abade, T. Gomes, J. L. Silva, and J. C. Campos, "Design and evaluation of a smart library using the apex framework," in *International Conference on Distributed, Ambient, and Pervasive Interactions*, Springer, 2014, pp. 307–318.

[2] N. Ahmed, S. S. Kanhere, and S. Jha, "Probabilistic coverage in wireless sensor networks," in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l*, IEEE, 2005, 8–pp.

[3] V. Akbarzadeh, C. Gagne, M. Parizeau, M. Argany, and M. A. Mostafavi, "Probabilistic sensing model for sensor placement optimization based on line-of-sight coverage," *IEEE transactions on instrumentation and measurement*, vol. 62, no. 2, pp. 293–303, 2012.

[4] N. Alshammari, T. Alshammari, M. Sedky, J. Champion, and C. Bauer, "Openshs: Open smart home simulator," *Sensors*, vol. 17, no. 5, p. 1003, 2017.

[5] O. Ardakanian, A. Bhattacharya, and D. Culler, "Non-intrusive techniques for establishing occupancy related energy savings in commercial buildings," in *Proc. 3rd International Conference on Systems for Energy-Efficient Built Environments (BuildSys)*, ACM, 2016, pp. 21–30.

[6] O. Ardakanian, A. Bhattacharya, and D. Culler, "Non-intrusive techniques for establishing occupancy related energy savings in commercial buildings," in *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, 2016, pp. 21–30.

[7] I. B. Arief-Ang, F. D. Salim, and M. Hamilton, "Da-hoc: Semi-supervised domain adaptation for room occupancy prediction using co2 sensor data," in *Proc. 4th International Conference on Systems for Energy-Efficient Built Environments (BuildSys)*, ACM, 2017, In Press.

[8] I. Armac and D. Retkowitz, "Simulation of smart environments," in *IEEE International Conference on Pervasive Services*, IEEE, 2007, pp. 257–266.

[9] R. Astudillo and P. I. Frazier, "Thinking inside the box: A tutorial on grey-box bayesian optimization," in *2021 Winter Simulation Conference (WSC)*, IEEE, 2021, pp. 1–15.

[10] L. Atzori, A. Iera, and G. Morabito, "Understanding the internet of things: Definition, potentials, and societal role of a fast evolving paradigm," *Ad Hoc Networks*, vol. 56, pp. 122–140, 2017.

[11] J. C. Augusto and M. J. Hornos, "Software simulation and verification to increase the reliability of intelligent environments," *Advances in Engineering Software*, vol. 58, pp. 18–34, 2013.

[12] *Autodesk revit®*, https://www.autodesk.com/products/revit/, Accessed: 2022-12-20.

[13] S. Azhar, "Building information modeling (bim): Trends, benefits, risks, and challenges for the aec industry," *Leadership and management in engineering*, vol. 11, no. 3, pp. 241–252, 2011.

[14] G. Bang and I. Ko, "A user empirical context model for a smart home simulator," in *Advances in Computer Science and Ubiquitous Computing*, Springer, 2016, pp. 555–560.

[15] R. Barr and et al., "JiST: Embedding simulation time into a virtual machine," in *IN EUROSIM CONGRESS ON MODELLING AND SIMULATION*, 2004.

[16] J. Barry and C. Thron, "A computational physics-based algorithm for target coverage problems," in *Advances in nature-inspired computing and applications*, Springer, 2019, pp. 269–290.

[17] B. Bertran, J. Bruneau, D. Cassou, N. Loriant, E. Balland, and C. Consel, "Diasuite: A tool suite to develop sense/compute/control applications," *Science of Computer Programming*, vol. 79, pp. 39–51, 2014.

[18] S. Bicakci and H. Gunes, "Hybrid simulation system for testing artificial intelligence algorithms used in smart homes," *Simulation Modelling Practice and Theory*, vol. 102, p. 101 993, 2020.

[19] *Bimobject*, https://www.bimobject.com/en, Accessed: 2022-12-20.

[20] *BLE RSSI dataset for indoor localization and navigation data set*, https://archive.ics.uci.edu/ml/datasets/BLE+RSSI+Dataset+for+Indoor+\localization+and+Navigation, Accessed: 2018-12-09.

[21] K. Bouchard, A. Ajroud, B. Bouchard, and A. Bouzouane, "Simact: A 3d open source smart home simulator for activity recognition," in *Advances in Computer Science and Information Technology*, Springer, 2010, pp. 524–533.

[22] J. Bruneau and C. Consel, "Diasim: A simulator for pervasive computing applications," *Software: Practice and Experience*, vol. 43, no. 8, pp. 885–909, 2013.

[23] J. Bruneau, C. Consel, M. OMalley, W. Taha, and W. M. Hannourah, "Virtual testing for smart buildings," in *2012 Eighth International Conference on Intelligent Environments*, IEEE, 2012, pp. 282–289.

[24] T. Buchholz and M. Schiffers, "Quality of context: What it is and why we need it," in *In Proceedings of the 10th Workshop of the OpenView University Association: OVUA '03*, 2003.

[25] M. Buchmayr, W. Kurschl, and J. Küng, "A simulator for generating and visualizing sensor data for ambient intelligence environments," *Procedia Computer Science*, vol. 5, pp. 90–97, 2011, The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information, Systems (MobiWIS 2011), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2011.07.014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050911003395.

[26] M. Buchmayr, W. Kurschl, and J. Küng, "A simulator for generating and visualizing sensor data for ambient intelligence environments," *Procedia Computer Science*, vol. 5, pp. 90–97, 2011.

[27] C. Burghardt, S. Propp, T. Kirste, and P. Forbrig, "Rapid prototyping and evaluation of intention analysis for smart environments," in *International Conference on Intelligent Interactive Assistance and Mobile Multimedia Computing*, Springer, 2009, pp. 239–250.

[28] L. M. Candanedo, V. Feldheim, and D. Deramaix, "A methodology based on hidden markov models for occupancy detection and a case study in a low energy residential building," *Energy and Buildings*, vol. 148, no. Supplement C, pp. 327–341, 2017.

[29] F. Cardinaux, S. Brownsell, D. Bradley, and M. S. Hawley, "A home daily activity simulation model for the evaluation of lifestyle monitoring systems," *Computers in biology and medicine*, vol. 43, no. 10, pp. 1428–1436, 2013.

[30] M. Caruso, Ç. Ilban, F. Leotta, M. Mecella, and S. Vassos, "Synthesizing daily life logs through gaming and simulation," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, 2013, pp. 451–460.

[31] S. Casaccia, R. Rosati, L. Scalise, and G. M. Revel, "Measurement of activities of daily living: A simulation tool for the optimisation of a passive infrared sensor network in a smart home environment," in *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, IEEE, 2020, pp. 1–6.

[32] CASAS, *Center of advanced studies in adaptive systems (casas) at washington state university*, http://casas.wsu.edu/, Accessed: 2022-07-30.

[33] J. Cassens, F. Schmitt, T. Mende, and M. Herczeg, "Casi–a generic context awareness simulator for ambient systems," in *International Joint Conference on Ambient Intelligence*, Springer, 2012, pp. 421–426.

[34] *Center of advanced studies in adaptive systems (casas)*, http://casas.wsu.edu/, Accessed: 2021-08-30.

[35] W.-K. Chang and T. Hong, "Statistical analysis and modeling of occupancy patterns in open-plan offices using measured lighting-switch data," *Building Simulation*, vol. 6, no. 1, pp. 23–32, 2013.

[36] *Cpn tools*, https://cpntools.org/, Accessed: 2021-08-17.

[37] J. Dahmen and D. Cook, "Synsys: A synthetic data generation system for healthcare applications," *Sensors*, vol. 19, no. 5, p. 1181, 2019.

[38] P. De, A. Raniwala, S. Sharma, and T. Chiueh, "MiNT: A miniaturized network testbed for mobile wireless research," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 4, 2005, 2731–2742 vol. 4.

[39] G. Dorffner, "Neural networks for time series processing," vol. 6, pp. 447–468, 1996.

[40] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proceedings of the 21st international conference on Software engineering*, 1999, pp. 411–420.

[41] A. Elbayoudi, A. Lotfi, C. Langensiepen, and K. Appiah, "Modelling and simulation of activities of daily living representing an older adult's behaviour," in *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2015, pp. 1–8.

[42] V. L. Erickson, S. Achleitner, and A. E. Cerpa, "Poem: Power-efficient occupancy-based energy management system," in *Proc. 12th International Conference on Information Processing in Sensor Networks (IPSN)*, ACM/IEEE, 2013, pp. 203–216.

[43] M. P. Fanti, G. Faraut, J.-J. Lesage, and M. Roccotelli, "An integrated framework for binary sensor placement and inhabitants location tracking," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 1, pp. 154–160, 2016.

[44] M. P. Fanti, M. Roccotelli, G. Faraut, and J.-J. Lesage, "Smart placement of motion sensors in a home environment," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2017, pp. 894–899.

[45] C. Fernández-Llatas *et al.*, "Ambient assisted living spaces validation by services and devices simulation," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2011, pp. 1785–1788.

[46] F. Fiebig, S. Kochanneck, I. Mauser, and H. Schmeck, "Detecting occupancy in smart buildings by data fusion from low-cost sensors: Poster description," in *Proc. 8th International Conference on Future Energy Systems (e-Energy)*, ACM, 2017, pp. 259–261.

[47] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[48] Y. Francillette *et al.*, "Modeling the behavior of persons with mild cognitive impairment or alzheimer's for intelligent environment simulation," *User Modeling and User-Adapted Interaction*, vol. 30, no. 5, pp. 895–947, 2020.

[49] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[50] Q. Fu, P. Li, C. Chen, L. Qi, Y. Lu, and C. Yu, "A configurable context-aware simulator for smart home systems," in *2011 6th International Conference on Pervasive Computing and Applications*, IEEE, 2011, pp. 39–44.

[51] C. Garcıa-Rodrıguez, R. Martınez-Tomás, and J. M. Cuadra-Troncoso, "Smart spaces and monitoring simulation," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, Springer, 2013, pp. 190–199.

[52] S. K. Ghai, L. V. Thanayankizil, D. P. Seetharam, and D. Chakraborty, "Occupancy detection in commercial buildings using opportunistic context sources," in *Pervasive Computing and Communications Workshops*, IEEE, 2012, pp. 463–466.

[53] C. Godsey and M. Skubic, "Using elements of game engine architecture to simulate sensor networks for eldercare," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2009, pp. 6143–6146.

[54] S. Golestan, I. Nikolaidis, and E. Stroulia, "Towards a simulation framework for smart indoor spaces," *Sensors*, vol. 20, no. 24, p. 7137, 2020.

[55] S. Golestan, A. Petcovici, I. Nikolaidis, and E. Stroulia, "Simulation-based deployment configuration of smart indoor spaces," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, IEEE, 2019, pp. 358–363.

[56] B. d. T. P. Gomes *et al.*, "A middleware with comprehensive quality of context support for the internet of things applications," *Sensors*, vol. 17, no. 12, 2017, ISSN: 1424-8220. DOI: 10.3390/s17122853. [Online]. Available: http://www.mdpi.com/1424-8220/17/12/2853.

[57] O. Gungor, T. Rosing, and B. Aksanli, "Caheros: Constraint-aware heuristic approach for robust sensor placement," in *2021 IEEE Sensors*, IEEE, 2021, pp. 1–4.

[58] O. Gungor, T. S. Rosing, and B. Aksanli, "Respire: Robust sensor placement optimization in probabilistic environments," in *2020 IEEE Sensors*, IEEE, 2020, pp. 1–4.

[59] M. Gupta and R. Sandhu, "Authorization framework for secure cloud assisted connected cars and vehicular internet of things," in *Proceedings of the 23nd ACM on symposium on access control models and technologies*, 2018, pp. 193–204.

[60] A. Helal, A. Mendez-Vazquez, and S. Hossain, "Specification and synthesis of sensory datasets in pervasive spaces," in *2009 IEEE Symposium on Computers and Communications*, IEEE, 2009, pp. 920–925.

[61] B. Ho, D. Vogts, and J. Wesson, "A smart home simulation tool to support the recognition of activities of daily living," in *Proceedings of the South African Institute of Computer Scientists and Information Technologists 2019*, 2019, pp. 1–10.

[62] R. A. Horn, "The hadamard product," in *Proc. Symp. Appl. Math*, vol. 40, 1990, pp. 87–169.

[63] B.-J. Hsieh, P.-C. Hsieh, and X. Liu, "Reinforced few-shot acquisition function learning for bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7718–7731, 2021.

[64] C. Hu, L. Dai, X. Yan, W. Gong, X. Liu, and L. Wang, "Modified nsga-iii for sensor placement in water distribution system," *Information Sciences*, vol. 509, pp. 488–500, 2020.

[65] Q. Hu, F. Oldewurtel, M. Balandat, E. Vrettos, D. Zhou, and C. J. Tomlin, "Building model identification during regular operation - empirical results and challenges," in *Proc. American Control Conference*, 2016, pp. 605–610.

[66] S. Jang and W. Woo, "Ubi-ucam: A unified context-aware application model," in *International and Interdisciplinary Conference on Modeling and Using Context*, Springer, 2003, pp. 178–189.

[67] C. Jiang and A. Mita, "Sisg4hei_alpha: Alpha version of simulated indoor scenario generator for houses with elderly individuals," *Journal of Building Engineering*, vol. 35, p. 101 963, 2021.

[68] M. Jin, N. Bekiaris-Liberis, K. Weekly, C. Spanos, and A. Bayen, "Sensing by proxy: Occupancy detection based on indoor co2 concentration," in *Proc. 9th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2015, pp. 1–10.

[69] O. Kamara-Esteban, G. Azkune, A. Pijoan, C. E. Borges, A. Alonso-Vicario, and D. López-de-Ipiña, "Massha: An agent-based approach for human activity simulation in intelligent environments," *Pervasive And Mobile Computing*, vol. 40, pp. 279–300, 2017.

[70] C. Kang, Y. Oh, and W. Woo, "Widget-based simulator for testing smart space," in *Transactions on Edutainment III*, Springer, 2009, pp. 59–69.

[71] J. Kang *et al.*, "Hierarchical policy learning for hybrid communication load balancing," in *ICC 2021-IEEE International Conference on Communications*, IEEE, 2021, pp. 1–6.

[72] KIT, *Kit energy smart home lab*, Online https://github.com/aifb/eshl-occupancy/, Accessed Jan. 2018.

[73] B. Kormányos and B. Pataki, "Multilevel simulation of daily activities: Why and how?" In *2013 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA)*, IEEE, 2013, pp. 1–6.

[74] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos, "Efficient sensor placement optimization for securing large water distribution networks," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 516–526, 2008.

[75] A. Kushwah, S. Kumar, and R. M. Hegde, "Multi-sensor data fusion methods for indoor activity recognition using temporal evidence theory," *Pervasive and Mobile Computing*, vol. 21, pp. 19–29, 2015.

[76] R. Laidi and D. Djenouri, "Udeploy: User-driven learning for occupancy sensors deployment in smart buildings," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, IEEE, 2018, pp. 209–214.

[77] C. Lee, L. Zappaterra, K. Choi, and H.-A. Choi, "Securing smart home: Technologies, security challenges, and security requirements," in *2014 IEEE Conference on Communications and Network Security*, IEEE, 2014, pp. 67–72.

[78] J. W. Lee, S. Cho, S. Liu, K. Cho, and S. Helal, "Persim 3d: Context-driven simulation and modeling of human activities in smart spaces," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1243–1256, 2015.

[79] W. Lee, S. Cho, W. Song, K. Um, and K. Cho, "Ubisim: Multiple sensors mounted smart house simulator development," in *2013 IEEE 11th International Conference on Dependable, Autonomic and Secure Computing*, IEEE, 2013, pp. 450–453.

[80] Z. Lei, S. Yue, C. Yu, and S. Yuanchun, "Shsim: An osgi-based smart home simulator," in *2010 3rd IEEE International Conference on Ubi-Media Computing*, IEEE, 2010, pp. 87–90.

[81] J. Lertlakkhanakul, J. W. Choi, and M. Y. Kim, "Building data model and simulation platform for spatial interaction management in smart home," *Automation in Construction*, vol. 17, no. 8, pp. 948–957, 2008.

[82] Y. Li *et al.*, "Openbox: A generalized black-box optimization service," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3209–3219.

[83] *Lifestyle options retirement communities*, https://lifestyleoptions.ca/terralosa/, Accessed: 2022-12-20.

[84] P. Liu, J. Fang, and H. Huang, "A multi-objective optimized node deployment algorithm for wireless sensor networks based on the improved abc," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1848, 2021, p. 012 039.

[85] J. Lundström, J. Synnott, E. Järpe, and C. D. Nugent, "Smart home simulation using avatar control and probabilistic sampling," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, IEEE, 2015, pp. 336–341.

[86] S. Majumder *et al.*, "Smart homes for elderly healthcare—recent advances and research challenges," *Sensors*, vol. 17, no. 11, p. 2496, 2017.

[87] D. Martino-Saltzman, B. B. Blasch, R. D. Morris, and L. W. McNeal, "Travel behavior of nursing home residents perceived as wanderers and nonwanderers," *The Gerontologist*, vol. 31, no. 5, pp. 666–672, 1991.

[88] A. Masciadri, F. Veronese, S. Comai, I. Carlini, and F. Salice, "Disseminating synthetic smart home data for advanced applications," in *CIKM 2018 Workshops*, CEUR-WS, vol. 2482, 2018, pp. 1–7.

[89] MATLAB, *Toolbox for tsnn*, Online https://www.mathworks.com/help/nnet/modeling-and-prediction-with-narx-and-time-delay-networks.html, 2018.

[90] K. McGlinn, L. Hederman, and D. Lewis, "Simcon: A context simulator for supporting evaluation of smart building applications when faced with uncertainty," *Pervasive and Mobile Computing*, vol. 12, pp. 139–159, 2014.

[91] A. Mendez-Vazquez, A. Helal, and D. Cook, "Simulating events to generate synthetic data for pervasive spaces," in *Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*, Citeseer, 2009.

[92] D. Merico and R. Bisiani, "An agent-based data-generation tool for situation-aware systems," in *2011 Seventh International Conference on Intelligent Environments*, IEEE, 2011, pp. 129–134.

[93] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J. Oh, "Semisupervised deep reinforcement learning in support of iot and smart city services," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 624–635, 2018, ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2712560.

[94] P. Mohebbi, E. Stroulia, and I. Nikolaidis, "Sensor-data fusion for multi-person indoor location estimation," *Sensors*, vol. 17, no. 10, 2017, ISSN: 1424-8220. DOI: 10.3390/s17102377. [Online]. Available: http://www.mdpi.com/1424-8220/17/10/2377.

[95] J. J. Moré, "The levenberg-marquardt algorithm: Implementation and theory," in *Numerical analysis*, Springer, 1978, pp. 105–116.

[96] N. Mori, M. Takeda, and K. Matsumoto, "A comparison study between genetic algorithms and bayesian optimize algorithms by novel indices," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 2005, pp. 1485–1492.

[97] *Mqtt*, http://mqtt.org/, Accessed: 2018-12-07.

[98] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[99] B. Mustafa, "Simulation support for monitored assisted living system development: A vision for the future," in *2013 Seventh International Conference on Next Generation Mobile Apps, Services and Technologies*, IEEE, 2013, pp. 244–249.

[100] D. C. Nazário, A. de Andrade, L. Borges, W. R. Ramos, J. L. Todesco, and M. A. R. Dantas, "An enhanced quality of context evaluating approach in the e-health sensor platform," in *Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, ser. Q2SWinet '15, Cancun, Mexico: ACM, 2015, pp. 1–7, ISBN: 978-1-4503-3757-1. DOI: 10.1145/2815317.2815320. [Online]. Available: http://doi.acm.org/10.1145/2815317.2815320.

[101] H. Nishikawa *et al.*, "Ubireal: Realistic smartspace simulator for systematic testing," in *International conference on ubiquitous computing*, Springer, 2006, pp. 459–476.

[102] N. Noury and T. Hadidi, "Computer simulation of the activity of the elderly person living independently in a health smart home," *Computer methods and programs in biomedicine*, vol. 108, no. 3, pp. 1216–1228, 2012.

[103] E. O'Neill, O. Conlan, and D. Lewis, "Modeling and simulation to assist context aware system design," *Simulation*, vol. 87, no. 1-2, pp. 149–170, 2011.

[104] E. O'Neill, D. Lewis, and O. Conlan, "A simulation-based approach to highly iterative prototyping of ubiquitous computing systems," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 2009, pp. 1–10.

[105] E. O'Neill, O. Conlan, and D. Lewis, "Onments," *Pervasive and Mobile Computing*, vol. 9, no. 1, pp. 76–97, 2013.

[106] *Opensimulator*, http://opensimulator.org/wiki/Main_Page, Accessed: 2021-08-17.

[107] J. L. G. Ortega, L. Han, and N. Bowring, "A novel dynamic hidden semi-markov model (d-hsmm) for occupancy pattern detection from sensor data stream," in *Proc. 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2016, pp. 1–5.

[108] M. Ortiz-Barrios, J. Lundström, J. Synnott, E. Järpe, and A. Sant'Anna, "Complementing real datasets with simulated data: A regression-based approach," *Multimedia Tools and Applications*, pp. 1–24, 2019.

[109] *Osgi*, https://www.osgi.org/, Accessed: 2021-08-17.

[110] G. Papamakarios, D. Giakoumis, K. Votis, S. Segkouli, D. Tzovaras, and C. Karagiannidis, "Synthetic ground truth data generation for automatic trajectory-based adl detection," in *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, IEEE, 2014, pp. 33–36.

[111] J. Park, M. Moon, S. Hwang, and K. Yeom, "Cass: A context-aware simulation system for smart home," in *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007)*, IEEE, 2007, pp. 461–467.

[112] S. Park, A. Savvides, and M. B. Srivastava, "SensorSim: A simulation framework for sensor networks," in *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '00, Boston, Massachusetts, USA: ACM, 2000, pp. 104–111, ISBN: 1-58113-304-9. DOI: 10.1145/346855.346870. [Online]. Available: http://doi.acm.org/10.1145/346855.346870.

[113] Y. Peng, A. Rysanek, Z. Nagy, and A. Schlüter, "Using machine learning techniques for occupancy-prediction-based cooling control in office buildings," *Applied Energy*, vol. 211, pp. 1343–1358, 2018.

[114] P. F. Pereira and N. M. Ramos, "The influence of sensor placement in the study of occupant behavior in a residential building," in *2018 International Conference on Smart Energy Systems and Technologies (SEST)*, IEEE, 2018, pp. 1–6.

[115] M. Pesic, H. Schonenberg, and W. M. Van der Aalst, "Declare: Full support for loosely-structured processes," in *11th IEEE international enterprise distributed object computing conference (EDOC 2007)*, IEEE, 2007, pp. 287–287.

[116] H. Prendinger, B. Brandherm, and S. Ullrich, "A simulation framework for sensor-based systems in second life," *PRESENCE: Teleoperators and Virtual Environments*, vol. 18, no. 6, pp. 468–477, 2009.

[117] J. Renoux and F. Klugl, "Simulating daily activities in a smart home for data generation," in *2018 Winter Simulation Conference (WSC)*, IEEE, 2018, pp. 798–809.

[118] P. Rocha, A. Siddiqui, and M. Stadler, "Improving energy efficiency via smart building energy management systems: A comparison with policy measures," *Energy and Buildings*, vol. 88, pp. 203–213, 2015.

[119] T. Rodner and L. Litz, "Data-driven generation of rule-based behavior models for an ambient assisted living system," in *2013 IEEE Third International Conference on Consumer Electronics¿ Berlin (ICCE-Berlin)*, IEEE, 2013, pp. 35–38.

[120] B. B. Sánchez, R. Alcarria, Á. Sánchez-Picot, and D. Sánchez-de-Rivera, "A methodology for the design of application-specific cyber-physical social sensing co-simulators," *Sensors*, vol. 17, no. 10, p. 2177, 2017.

[121] F. C. Sangogboye, K. Arendt, A. Singh, C. T. Veje, M. B. Kjaergaard, and B. N. Jorgensen, "Performance comparison of occupancy count estimation and prediction with common versus dedicated sensors for building model predictive control," *Building Simulation*, vol. 10, no. 6, pp. 829–843, 2017.

[122] K. Sanmugalingam and G. Coulouris, "A generic location event simulator," in *International Conference on Ubiquitous Computing*, Springer, 2002, pp. 308–315.

[123] D. Schaumann, Y. E. Kalay, S. W. Hong, and D. Simeone, "Simulating human behavior in not-yet built environments by means of event-based narratives," in *Proceedings of the Symposium on Simulation for Architecture & Urban Design*, 2015, pp. 5–12.

[124] *Second life*, https://secondlife.com/, Accessed: 2021-08-17.

[125] D. W. Seo, H. Kim, J. S. Kim, and J. Y. Lee, "Hybrid reality-based user experience and evaluation of a context-aware smart home," *Computers in Industry*, vol. 76, pp. 11–23, 2016.

[126] P. Sernani, D. Calvaresi, P. Calvaresi, M. Pierdicca, E. Morbidelli, and A. F. Dragoni, "Testing intelligent solutions for the ambient assisted living in a simulator," in *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2016, pp. 1–5.

[127] P. Sernani, F. Dalpiaz, A. F. Dragoni, and S. Brinkkemper, "Smart tales: An awareness game for ambient assisted living," in *European Conference on Ambient Intelligence*, Springer, 2015, pp. 187–204.

[128] A. Shan, X. Xu, and Z. Cheng, "Target coverage in wireless sensor networks with probabilistic sensors," *Sensors*, vol. 16, no. 9, p. 1372, 2016.

[129] J. L. Silva, J. C. Campos, and M. D. Harrison, "Prototyping and analysing ubiquitous computing environments using multiple layers," *International Journal of Human-Computer Studies*, vol. 72, no. 5, pp. 488–506, 2014.

[130] *Smartcondo™*, https://www.ualberta.ca/health-sciences-education-research/simulation-experiences/hserc-spaces/smart-condo, Accessed: 2018-12-09.

[131] D. Spoladore, S. Arlati, and M. Sacco, "Semantic and virtual reality-enhanced configuration of domestic environments: The smart home simulator," *Mobile Information Systems*, vol. 2017, 2017.

[132] C. Stahl, J. Frey, J. Alexandersson, and B. Brandherm, "Synchronized realities," *Journal of Ambient Intelligence and Smart Environments*, vol. 3, no. 1, pp. 13–25, 2011.

[133] C. Stahl and T. Schwartz, "Modeling and simulating assistive environments in 3-d with the yamamoto toolkit," in *2010 International Conference on Indoor Positioning and Indoor Navigation*, IEEE, 2010, pp. 1–6.

[134] J.-M. Su and C.-F. Huang, "An easy-to-use 3d visualization system for planning context-aware applications in smart buildings," *Computer Standards & Interfaces*, vol. 36, no. 2, pp. 312–326, 2014.

[135] S. Sundresh, W. Kim, and G. Agha, "SENS: A sensor, environment and network simulator," in *Proceedings of the 37th Annual Symposium on Simulation*, ser. ANSS '04, Washington, DC, USA: IEEE Computer Society, 2004, pp. 221–, ISBN: 0-7695-2110-X. [Online]. Available: http://dl.acm.org/citation.cfm?id=987679.987699.

[136] J. Synnott, L. Chen, C. Nugent, and G. Moore, "Ie sim–a flexible tool for the simulation of data generated within intelligent environments," in *International Joint Conference on Ambient Intelligence*, Springer, 2012, pp. 373–378.

[137] J. Synnott, L. Chen, C. D. Nugent, and G. Moore, "The creation of simulated activity datasets using a graphical intelligent environment simulation tool," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2014, pp. 4143–4146.

[138] J. Synnott, C. Nugent, and P. Jeffers, "Simulation of smart home activity datasets," *Sensors*, vol. 15, no. 6, pp. 14 162–14 179, 2015.

[139] B. L. Thomas, A. S. Crandall, and D. J. Cook, "A genetic algorithm approach to motion sensor placement in smart environments," *Journal of reliable intelligent environments*, vol. 2, no. 1, pp. 3–16, 2016.

[140] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[141] A. Trivedi, J. Gummeson, D. Irwin, D. Ganesan, and P. Shenoy, "Ischedule: Campus-scale hvac scheduling via mobile wifi monitoring," in *Proc. 8th International Conference on Future Energy Systems (e-Energy)*, ACM, 2017, pp. 132–142.

[142] *Vaalid project : Accessibility and usability validation framework for aal interaction design process*, https://pm4health.com/projects/vaalid/, Accessed: 2021-08-17.

[143] T. Van Nguyen, J. G. Kim, and D. Choi, "Iss: The interactive smart home simulator," in *2009 11th international conference on advanced communication technology*, IEEE, vol. 3, 2009, pp. 1828–1833.

[144] A. Vasilateanu, I. A. Popescu, A. S. Cergan, and N. Goga, "Smart home simulation system," in *2016 IEEE International Symposium on Systems Engineering (ISSE)*, IEEE, 2016, pp. 1–5.

[145] C. Velasquez, C. Soares, R. Morla, R. S. Moreira, J. Torres, and P. Sobral, "A 3d simulation framework for safe ambient-assisted home care," in *Proceedings of the fifth international conference on mobile ubiquitous computing, systems, services and technologies*, 2011, pp. 61–66.

[146] F. Veronese, D. Proserpio, S. Comai, M. Matteucci, and F. Salice, "Sharon: A simulator of human activities, routines and needs.," in *AAATE Conf.*, 2015, pp. 560–566.

[147] I. Vlasenko, I. Nikolaidis, and E. Stroulia, "The smart-condo: Optimizing sensor placement for indoor localization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 436–453, 2014.

[148] M. Volpp *et al.*, "Meta-learning acquisition functions for transfer learning in bayesian optimization," *arXiv preprint arXiv:1904.02642*, 2019.

[149] Z. Wang, H. Xie, Z. Hu, D. Li, J. Wang, and W. Liang, "Node coverage optimization algorithm for wireless sensor networks based on improved grey wolf optimizer," *Journal of Algorithms & Computational Technology*, vol. 13, p. 1 748 302 619 889 498, 2019.

[150] D. Weitz, D. Marıa, F. Lianza, N. Schmidt, and J. P. Nant, "Smart home simulation model for synthetic sensor datasets generation," *Sistemas y Telemática*, vol. 14, no. 39, pp. 71–84, 2016.

[151] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[152] H. Wu, Z. Liu, J. Hu, and W. Yin, "Sensor placement optimization for critical-grid coverage problem of indoor positioning," *International Journal of Distributed Sensor Networks*, vol. 16, no. 12, p. 1 550 147 720 979 922, 2020.

[153] Z. Xu, Y. Guo, and J. H. Saleh, "Multi-objective optimization for sensor placement: An integrated combinatorial approach with reduced order model and gaussian process," *Measurement*, vol. 187, p. 110 370, 2022.

[154] C. Yang, "An adaptive sensor placement algorithm for structural health monitoring based on multi-objective iterative optimization using weight factor updating," *Mechanical Systems and Signal Processing*, vol. 151, p. 107 363, 2021.

[155] C. Yang, X. Hou, and S. Chang, "A synchronous placement and size-based multi-objective optimization method for heat dissipation design on antenna module of space solar power satellite," *Sustainable Energy Technologies and Assessments*, vol. 45, p. 101 183, 2021.

[156] C. Yang, K. Liang, and X. Zhang, "Strategy for sensor number determination and placement optimization with incomplete information based on interval possibility model and clustering avoidance distribution index," *Computer Methods in Applied Mechanics and Engineering*, vol. 366, p. 113 042, 2020.

[157] H. Ying and S. Lee, "A mask r-cnn based approach to automatically construct as-is ifc bim objects from digital images," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, IAARC Publications, vol. 36, 2019, pp. 764–771.

[158] Y. Yoon and Y.-H. Kim, "An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1473–1483, 2013.

[159] X. Yu, K. Ergun, L. Cherkasova, and T. Š. Rosing, "Optimizing sensor deployment and maintenance costs for large-scale environmental monitoring," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3918–3930, 2020.

[160] Y. Zhan and H. Haddadi, "Mosen: Sensor network optimization in multiple-occupancy smart homes," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, IEEE, 2021, pp. 384–388.

[161] Z. Zhang *et al.*, "Srengine: An osgi-based context-aware inference engine for smart room," in *2011 6th International Conference on Pervasive Computing and Applications*, IEEE, 2011, pp. 267–271.

[162] Y. Zhao, F. F. Pour, S. Golestan, and E. Stroulia, "Bim sim 3 d: Multi-agent human activity simulation in indoor spaces," in *Proceedings of the 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems*, IEEE Press, 2019, pp. 18–24.

[163] D. P. Zhou, Q. Hu, and C. Tomlin, "Quantitative comparison of data-driven and physics-based models for commercial building hvac systems," in *Proc. American Control Conference*, 2017, pp. 2900–2906.