Article

OpenSHS: Open Smart Home Simulator

Nasser Alshammari [1,2]*, Talal Alshammari [1,3], Mohamed Sedky [1], Justin Champion [1], and Carolin Bauer [1]

1 Staffordshire University, College Road, ST4 2DE Stoke-on-Trent, UK; {nasser.alshammari, talal.alshammari}@research.staffs.ac.uk, {m.h.sedky, j.j.champion, c.i.bauer}@staffs.ac.uk
2 College of Information and Computer Science, Aljouf University, Sakaka, Saudi Arabia; nashamri@ju.edu.sa
3 College of Computer Science and Engineering, University of Hail, Hail, Saudi Arabia; talal.alshammari@uoh.edu.sa
* Correspondence: nasser.alshammari@research.staffs.ac.uk; Tel.: +44-7502238824

Academic Editor: name
Version April 13, 2017 submitted to Sensors; Typeset by LATEX using class file mdpi.cls

Abstract: This paper develops a new hybrid, open-source, cross-platform 3D smart home simulator, OpenSHS, for dataset generation. OpenSHS offers an opportunity for researchers in the field of Internet of Things (IoT), machine learning and smart home simulation to test and evaluate their models. Following a hybrid approach, a OpenSHS combines advantages from both interactive and model-based approaches. This approach reduces the time and efforts required to generate simulated smart home datasets. We have designed a replication algorithm for extending and expanding a dataset. A small sample dataset produced, by OpenSHS, can be extended without affecting the logical order of the events. The replication provides a solution for generating large representative smart home datasets. We have built an extensible library of smart devices that facilitates the simulation of current and future smart home environments. Our tool divides the dataset generation process into three distinct phases: first design, the researcher designs the initial virtual environment by building the home, importing smart devices and creating contexts; second simulation, the participant simulates his/her context-specific events; and third aggregation, the researcher applies the replication algorithm to generate the final dataset. We conducted a study to asses the ease of use of our tool on the System Usability Scale (SUS).

Keywords: smart home; simulation; internet of things; machine learning; visualisation

---

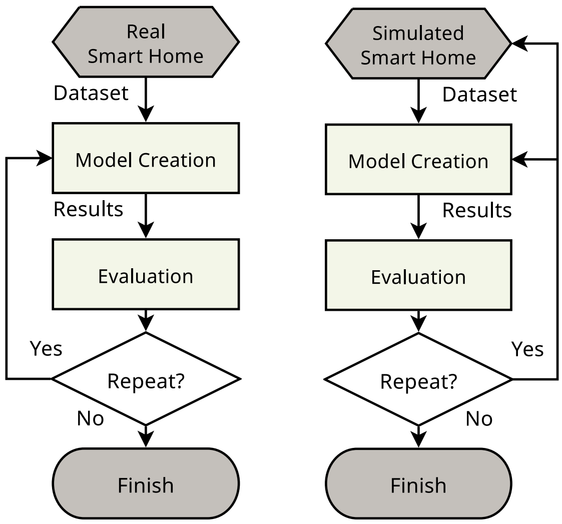
1. Introduction

With the recent rise of the Internet of Things, analysing data captured from smart homes is gaining more research interest. Moreover, developing intelligent Machine Learning techniques that are able to provide services to the smart home inhabitants are becoming popular research areas.

Intelligent services, such as classification and recognition of Activities of Daily Living (ADL) and anomaly detection in elderly daily behaviour, require the existence of good datasets that enable testing and validation of the results [1–4]. The medical field also recognised the importance of analysing ADLs and how these techniques are effective at detecting medical conditions for the patients [5]. These research projects require either real or synthetic datasets that are representative of the scenarios captured from a smart home. However, the cost to build real smart homes and the collection of datasets for such scenarios is expensive and sometimes infeasible to many projects [4,6–9]. Moreover, several issues face the researchers before actually building the smart home such as finding the optimal placement of the sensors [10], lack of flexibility [9,11], finding appropriate participants [4,7], and privacy and ethical issues [12].

Even though there exist real smart home datasets [13–15], sometimes they do not meet the needs of the conducted research project. Such as, the need to add more sensors or to control the type of the generated scenarios. Very few of such datasets record the readings of the sensors in real-time and provide a detailed time-stamped field like the ARAS dataset [14]. Moreover, preparing a real dataset could be a laborious task and if not done with care, it could lead to producing erroneous output.

When building real smart home testbeds, there are several challenges facing the preparation of real datasets. One challenge is having a robust and continuous capturing mechanism for the sensors' data. Another challenge is following an appropriate annotation method for the inhabitants' activities. A number of methodologies are followed to capture the interactions of the inhabitants and the smart home. Generally, these methods can be divided into two groups: inhabitants-reported methods and device-reported methods [6]. The inhabitants-reported methods could be done with logs or questionnaires. The usage of such methods could be problematic and could lead to erroneous recordings due to inhabitants' mistakes [16]. Moreover, annotating streaming data in real-time is a process prone to mistakes because of the possibility that the participants might forget updating the annotation of current activity. Also, there could be inconsistencies when updating the annotations during the transition from one activity to another. If annotation is a post processing step and done after the fact, this can lead to inaccurate labelling. Therefore, recent research moved to device-reported methods because of the unobtrusive and transparent nature of these methods, especially in elderly care research. Examples of device-reported methods are smart wearable/stationary devices, and computer vision based solutions.



(a) A real testbed. **(b)** A simulated testbed.

Figure 1. The workflow with real and simulated smart homes testbeds.

The existence of a dataset simulation tool overcomes the drawbacks/challenges of generating real datasets. When developing Machine Learning models, targeting specific functionalities, researchers rely on the existence of good representative datasets. A common practice in Machine Learning is to divide the dataset into two parts, training and testing. The model creation starts by initialising its parameters and training on a portion of the dataset. Then, the model will be tested on another portion of the same dataset and its results will be evaluated. The results of the evaluation could reveal the need to redesign the smart home by adding or removing smart devices, or changing the scenarios generated, etc. In the case of a real smart home, if the results revealed the need to change something, this is usually a costly and infeasible choice to make. Therefore, the researcher could only be able to tweak the model parameters as shown in figure 1a. On the other hand, with a simulated smart home, this can be easily done and the researcher can go back and modify the smart home design as shown in figure 1b.

The approaches for the smart home simulation tools can be divided to model-based and interactive approaches. The model-based approaches use statistical models to generate datasets while the interactive approaches relies on real-time capturing of fine grained activities using an avatar controlled by a human/simulated participant. Each approach has its advantages and disadvantages.

From what we mentioned earlier, it is apparent that the virtual simulation tool should offer far greater flexibility and lower cost than conducting an actual and physical smart home simulation [6]. The recent advances in computer graphics can provide an immersive and semi-realistic experiences that could come close to the real experience, such as Virtual Reality (VR) technologies. The simulation tool should also be open and easily available to both, the researchers and the test subjects.

Although there are some research efforts available in the literature for smart home simulation tools, they suffer from a number of limitations. The majority of these tools are not available in the public domain as an open-source project, or limited to a specific platform. In addition, most of the publicly available simulation tools offer lack the flexibility to add and customise new sensors or devices.

When generating datasets, the model-based approaches are capable of generating bigger datasets but the granularity of captured interactions are not as fine as the interactive approaches. However, the interactive approaches usually take longer time to produce the datasets as they capture the interactions in real-time.

In this paper, we present the architecture and implementation of OpenSHS, a novel smart home simulation tool. OpenSHS is a new hybrid, open-source, cross-platform 3D smart home simulator for dataset generation. Its significant contribution is that OpenSHS offers an opportunity for researchers in the field of Internet of Things (IoT) and Machine Learning to produce and share their smart home datasets as well as testing, comparing and evaluating their models objectively. Following a hybrid approach, OpenSHS combines advantages from both interactive and model-based approaches. This approach reduces the time and efforts required to generate simulated smart home datasets. OpenSHS includes an extensible library of smart devices that facilitates the simulation of current and future smart home environments. We have designed a replication algorithm for extending and expanding a dataset. A small sample dataset produced, by OpenSHS, can be extended without affecting the logical order of the events. The replication provides a solution for generating large representative smart home datasets. Moreover, OpenSHS offers a feature to shortening and extending the duration of the generated activities.

The rest of this paper is structured as follows: the following section reviews existing smart simulation tools and datasets. Section 3 presents the architecture of OpenSHS and its implementation. Section 4 presents a usability study and section 5 presents the advantages of OpenSHS. Followed by section 6 which lists the limitations of OpenSHS and the planned future work for this project, the paper concludes.

2. Related Work

The literature is rich with efforts that focus on generating datasets for smart home applications. These efforts can be classified into two main categories, datasets generated either from real smart homes testbeds or using smart home simulation tools.

2.1. Real Smart Home Testbeds

One of the recent projects for building real smart homes for research purposes was the work carried out by the Centre for Advanced Studies in Adaptive Systems (CASAS) [17] where they created a toolkit called 'smart home in a box' which is easily installed in a home to make it able to provide smart services. The components of the toolkit are small and can fit in a single box. The toolkit has been installed in 32 homes to capture the participants interactions. The datasets are publicly available online [18].

The TigerPlace [19] project is an effort to tackle the growing ageing population. Using passive sensor networks implemented in 17 apartments within an eldercare establishment. The sensors include motion sensors, proximity sensors, pressure sensors and other types. The data collection took more than two years for some of the testbeds.

SmartLab [20] is a smart laboratory devised to conduct experiments in smart living environments to assess in the development of independent living technologies. The laboratory has many types of sensors such as, pressure, passive infrared (PIR), and contact sensors. The participants interactions with SmartLab are captured in an XML-based schema called homeML [21].

The Ubiquitous Home [22] is a smart home that was built to study context-aware services by providing cameras, microphones, pressure sensors, accelerometers, and other sensor technologies. The home consists of several rooms equipped with different sensors. To provide contextual services to each resident, the Ubiquitous home recognises the resident by providing Radio-Frequency Identification (RFID) tags and by utilising the installed cameras.

PlaceLab [23] is a 1000 sq.ft. smart apartment that has several rooms. The apartment has many sensors distributed throughout each room, such as electrical current sensors, humidity sensors, light sensors, water flow sensors, etc. Volunteering participant can live in PlaceLab to generate a dataset of their interaction and behaviour. The project produced several datasets for different scenarios [24].

HomeLab [25] is a smart home equipped with 34 cameras distributed around several rooms. The project has an observation room that allows the researcher to observe and monitor the conducted experiments. HomeLab aims to provide datasets to study human behaviour in smart environments and investigate technology acceptance and usability.

The GatorTech smart home [26] is a programmable and customisable smart home that focuses on studying the ability of pervasive computing systems to evolve and adapt for future advances in sensors technology.

2.2. Smart Home Simulation Tools

Smart home simulation tools can be categorised into two main approaches, according to Synnott *et al.* [6], model-based and interactive approaches.

2.2.1. Model-Based Approach

This approach uses pre-defined models of activities to generate synthetic data. These models specify the order of events, the probability of their occurrence, and the duration of each activity. This approach facilitates the generation of large datasets in a short period of time. However, the downside of this approach is that it cannot capture intricate interactions or unexpected accidents that are common in real homes. An example of such approach is the work done by Mendez-Vazquez *et al.* [7].

PerSim 3D [27] is a tool to simulate and model user activities in smart spaces. The aim of this tool is to generate realistic datasets for complex scenarios of the inhabitants activities. The tool provides a Graphical User Interface (GUI) for visualising the activities in 3D. The researcher can define contexts and set ranges of acceptable values for the sensors in the smart home. However, the tool is not available freely in the public domain.

SIMACT [28] is a 3D smart home simulator designed for activity recognition. SIMACT has many pre-recorded scenarios that were captured from clinical experiments, which can be used to generate datasets for the recognition of ADLs. SIMACT is a 3D open-source and cross-platform project developed with Java and uses Java Monkey Engine (JME) [29] as its 3D engine.

DiaSim [30] is a simulator developed using Java for pervasive computing systems that can deal with heterogeneous smart home devices. It has a scenario editor that allows the researcher to build the virtual environment to simulate a certain scenario.

The Contex-Aware Simulation System (CASS) [31] is another tool that aims at generating context information and test context-awareness applications in a virtual smart home. CASS allows the researcher to set rules for different contexts. A rule can be, for example, turn the air conditioner if a room reaches a specific temperature. The tool is able to detect conflicts between the rules of the pre-defined contextual scenarios and determine the best positioning of the sensors. CASS provides a 2D visualisation GUI for the virtual smart home.

The Context-Awareness Simulation Toolkit (CAST) [32] is a simulation tool designed to test context-awareness applications and provides visualisations of different contexts. The tool generates context information from the users in a virtual smart home. CAST was developed with the proprietary technology Adobe Flash and is not available in the public domain.

2.2.2. Interactive Approach

Contrary to the previous approach, the interactive approach can capture more interesting interactions and fine details. This approach relies on having an avatar that can be controlled by a researcher, human participant or simulated participant. The avatar moves and interacts with the virtual environment which has virtual sensors and/or actuators. The interactions could be done passively or actively. One example of passive interactions is having a virtual pressure sensor installed on the floor and when the avatar walks on it, the sensor should detect this and emit a signal. Active interactions involve actions such as opening a door or turning the light on or off. The disadvantage of this approach, however, is that it is a time-consuming approach to generate sufficient datasets as all interactions must be captured in real-time.

Park *et al.* [33] presented a virtual space simulator that is able to generate inhabitants data for classifications problems. In order to model inhabitant activities in 3D, The simulator was built using Unity3D [34].

The intelligent environment simulation (IE Sim) [35] is a tool used to generate simulated datasets that captures normal and abnormal ADLs of inhabitants. It allows the researcher to design smart homes by providing a 2D graphical top-view of the floor plan. The researcher can add different types of sensors such as, a temperature sensors, pressure sensors, etc. Then, using an avatar, the simulation can be conducted to capture ADLs. The format of the generated dataset is homeML [21]. Up to the knowledge of the authors, IE Sim is not available in the public domain.

Ariani *et al.* [36] developed a smart home simulation tool that uses ambient sensors to capture the interactions of the inhabitants. The tool has a map editor that allows the researcher to design a floor plan for a smart home by drawing shapes on a 2D canvas. Then, the researcher can add ambient sensors to the virtual home. The tool can simulate binary motion detectors and binary pressure sensors. To simulate the activities and interactions in the smart home, they used the A* pathfinding algorithm [37], to simulate the movement of the inhabitants. During the simulation, all interactions are sampled at 5 Hz and stored into an XML file.

UbiREAL [38] is a Java based simulation tool that allows the development of ubiquitous applications in a 3D virtual smart space. It allows the researcher to simulate the operations and communications of the smart devices at the network level.

V-PlaceSims [39] is a simulation tool that allows a smart home designer to design a smart home from a floor plan. Then, allows multiple users to interact with this environment through a web interface. The focus of this tool is the improvement of the designs and management of the smart home.

In addition to the outlined above simulation tools, there are other commercial simulation tools targeting the industry such as [40–42].

Generally, the model-based approach allows the researcher to generate large datasets in short simulation time but sacrifices the granularity of capturing realistic interactions. On the other hand, the interactive approach captures these realistic interactions but sacrifices the short and quick simulation time and therefore, the generated datasets are usually smaller than the ones generated by model-based approach.

2.3. Analysis

Synnott *et al.* [6] identified several challenges that face the smart home simulation research. One of the key challenges is that many of the available simulation tools [9,11,31,38,39,43–45] focus on testing applications that provide context-awareness and visualisation rather than focusing on generating representative datasets. Few of the available tools do focus on generating datasets [1,12,46,47]. Another key challenge is to have the flexibility and scalability to add new/customised types of smart devices, change their generated output(s), change their positions within the smart home, etc. The multiple inhabitants support is also one of the limitations facing the currently available tools as this feature is known to be difficult to implement [6].

The review of available smart home simulation tools reveals that the majority of the reported work lacks the openness and availability of the software implementation, which hinders their benefit to the wider research community. Moreover, less than half of the reviewed tools (10 out of 23) does not support multiple operating systems which can be an issue when working with research teams and/or test subjects. Table 1 shows our analysis of the available simulation tools. SIMACT [28] is the only open-source and cross-platform simulation tool available, however, the data generation approach used in that tool is based on a pre-defined script that the researcher plays back within the 3D simulation view.

Apart from the work by [48], this analysis shows that none of the reviewed simulation tools follows a hybrid approach i.e. a tool that combines the ability of model-based tools to generate large datasets in a reasonable time while keeping the fine-grained interactions that are exhibited by the interactive tools.

Our review shows that fewer simulation tools focus on generating datasets while the majority of the reviewed tools focus on visualisation and context-awareness applications.

Rec

Table 1. Analysis of smart home simulation tools.

Tool/author(s)	Date	Open-source	3D	Cross-platform	Approach	Focus	Multi-inhabitants
Park *et al.* [33]	2015	No	Yes	Yes	Interactive	Visualisation	No
PerSim 3D [27]	2015	No	Yes	Yes	Model-based	Dataset generation	No
IE sim extended [48]	2015	No	No	No	Interactive	Dataset generation	No
IE sim [35]	2014	No	No	No	Interactive	Dataset generation	No
Kormányos *et al.* [49]	2013	No	No	No	Model-based	Visualisation	No
Ariani *et al.* [36]	2013	No	No	No	Interactive	Dataset generation	Yes
Fu *et al.* [11]	2011	No	No	Yes	Interactive	Visualisation	Yes
Jahromi *et al.* [50]	2011	No	No	No	Model-based	Visualisation	No
Buchmayr *et al.* [1]	2011	No	No	No	Interactive	Dataset generation	No
SimCon [46]	2010	No	Yes	Yes	Interactive	Dataset generation	No
YAMAMOTO [43]	2010	No	No	Not reported	Interactive	Visualisation	No
SIMACT [28]	2010	Yes	Yes	Yes	Model-based	Visualisation	No
Poland *et al.* [12]	2009	No	Yes	Yes	Interactive	Dataset generation	No
ISS [51]	2009	No	No	No	Interactive	Visualisation	Yes
DiaSim [30]	2009	No	No	Yes	Model-based	Visualisation	No
V-PlaceSims [39]	2008	No	Yes	No	Interactive	Visualisation	Yes
Armac *et al.* [9]	2007	Not reported	No	Not reported	Interactive	Visualisation	Yes
CASS [31]	2007	No	No	No	Model-based	Visualisation	Yes
Krzyska *et al.* [47]	2006	No	No	Yes	Interactive	Dataset generation	Yes
CAST [32]	2006	No	No	No	Model-based	Visualisation	No
UbiREAL [38]	2006	No	No	Yes	Interactive	Visualisation	Yes
TATUS [44]	2005	No	Yes	Not reported	Interactive	Visualisation	Yes
UbiWise [45]	2002	No	Yes	Yes	Interactive	Visualisation	Yes

3. OpenSHS Architecture and Implementation

This paper proposes a new hybrid, open-source, and cross-platform 3D smart home simulation tool for dataset generation, OpenSHS [52], which is downloadable from http://www.openshs.org under the GPLv2 license [53]. OpenSHS tries to provide a solution to the issues and challenges identified by Synnott *et al.* [6]. OpenSHS follows a hybrid approach, to generate datasets, combining the advantages of both model-based and interactive approaches. This section presents the architecture of OpenSHS and the technical details of its implementation, which is based on Blender [54] and Python. In this section, we will refer to two entities, the researcher and the participant. The researcher is responsible for most of the work with OpenSHS. The participant is any person volunteering to simulate their own activities.

Working with OpenSHS can be divided into three main phases: design phase, simulation phase, and aggregation phase. The following subsections will describe each phase.

3.1. Design Phase

In this phase, as shown in figure 2, the researcher builds the virtual environment, imports the smart devices, assign activities' labels and design the contexts.

3.1.1. Designing Floor Plan

The researcher designs the 3D floor plan by using Blender which allows the researcher to easily model the house architecture and control different aspect such as the dimensions and the square footage. In this step, the number of rooms and the overall architecture of the home is defined according to the requirements of the experiment.

3.1.2. Importing Smart Devices

After the design of the floor plan, the smart devices can be imported into the smart home from the smart devices library, offered by OpenSHS. The current version of OpenSHS includes the following list of active and passive devices/sensors:

- Pressure sensors (e.g. activated carpet, bed, couch, etc.),
- Door sensors,
- Lock devices,
- Appliance switches (TV, oven, fridge, etc.),
- Light controllers.

The Smart devices library is designed to be a repository of different types of smart devices and sensors. This list is extensible as it is programmed with Python. Moreover, the researcher can build a customised sensor/device.

3.1.3. Assigning Activity Labels

OpenSHS enables the researcher to define unlimited number of activity labels. The researcher decides how many labels are needed according to their experiment's requirements. Figure 4 shows a prototype where the researcher identified five labels. Namely, 'sleep', 'eat', 'personal', 'work' and 'other'. This list of activity labels represents a sample of activities, which the researchers can tailor it to their needs.

3.1.4. Designing Contexts

After designing the smart home model, the researcher designs the contexts to be simulated. The contexts are specific time frames that the researcher is interested to simulate e.g. morning, afternoon, evening contexts. For instance, if the researcher aims to simulate the activities that a participant performs when he/she comes back from work during a weekday, then the researcher will design a context for that time period. Finally, the researcher specifies the initial states of the devices for each context.

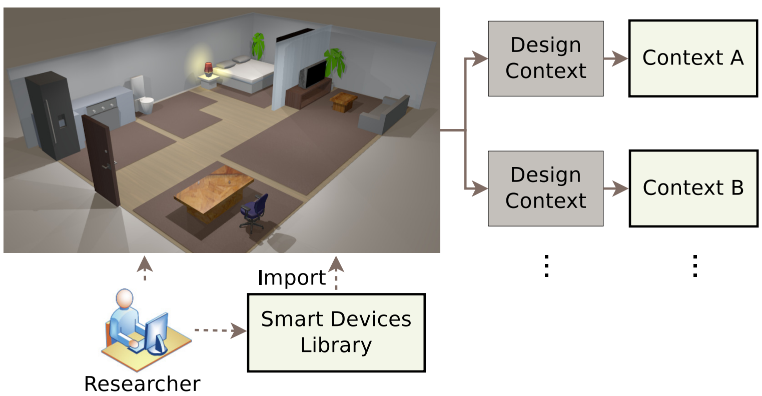


Figure 2. The design phase.

3.2. Simulation Phase

Figure 3 shows the overall architecture of the simulation phase. The researcher starts the tool from the OpenSHS interface module which allows the researcher to specify which context to simulate. Each context has a default starting date and time and the researcher can adjust the date and time if he/she wants. Every context has a default state for the sensors and for the 3D position of the avatar. Then, the participant starts simulating his/her ADLs in that context. During the simulation time, the sensors' outputs and the state of different devices are captured and stored in a temporary dataset.

OpenSHS adapts a sampling rate of one second by default, which the researcher can re-configure as required. Once the participant finishes a simulation, the application control is sent back to the main module to start the simulation of another context.

The simulation phase aims to capture the granularity of the participants' realistic interactions. However, capturing these fine-grained activities in extended periods of time adds a burden on the participant(s) and sometimes becomes infeasible. OpenSHS offers a solution that mitigates this issue by adapting a fast-forwarding mechanism.

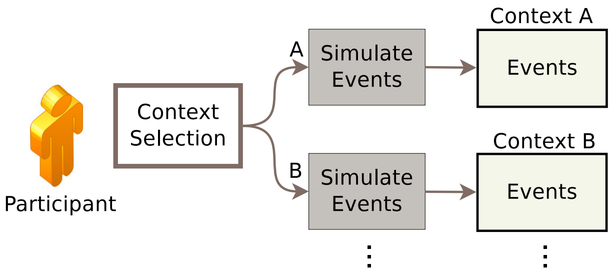


Figure 3. The simulation phase.

3.2.1. Fast-Forwarding

OpenSHS allows the participant to control the time span of a certain activity, fast-forwarding. For example, if the participant wants to watch the TV for a period of time and does not want to perform the whole activity in real-time (since there are no changes in the readings of the home's sensors), the participant can initiate that activity and spawn a dialog to specify how long this activity lasts. This feature allows the simulation process to be quick and streamlined. The tool will simply copy and repeat the existing state of all sensors and devices during the specified time period. Figure 4 shows the activity fast-forwarding dialog during a simulation.

Figure 4. The activity selection and fast-forwarding dialog.

3.2.2. Activities Labelling

The researcher is responsible for familiarising the participant with the available activity labels to choose from. During a simulation and before transitioning from one activity to another, the participant will spawn the activity dialog shown in figure 4 to choose the new activity from the available list. To ensure a clean transition from one activity to another, OpenSHS will not commit the new label at the exact moment of choosing the new label. Instead, the new label will be committed when a sensor changes its state. For example, in figure 6 the transition from the first activity (sleep) to the second (personal) is committed to the dataset when the sensor `bedroomLight` changes its state even though the participant did change the label a couple of seconds earlier.

3.3. Aggregation Phase

After performing the simulation by the participants, the researcher can aggregate the participants' generated sample activities i.e. events, in order to produce the final dataset. The results of the simulation phase form a pool of sample activities for each context. The aggregation phase aims to provide a solution for the generation of large datasets in short simulation time. Hence, this work develops an algorithm that replicates the output of the simulation phase by drawing appropriate samples for each designated context.

This feature encapsulates model-based approach advantage with the interactive approach adapted by the simulation phase, which allows OpenSHS to combine the advantages of both approaches, a hybrid approach.

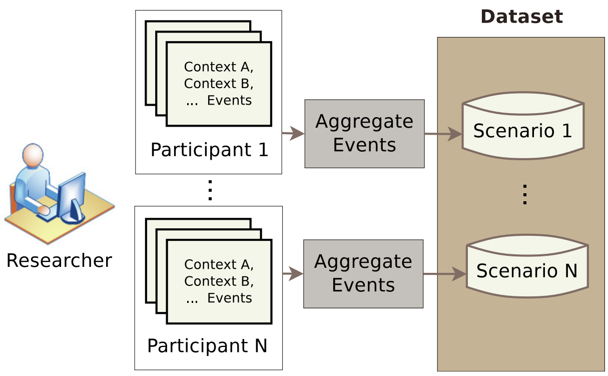


Figure 5. The aggregation phase.

3.3.1. Events Replication

It was clear from the beginning of the development of this project that it is not feasible for a participant to sit down and simulate his/her ADLs for a whole day. Moreover, we wanted to capture the interactions between the inhabitant and the smart home in real-time. At the same time, we wanted the process to be less tedious and streamlined as much as possible. These requirements brought up the concept of real-time context simulations. Instead of having the user simulating his/her ADLs for long periods, the user simulates only a specific context in real-time. For example, let us assume we are interested in an 'early morning' context and we want to capture the activities that the inhabitant is doing in this time frame, such as, what is usually done in the weekdays compared to the weekends in the same context (The 'early morning' context). The user will only perform sample simulations of different events in real-time. The number of samples simulated, the richer the generated dataset will be.

To gain more insight of how OpenSHS works, we have built a virtual smart home environment consisting of a bed room, a living room, a bath room, a kitchen, and an office. Each room is equipped with several sensors totalling twenty-nine sensors of different types. The sensors are binary and they are either on or off at any given time step.

The result of performing a context simulation can be illustrated by figure 6. The sample consists of three activity labels, namely 'sleep', 'personal', and 'other'. Each activity label corresponds to a set of sensors' readings. The sensors' readings in the previous figure are readings of binary sensors and the small circles correspond to an 'ON-state' of that sensor.

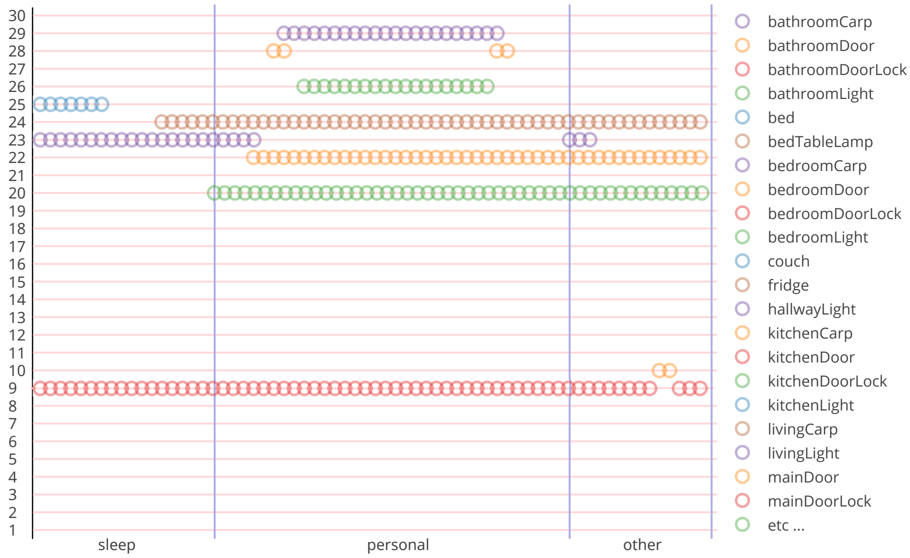


Figure 6. Twenty nine binary sensors' output and the corresponding activity labels.

Table 2. A set of recorded samples for a certain context.

SAMPLE 1	sleep	personal	work	eat	other
SAMPLE 2	sleep	personal	other		
SAMPLE 3	sleep	personal	other		
SAMPLE 4	sleep	eat	personal	other	
SAMPLE 5	sleep	eat	personal	other	

It is not realistic to aggregate the final dataset by trivially duplicating the contexts samples. There is a need for an algorithm that can replicate the recorded samples to generate a larger dataset. We have designed a replication algorithm for extending and expanding the recorded samples. A small number of simulated events can be extended without affecting their logical order.

To explain the replication algorithm, it is best illustrated by an example. Table 2 shows a set of five samples with their activity labels for a certain context. The first sample has five activities and the second sample has three activities and so on. When the researcher aggregates the final dataset, the samples of every context are grouped by the number of activities in each sample. So for the previous example, sample 1 will be in one group, sample 2 and 3 will be in a second group, and sample 4 and 5 will be in a third group. Then, a random group will be chosen and from that group, a sample will be drawn for each activity. For example, let us take the second group which contains sample 2 and 3. The number of activities in this group is three. So, for the first activity we will either pick the 'sleep' activity from sample 2 or the 'sleep' activity from sample 3. The same procedure will be done for the second and third activities. The output will resemble what is shown in table 3.

Table 3. Ten replicated copies based on the samples from table 2.

i	Activity 1	Activity 2	Activity 3	Activity 4	Activity 5
1.	**sample 1** sleep	**sample 1** personal	**sample 1** work	**sample 1** eat	**sample 1** other
2.	**sample 4** sleep	**sample 5** eat	**sample 5** personal	**sample 4** other	
3.	**sample 3** sleep	**sample 3** personal	**sample 2** other		
4.	**sample 3** sleep	**sample 3** personal	**sample 2** other		
5.	**sample 5** sleep	**sample 4** eat	**sample 5** personal	**sample 5** other	
6.	**sample 1** sleep	**sample 1** personal	**sample 1** work	**sample 1** eat	**sample 1** other
7.	**sample 2** sleep	**sample 2** personal	**sample 2** other		
8.	**sample 5** sleep	**sample 5** eat	**sample 5** personal	**sample 5** other	
9.	**sample 4** sleep	**sample 4** eat	**sample 4** personal	**sample 5** other	
10.	**sample 2** sleep	**sample 2** personal	**sample 2** other		

The context samples shown in table 2 will produce 25 unique replicated copies. In general, the number of unique replicated copies for a single context can be calculated by the equation 1. Let $\mathcal{G}$ denotes the number of the groups of unique length of activities, and let $\mathcal{S}_g$ denotes the number of samples for the group g, and let $\mathcal{A}$ denotes the number of activities within a sample $\mathcal{S}_g$. The total number of unique replicated copies $\mathcal{R}$ is:

$$\mathcal{R} = \sum_{g=1}^{\mathcal{G}} \mathcal{S}_g^{\mathcal{A}} \tag{1}$$

OpenSHS can modify the original duration of a performed activity by shortening and/or expanding it. To preserve the structure of a certain activity, we look for the longest steady and unchanged sequence of readings. Then, our algorithm randomly chooses a new duration for this sequence. The new modified sequence length can vary between 5% of the original sequence length, up to its full length. The researcher can use this feature by passing the `variable-activities` option to the aggregation parameters as will be shown next.

The researcher can configure a number of parameters to control the generated output such as:

- `days`: the number of days to be generated,
- `start-date`: specifies the starting date for the dataset,
- `time-margin`: the variability of the starting time for the replicated events. For example, assuming we have a sample that was recorded at 7:30am and we specified the time margin to be 10 minutes. The replicated sample could start any time from 7:25am up to 7:35am,
- `variable-activities`: make the duration for each activity variable.

3.3.2. Dataset Generation

After running the aggregation algorithm, the researcher can combine all the scenarios, generated by different participants, into one final comma separated values (CSV) dataset output. Table 4 shows a sample generated dataset.

Table 4. A sample of the final dataset output.

timestamp	bedTableLamp	bed	bathroomLight	bathroomDoor	...	Activity
2016-04-01 08:00:00	0	1	0	0	...	sleep
2016-04-01 08:00:01	0	1	0	0	...	sleep
2016-04-01 08:00:02	0	1	0	0	...	sleep
2016-04-01 08:00:03	0	1	0	0	...	sleep
2016-04-01 08:00:04	1	1	0	0	...	sleep
2016-04-01 08:00:05	1	0	0	0	...	sleep
2016-04-01 08:00:06	1	0	0	1	...	personal
2016-04-01 08:00:07	1	0	0	1	...	personal
2016-04-01 08:00:08	1	0	1	1	...	personal
2016-04-01 08:00:09	1	0	1	1	...	personal
2016-04-01 08:00:10	1	0	1	1	...	personal
⋮	⋮	⋮	⋮	⋮	⋮	

The `time-margin` parameter does add a level of sophistication to the timing of the recorded activities. This useful for applications that relies heavily on the time dimension of activities, for example, in anomaly detection research.

3.4. Implementation

OpenSHS implementation relies on Blender and its game engine. Blender's game engine is programmable by Python.

3.4.1. Blender

Blender was chosen to build the majority of the simulation tool and to act as an infrastructure for OpenSHS. The reasons for this choice can be summarised as:

- **Open-source:** Blender is an open-source 3D modelling and animation software and an actively developed project by the open-source community. It allows the user to create 3D models and visual effects. The Game Engine component of Blender allows the user to build complex 3D interactive games and script them with Python which is an important feature for OpenSHS.
- **Cross-platform**: Blender is available for the three major operating systems. Namely, GNU/Linux, Microsoft Windows, and Apple macOS. Blender uses OpenGL [55] for its Game Engine which is also, a cross-platform 3D technology available for the major operating systems.
- **The Blender Game Engine**: Blender's Game Engine allowed us to add the interactivity to the simulations. The physics engine facilitates the simulation of different types of real sensors and devices. For example, blender has a 'Near' sensor which will only be activated when the 3D avatar controlled by the user is physically near other objects in the scene. Therefore, such sensor could be used to simulate a proximity sensor easily.

3.4.2. Python

The interaction with the simulation tool is done by controlling a 3D avatar that navigates the smart home space through a first-person perspective similar to most first-person games. Figure 7 shows the 3D avatar navigating the living room. Since Blender's Game Engine uses Python as a programming language, we developed all the logic and interactions between the avatar and the virtual environment with it. Moreover, all of OpenSHS modules are programmed by Python.

Figure 7. Navigating the smart home space through first-person perspective.

4. OpenSHS Usability

Measuring the usability of a software tool is a difficult and tricky task since it involves subjective qualities and depends on the context used. John Brooke [56] defines it as *"The general quality of the appropriateness to a purpose of any particular artifact"*. He developed the widely used System Usability Scale (SUS) which is a questionnaire consisting of ten questions that measures various aspects of the usability of a system. The score of SUS ranges from 0 to 100.

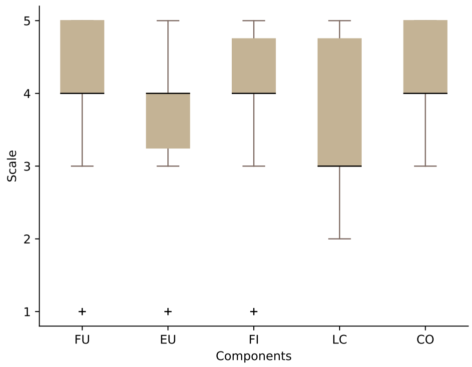
To assess OpenSHS usability, we conducted a usability study using SUS. Our sample consists of graduate students and researchers interested in smart home research. We carried out multiple sessions and in each session we started by introducing OpenSHS and then by presenting its functionalities. After that, we answered any questions the participants had in mind. Afterwards, we allowed the participants to use OpenSHS and explore its features. Finally, the participants were asked to answer few questions, such as how frequently do they use their computer on daily basis and whether they play first-person 3D video games or not. Then, the participants were asked to fill the SUS questionnaire.

We carried out two usability studies. One form the perspective of the researchers and the other from the perspective of the participants using OpenSHS. The researchers group were asked to evaluate OpenSHS usability throughout the three phases (design, simulation, aggregation). The participants group were only asked to evaluate the simulation phase.

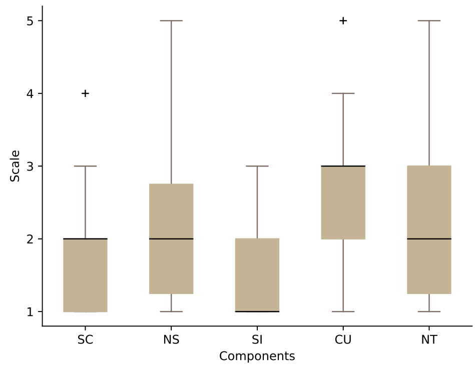
For the researchers group, we collected data from 14 researchers, 85.7% were male and 14.3% female. The average age of the researchers was 36 ($min_{age} = 31, max_{age} = 43$). All of the researchers reported that they do use their computers on a daily basis and 93% of them did play 3D first-person games. The aspects that the SUS questionnaire investigates can be summarised as:

1. **Frequent use (FU)**: I think that I would like to use this system frequently.
2. **System complexity (SC)**: I found the system unnecessarily complex.
3. **Ease of use (EU)**: I thought the system was easy to use.
4. **Need for support (NS)**: I think that I would need the support of a technical person to be able to use this system.
5. **System's functions integration (FI)**: I found the various functions in this system were well integrated.
6. **System inconsistencies (SI)**: I thought there was too much inconsistency in this system.
7. **Learning curve (LC)**: I would imagine that most people would learn to use this system very quickly.
8. **How cumbersome the system is (CU)**: I found the system very cumbersome to use.
9. **Confidence in the system (CO)**: I felt very confident using the system.
10. **Need for training before use (NT)**: I needed to learn a lot of things before I could get going with this system.

Figure 8 shows the results of our SUS questionnaire for the researchers group. The odd-numbered statements contributes positively for the overall score if the participant agrees with them (figure 8a). On the other hand, the even-numbered statements contributes negatively if the researcher agrees with them (figure 8b). Calculating the score of our sample revealed that the average SUS score of OpenSHS is 71.25 out of 100 ($score_{min} = 40, score_{max} = 85$).



(a) The positive components.

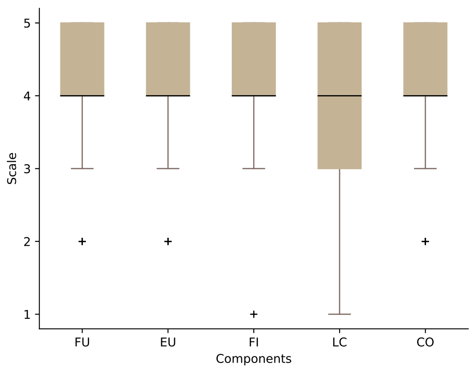


(b) The negative components.

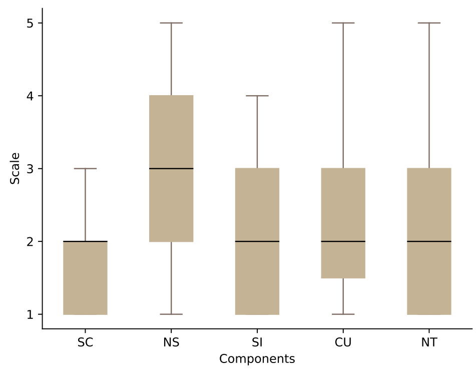
Figure 8. The result of System Usability Scale (SUS) questionnaire for the researchers group.

For the participants group, 31 participants were asked to answer the SUS questionnaire. 77.5% were male and 22.5% female and average age of the participants was 27 ($min_{age} = 21, max_{age} = 36$). 97% did play first-person games and all of the participants reported that they use their computers on daily basis. Figure 9 shows the participants group results. The SUS score for this group is 72.66 out of 100 ($score_{min} = 50, score_{max} = 87$).

The usability results for both groups are promising but, at the same time, they indicate that there is a room for improvements. Both groups agree that the learning curve (LC) component of the questionnaire needs improvement. The results also show the need for support from a technical person to use the system.



(a) The positive components.



(b) The negative components.

Figure 9. The result of System Usability Scale (SUS) questionnaire for the participants group.

5. OpenSHS Advantages

When comparing OpenSHS against the available simulation tools reviewed in Table 1, unlike the majority of such tools, our tool is based on Blender and Python which are open-source and cross-platform solutions, this offers the following benefits:

- Improving the quality of the state of the art datasets by allowing the scientific community to openly converge on standard datasets for different domains,
- Easier collaborations between research teams from around the globe,
- Faster developments and lower entry barriers,
- Facilitates the objective evaluations and assessments.

Our tool allows the simulations to be conducted in 3D from a first-person perspective. The only open-source tool we could identify in the literature was SIMACT [28]. However, SIMACT does not allow the participant to create specialised simulations. Instead, it relies on pre-recorded data captured from clinical trials.

Table 5 shows our analysis and comparison of OpenSHS with the existing simulation tools that are focusing on dataset generation. IE sim [35] was extended to use a probabilistic model (Poisson distribution) to augment the interactively recorded data by IE sim. Therefore, the extended version of IE sim uses a hybrid approach. However, IE sim is a 2D simulator which takes part of the realism out of the simulation. This might be a problem when 3D motion data is important to the researcher, for example in anomaly detection algorithms, as identified by [48].

The fast-forwarding feature makes the simulation less cumbersome especially when the simulation has long periods of inactivity as in eldercare research. This feature is relevant to interactive and hybrid approaches.

Table 5. Comparing OpenSHS with other smart home simulation tools for dataset generation.

Tool/author(s)	Date	Open-source	3D	Cross-platform	Approach	Multi-inhabitants	Fast-forwarding
OpenSHS	**2017**	**Yes**	**Yes**	**Yes**	**Hybrid**	**Partially**	**Yes**
PerSim 3D [27]	2015	No	Yes	Yes	Model-based	No	Not applicable
IE sim extended [48]	2015	No	No	No	Hybrid	No	Yes
IE sim [35]	2014	No	No	No	Interactive	No	No
Ariani *et al.* [36]	2013	No	No	No	Interactive	Yes	No
Buchmayr *et al.* [1]	2011	No	No	No	Interactive	No	No
SimCon [46]	2010	No	Yes	Yes	Interactive	No	No
Poland *et al.* [12]	2009	No	Yes	Yes	Interactive	No	No
Krzyska *et al.* [47]	2006	No	No	Yes	Interactive	Yes	No

The approach that OpenSHS uses to generate datasets can be thought of as a middle ground between the model-based and interactive approaches. The replication mechanism that OpenSHS adapts, allows for a quick dataset generation, similar to the model-based approaches. Moreover, the replications have richer details as the activities are captured in real-time, similar to the interactive approaches. OpenSHS's fast-forwarding mechanism streamlines the performance of the simulation and allows the participant to skip in time while conducting a simulation. Overall, the advantages of OpenSHS can be summarised as follows:

1. **Accessibility**: The underlying technologies used to develop OpenSHS allowed it to work on multiple platforms, thus ensuring a better accessibility for the researchers and the participants alike.
2. **Flexibility:** OpenSHS gives the researchers the flexibility to simulate different scenarios according to their needs, by adding and/or removing sensors and smart devices. OpenSHS can be easily modified and customised in terms of positioning and changing the behaviour of the smart devices in the virtual smart home to meet the needs of a research project.

3. **Interactivity:** Capturing the interactions between the participant and the smart home in OpenSHS was done in a real-time fashion which facilitates the generation of richer datasets.
4. **Scalability:** Our simulation tool is scalable and easily extensible to add new types of smart devices and sensors. OpenSHS has a library of smart devices that we will keep developing and updating as new types of smart devices become available.
5. **Reproducibility:** By being an open-source project, OpenSHS does have the advantage of facilitating reproducibility and allowing research teams to produce datasets to validate other research activities.

6. Future Work

Although, OpenSHS currently supports the simulation of one smart home inhabitant, however multiple inhabitants simulations is partially supported. The current implementation of this feature does not allow real-time simulation of multiple inhabitants. Instead, The first inhabitant records his/her activities and then the second inhabitant can start another simulation. The second inhabitant will be able to see the first inhabitant's actions played back in the virtual environment. For future work, we plan to include full multiple inhabitants support in real-time. Moreover, the smart devices library, has few specialised sensors that will be updated in the future to include new types of sensors and devices. Another feature that could improve the design phase of the smart home, is the addition of a floor plan editor. Taking into consideration that OpenSHS is an open-source project, released under a free and permissive license, the project could envisage quick and rapid development that would facilitate the support of the aforementioned features.

The more realistic the simulation is, the less the need for building actual smart homes to carry out research. Following the growing advancements in computer graphics, Virtual Reality (VR) is becoming more accessible and affordable. BlenderVR [57] is an open-source framework that extends Blender and allows it to produce immersive and realistic simulations. Since OpenSHS is based on Blender, one of our future goals is to investigate the incorporation of BlenderVR into our tool to provide more true to life experiences for the smart home simulation and visualisation. In terms of accessibility, we aim to make OpenSHS as accessible as possible. Nowadays, the web technologies and web browsers can be a good platform to facilitate the wider distribution of OpenSHS. Technologies such as WebGL [58] can be used to run OpenSHS in different web browsers and Blender can export to these technologies.

7. Conclusion

Many smart home research projects require the existence of representative datasets for their respective applications and research interests and to evaluate and validate their results. Many simulation tools available in the literature focus on context-awareness and few tools have set dataset generation as their aim. Moreover, there is a lack of open-source simulation tools in the public domain. We developed OpenSHS, an open-source, 3D and cross-platform simulation tool for smart home dataset generation. OpenSHS has many features that allow the researchers to easily design different scenarios and produce highly intricate and representative datasets. Our tool offers a library of smart sensors and devices that can be expanded to include future emerging technologies.

OpenSHS allows the researchers to rapidly generate seeds of events. We have presented a replication algorithm that can extend the simulated events to generate multiple unique large datasets. Moreover, conducting a simulation with a participant can be done in a reasonable time and we provided tools that streamlines the process such as fast-forwarding.

Our tool divides the dataset generation process into three distinct phases, design, simulation and aggregation. In the design phase, the researcher creates the initial virtual environment by building the home, importing smart devices and creating contexts. In the simulation phase, the participant uses the virtual home to generate context-specific events. In the final stage, the researcher applies the replication algorithm to generate the aggregated dataset.

We conducted a usability study using the System Usability Scale (SUS) to assess how usable OpenSHS is. The results of this study were promising, yet they left room for more improvements.

One of the identified issues in smart home simulations tools, is having the support for multiple inhabitants. This is a challenging task both for the simulation tool and for the participants. Currently, OpenSHS offers partial support for multiple inhabitants. To increase the realism of the simulations, we plan to integrate VR technologies into OpenSHS in the future. The accessibility for both the researchers and the participants is an important feature, hence, we plan to port the implementation of OpenSHS to run in a web browser.

Acknowledgments: Nasser Alshammari and Talal Alshammari are carrying out their PhD studies in Staffordshire University. The Ministry of Education in Saudi Arabia funds and supports their research projects.

Author Contributions: Nasser Alshammari conceived and developed OpenSHS and its replication algorithm as well as performed the experiments. Talal Alshammari completed a review of existing simulation tools and approaches and contributed to the usability study. Mohamed Sedky, Justin Champion and Carolin Bauer provided guidance and direction for the implementation, development and evaluation of the research. All authors significantly contributed to the writing and review of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Buchmayr, M.; Kurschl, W.; Küng, J. A simulator for generating and visualizing sensor data for ambient intelligence environments. *Procedia Computer Science* **2011**, *5*, 90–97.
2. Rodner, T.; Litz, L. Data-driven generation of rule-based behavior models for an ambient assisted living system. IEEE Third International Conference on Consumer Electronics. IEEE, 2013, pp. 35–38.
3. Youngblood, G.M.; Cook, D.J.; Holder, L.B. Seamlessly engineering a smart environment. SMC, 2005, pp. 548–553.
4. Helal, S.; Lee, J.W.; Hossain, S.; Kim, E.; Hagras, H.; Cook, D. Persim-Simulator for human activities in pervasive spaces. 7th International Conference on Intelligent Environments (IE). IEEE, 2011, pp. 192–199.
5. Tapia, E.M.; Intille, S.S.; Larson, K. Activity recognition in the home using simple and ubiquitous sensors. International Conference on Pervasive Computing. Springer, 2004, pp. 158–175.
6. Synnott, J.; Nugent, C.; Jeffers, P. Simulation of Smart Home Activity Datasets. *Sensors* **2015**, *15*, 14162.
7. Mendez-Vazquez, A.; Helal, A.; Cook, D. Simulating events to generate synthetic data for pervasive spaces. Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research. Citeseer, 2009.
8. Lei, Z.; Yue, S.; Yu, C.; Yuanchun, S. SHSim: An OSGI-based smart home simulator. 3rd IEEE International Conference on Ubi-media Computing (U-Media). IEEE, 2010, pp. 87–90.
9. Armac, I.; Retkowitz, D. Simulation of smart environments. IEEE International Conference on Pervasive Services. IEEE, 2007, pp. 257–266.
10. Helal, S.; Kim, E.; Hossain, S. Scalable approaches to activity recognition research. Proceedings of the 8th International Conference Pervasive Workshop, 2010, pp. 450–453.
11. Fu, Q.; Li, P.; Chen, C.; Qi, L.; Lu, Y.; Yu, C. A configurable context-aware simulator for smart home systems. 6th International Conference on Pervasive Computing and Applications (ICPCA). IEEE, 2011, pp. 39–44.
12. Poland, M.P.; Nugent, C.D.; Wang, H.; Chen, L. Development of a smart home simulator for use as a heuristic tool for management of sensor distribution. *Technology and Health Care* **2009**, *17*, 171–182.
13. Cook, D.J.; Youngblood, G.M.; Heierman III, E.O.; Gopalratnam, K.; Rao, S.; Litvin, A.; Khawaja, F. MavHome: An Agent-Based Smart Home. PerCom, 2003, Vol. 3, pp. 521–524.
14. Alemdar, H.; Ertan, H.; Incel, O.D.; Ersoy, C. ARAS human activity datasets in multiple homes with multiple residents. 2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops. IEEE, 2013, pp. 232–235.
15. Munguia Tapia, E. Activity recognition in the home setting using simple and ubiquitous sensors. PhD thesis, Massachusetts Institute of Technology, 2003.

16. Healy, G.N.; Clark, B.K.; Winkler, E.A.; Gardiner, P.A.; Brown, W.J.; Matthews, C.E. Measurement of adults' sedentary time in population-based studies. *American journal of preventive medicine* **2011**, *41*, 216–227.
17. Cook, D.J.; Crandall, A.S.; Thomas, B.L.; Krishnan, N.C. CASAS: A smart home in a box. *Computer* **2013**, *46*.
18. WSU CASAS Datasets. http://ailab.wsu.edu/casas/datasets/. (accessed on 12 January 2017).
19. Skubic, M.; Alexander, G.; Popescu, M.; Rantz, M.; Keller, J. A smart home application to eldercare: Current status and lessons learned. *Technology and Health Care* **2009**, *17*, 183–201.
20. Nugent, C.; Mulvenna, M.; Hong, X.; Devlin, S. Experiences in the development of a smart lab. *International Journal of Biomedical Engineering and Technology* **2009**, *2*, 319–331.
21. McDonald, H.; Nugent, C.; Hallberg, J.; Finlay, D.; Moore, G.; Synnes, K. The homeML suite: shareable datasets for smart home environments. *Health and Technology* **2013**, *3*, 177–193.
22. Yamazaki, T. The ubiquitous home. *International Journal of Smart Home* **2007**, *1*, 17–22.
23. Intille, S.S.; Larson, K.; Tapia, E.M.; Beaudin, J.S.; Kaushik, P.; Nawyn, J.; Rockinson, R. Using a live-in laboratory for ubiquitous computing research. International Conference on Pervasive Computing. Springer, 2006, pp. 349–365.
24. PlaceLab datasets. http://web.mit.edu/cron/group/house_n/data/PlaceLab/PlaceLab.htm. (accessed on 05 February 2017).
25. De Ruyter, B.; Aarts, E.; Markopoulos, P.; Ijsselsteijn, W. Ambient intelligence research in homelab: Engineering the user experience. In *Ambient Intelligence*; Springer, 2005; pp. 49–61.
26. Helal, S.; Mann, W.; El-Zabadani, H.; King, J.; Kaddoura, Y.; Jansen, E. The gator tech smart house: A programmable pervasive space. *Computer* **2005**, *38*, 50–60.
27. Lee, J.W.; Cho, S.; Liu, S.; Cho, K.; Helal, S. Persim 3D : Context-Driven Simulation and Modeling of Human Activities in Smart Spaces. *IEEE Transactions on Automation Science and Engineering* **2015**, *12*, 1243–1256.
28. Bouchard, K.; Ajroud, A.; Bouchard, B.; Bouzouane, A., SIMACT: A 3D Open Source Smart Home Simulator for Activity Recognition. In *Advances in Computer Science and Information Technology: AST/UCMA/ISA/ACN 2010 Conferences, Miyazaki, Japan, June 23-25, 2010. Joint Proceedings*; Kim, T.h.; Adeli, H., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2010; pp. 524–533.
29. Java Monkey Engine (JME). http://www.jmonkeyengine.com. (accessed on 26 November 2016).
30. Jouve, W.; Bruneau, J.; Consel, C. DiaSim: A parameterized simulator for pervasive computing applications **2009**. pp. 1–3.
31. Park, J.; Moon, M.; Hwang, S.; Yeom, K. CASS: A context-aware simulation system for smart home. 5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA). IEEE, 2007, pp. 461–467.
32. Kim, I.; Park, H.; Noh, B.; Lee, Y.; Lee, S.; Lee, H. Design and implementation of context-awareness simulation toolkit for context learning. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC). IEEE, 2006, Vol. 2, pp. 96–103.
33. Park, B.; Min, H.; Bang, G.; Ko, I. The User Activity Reasoning Model in a Virtual Living Space Simulator. *International Journal of Software Engineering and Its Applications* **2015**, *9*, 53–62.
34. Unity3D. https://unity3d.com/. (accessed on 22 November 2016).
35. Synnott, J.; Chen, L.; Nugent, C.; Moore, G. The creation of simulated activity datasets using a graphical intelligent environment simulation tool **2014**. pp. 4143–4146.
36. Ariani, A.; Member, S.; Redmond, S.J.; IEEE, M.; Chang, D.; Lovell, N.H.; IEEE, F. Simulation of a Smart Home Environment **2013**. pp. 27–32.
37. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* **1968**, *4*, 100–107.
38. Nishikawa, H.; Yamamoto, S.; Tamai, M.; Nishigaki, K.; Kitani, T.; Shibata, N.; Yasumoto, K.; Ito, M. UbiREAL: realistic smartspace simulator for systematic testing. International Conference on Ubiquitous Computing. Springer, 2006, pp. 459–476.
39. Lertlakkhanakul, J.; Choi, J.W.; Kim, M.Y. Building data model and simulation platform for spatial interaction management in smart home. *Automation in Construction* **2008**, *17*, 948–957.

40. FlexSim Software Products, Inc.. FlexSim Simulation Software. https://www.flexsim.com/. (accessed on 19 December 2016).
41. LLC, S. Simio Simulation Software. http://www.simio.com/. (accessed on 19 December 2016).
42. Automation, R. Arena Simulation Software. http://www.arenasimulation.com/. (accessed on 20 December 2016).
43. Stahl, C.; Schwartz, T. Modeling and simulating assistive environments in 3-D with the YAMAMOTO toolkit. International Conference on Indoor Positioning and Indoor Navigation (IPIN). IEEE, 2010, pp. 1–6.
44. O'Neill, E.; Klepal, M.; Lewis, D.; O'Donnell, T.; O'Sullivan, D.; Pesch, D. A testbed for evaluating human interaction with ubiquitous computing environments. First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities. IEEE, 2005, pp. 60–69.
45. Barton, J.J.; Vijayaraghavan, V. Ubiwise, a ubiquitous wireless infrastructure simulation environment. *HP Labs* **2002**.
46. McGlinn, K.; O'Neill, E.; Gibney, A.; O'Sullivan, D.; Lewis, D. SimCon: A Tool to Support Rapid Evaluation of Smart Building Application Design using Context Simulation and Virtual Reality. *J. UCS* **2010**, *16*, 1992–2018.
47. Krzyska, C. Smart House Simulation Tool. PhD thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2006.
48. Lundström, J.; Synnott, J.; Järpe, E.; Nugent, C.D. Smart home simulation using avatar control and probabilistic sampling. Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on. IEEE, 2015, pp. 336–341.
49. Kormányos, B.; Pataki, B. Multilevel simulation of daily activities: Why and how? 2013 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA). IEEE, 2013, pp. 1–6.
50. Jahromi, Z.F.; Rajabzadeh, A.; Manashty, A.R. A Multi-Purpose Scenario-based Simulator for Smart House Environments **2011**. *9*, 13–18.
51. Van Nguyen, T.; Kim, J.G.; Choi, D. ISS: the interactive smart home simulator. 11th International Conference on Advanced Communication Technology (ICACT). IEEE, 2009, Vol. 3, pp. 1828–1833.
52. Alshammari, N.; Alshammari, T.; Sedky, M.; Champion, J.; Bauer, C. openshs/openshs: First alpha release. https://doi.org/10.5281/zenodo.274214, 2017.
53. GNU General Public License, version 2. https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html, 1991. (accessed on 11 January 2017).
54. Blender. https://www.blender.org. (accessed on 06 November 2016).
55. OpenGL. https://www.opengl.org. (accessed on 05 November 2016).
56. Brooke, J.; others. SUS-A quick and dirty usability scale. *Usability evaluation in industry* **1996**, *189*, 4–7.
57. Katz, B.F.; Felinto, D.Q.; Touraine, D.; Poirier-Quinot, D.; Bourdot, P. BlenderVR: Open-source framework for interactive and immersive VR. Virtual Reality (VR), 2015 IEEE. IEEE, 2015, pp. 203–204.
58. WebGL. https://www.khronos.org/webgl/. (accessed on 05 November 2016).