



Jae Woong Lee^{1,*} and Abdelsalam Helal²

- ¹ Department of Computer Science, State University of New York at Oswego, Oswego, NY 13126, USA
- ² Department of Computer & Information Science & Engineering, University of Florida,
- Gainesville, FL 32611, USA; helal@cise.ufl.edu
- * Correspondence: jaewoong.lee@oswego.edu; Tel.: +1-315-312-2786
- + This article is the extension of the following conference paper: "Modeling and Reasoning of Contexts in Smart Spaces Based on Stochastic Analysis of Sensory Data", which was published in proceedings of 16th IEEE workshop on Context and Activity Modeling and Recognition (CoMoRea '20) collocated with the 19th Annual IEEE International Conference on Pervasive Computing and Communications in Austin, TX, USA, 23–27 March 2020. Please feel free to contact the corresponding author at jaewoong.lee@oswego.edu if you have any questions.

Abstract: In the last decade, smart spaces and automatic systems have gained significant popularity and importance. Moreover, as the COVID-19 pandemic continues, the world is seeking remote intervention applications with autonomous and intelligent capabilities. Context-aware computing (CAC) is a key paradigm that can satisfy this need. A CAC-enabled system recognizes humans' status and situation and provides proper services without requiring manual participation or extra control by humans. However, CAC is insufficient to achieve full automaticity since it needs manual modeling and configuration of context. To achieve full automation, a method is needed to automate the modeling and reasoning of contexts in smart spaces. In this paper, we propose a method that consists of two phases: the first is to instantiate and generate a context model based on data that were previously observed in the smart space, and the second is to discern a present context and predict the next context based on dynamic changes (e.g., user behavior and interaction with the smart space). In our previous work, we defined "context" as a meaningful and descriptive state of a smart space, in which relevant activities and movements of human residents are consecutively performed. The methods proposed in this paper, which is based on stochastic analysis, utilize the same definition, and enable us to infer context from sensor datasets collected from a smart space. By utilizing three statistical techniques, including a conditional probability table (CPT), K-means clustering, and principal component analysis (PCA), we are able to automatically infer the sequence of context transitions that matches the space-state changes (the dynamic changes) in the smart space. Once the contexts are obtained, they are used as references when the present context needs to discover the next context. This will provide the piece missing in traditional CAC, which will enable the creation of fully automated smart-space applications. To this end, we developed a method to reason the current state space by applying Euclidean distance and cosine similarity. In this paper, we first reconsolidate our context models, and then we introduce the proposed modeling and reasoning methods. Through experimental validation in a real-world smart space, we show how consistently the approach can correctly reason contexts.

Keywords: context awareness computing/reasoning; context model; context graph; smart spaces; sensor data; stochastic analysis; principal component analysis

1. Introduction

For the last decade, we have been experiencing speedy and dramatic changes in our lives due to smart technology. Every year, new devices and systems have been released with the prefix "smart", for instance, smart door, smart TV, and smart kitchen. One of the aims of



Citation: Lee, J.W.; Helal, A. Modeling and Reasoning of Contexts in Smart Spaces Based on Stochastic Analysis of Sensor Data. *Appl. Sci.* 2022, *12*, 2452. https://doi.org/ 10.3390/app12052452

Academic Editors: Rung-Ching Chen, Hui Yu, Hendry Hendry and Qiangfu Zhao

Received: 7 January 2022 Accepted: 20 February 2022 Published: 26 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/). this smartening is to enable a transformation from the manual and static world of humans to an automatic and dynamic world. In a smart space, such smart things are integrated, and hence needed, and convenient services are automatically provided. These days, it is no longer a dream for a house to adjust the indoor atmosphere, such as temperature, humidity, and brightness, and to help the residents to cook or clean [1]. Smart space technology has been utilized in other areas, such as educational institutes [2,3], medical/healthcare centers [4], and industrial facilities [5]. One of the biggest benefits for smart spaces and their automation of services is to enable untouchable applications. Since smart spaces are operated by sensing devices and intelligent systems, human assistance, in which services are physically contacted, is no longer needed. This is surely appropriate and useful during the COVID-19 pandemic, which requires people to maintain social distancing and avoid direct contact. As this worldwide pandemic continues without signs of ending, the need for such smart spaces is increasing, which will accelerate the development of advanced intelligent systems for smart spaces [6–10].

The performance of smart spaces depends on intelligent systems, which are internally installed. The intelligent systems discern users' current state and determine services that they might need or desire. The initial step in such systems is to accurately understand the context(s) of the space and its users. Thus, a context-aware design is one of the keys to developing intelligent systems for smart spaces. There has been research defining contexts in different ways, but formal definitions were proposed by Dey and Yau. In [11], Dey et al. defined "context" as "any information that can be used to characterize the situation of an entity; an entity is a person, place, or object that is considered relevant to the interaction between a user and applications themselves". Yau et al. [12] had a similar definition, that is, "any detectable attribute of a device, its interaction with external devices, and/or its surrounding environment". Even though the structures for context are organized differently, they agreed that particular entities should be digitalized in smart-space systems, so that context can be systematically and algorithmically captured. Context, which was defined in our prior work [13], is also described by the status of various entities located in a space. This context design enables the construction of an abstract space state and facilitates a more systematic process for recognizing contexts.

The status of entities used in defining a context can be obtained and digitalized by using the status of sensors attached to (or in range of) the corresponding entities, which are located in a smart space [14]. As Internet-of-Things (IoT) technology evolves, the traditional paradigm of entity–sensor has also evolved and changed, in which entities are allowed to access alternative devices to obtain status data using handy APIs provided by the device manufactures [15]. Such entities are more obtainable and supportive of modeling contexts. In smart spaces, once context models are fed the status data of entities, the present context is reasoned by analyzing the obtained status data. This is the basic flow of context-awareness computing (CAC) and works as the core of smart spaces. One advantage of a CAC-enabled system is that no human needs to be involved or interact with others once CAC is installed in a system.

However, the automaticity of the CAC approach is limited by a structural drawback, as the current CAC mechanism requires the manual configuration of context models. To install a CAC in a smart-space system, users must define possible contexts and configure them in advance. There is a burden on human efforts and a risk of errors or mistakes when manually designing contexts. There is a further issue regarding the unity of contexts when manually defined. Since individuals have their own perception of the contexts that occur in a space, context models may vary from user to user. Similarly, they recognize contexts differently. As a result, a CAC-enabled system with manually modeled contexts may recognize contexts differently, and thus cannot serve humans consistently. Thus, it is important to create capabilities to algorithmically (automatically) build realistic context models, which can perform uniformly. Such capabilities will enable practically powerful automations, which do not require non-scalable and limiting manual steps.

This paper proposes methods for modeling and reasoning contexts that can achieve full automation. In our definition, contexts are modeled by the sensors, which are attached to the entities, including objects, humans, and other built-environment artifacts. Based on the context models, the recognition process proceeds to reason a context by analyzing the given sensor data. Since both modeling and reasoning processes need to manage sensory data, it is critical to build a context in an efficiently organizable and easily manipulable structure. In our approach, we developed a method in which a well-organized context model is built by analyzing sensory data that describe the status of contextual entities. To model this context structure, we considered two key ideas. The first idea comes from the observation that most sensors triggered in the same context are somehow interrelated. The second idea comes from the observation that there are sensors that are significant and conducive to specific contexts. This observation raises the possibility of logically dividing a whole sensor dataset into groups, each of which contains data from inter-related sensors. Since each of the divided groups is likely an own-context, by virtue of the two aforementioned observations, it is important to logically divide a given dataset accurately, which, in turn, requires that we properly find interrelated sensors and their data. We address dividing and searching methods for building context models in more detail in the rest of this paper.

The proposed approach proceeds in two phases: First, context models are built, and then a present context is determined from the context models. In the first phase, the context models are inferred from a collection of obtained sensor datasets under supervised learning. For the inference process, we utilize three machine learning methods along with statistical analysis: *K*-means clustering, a conditional probability table (CPT), and principal components analysis (PCA). The second phase begins after the context models are successfully established. In this phase, we compare the current state space to all the context models and choose the context that is most similar to the current state space. The similarity is calculated by two distance methods: Euclidean distances and cosine similarity. To systematically and algorithmically facilitate the two phases of our approach, we introduce and utilize a high-level contextual structure, called a context graph, to be utilized in developing and using (reasoning) the context models.

The modeling and reasoning process may lead to privacy and security issues for smart-space systems and their users. In smart spaces, the most used and very efficient sensing devices are cameras, which can capture the data of any user and/or objects seen in the camera range. The data may contain more information than that which must be obtained; there may be someone who does not wish to be sensed or private objects that must not be digitalized. At-home settings, which are more private and secured, require further caution when cameras are used [16]. For these reasons, and out of an abundance of caution, only data obtained by means other than cameras are used in this paper.

We organize this paper as follows. In Section 2, we describe existing work related to context-awareness computing, context modeling, and reasoning methods. Section 3 introduces the design of the proposed context model and context graph, which provide a fundamental formalism to our approach. The principles of the proposed approach for modeling and reasoning contexts from a real-world situation are presented in Section 4. Experimental validation and case studies follow in Section 5. Discussions and conclusions for this research are presented in Section 6.

2. Related Work

As shown in [17,18], research on modeling and reasoning context has a long history, and various approaches have been proposed in different areas, such as human–computer interaction (HCI) [19,20], smart home simulation [13,21,22], context-awareness computing [23,24], and activity recognition learning [25–30]. We categorize prior work into three categories: The first category focuses on syntactic and systematic approaches based on humans' natural perspectives in understanding contextual situations. Approaches in the second category have developed applicable and algorithmic methods to define contexts.

The third category contains domain-specific approaches, each of which proposes context models in a particular area, specific to its own reasoning procedures.

2.1. Syntactic and Systematic Approaches

The purpose of these approaches is to organize context models based on a syntactic analysis of contexts using humans' natural perspectives. Ontology-based analysis has traditionally been applied in modeling and reasoning contexts [31–33]. In this research, context is designed in an ontology-oriented model, which facilitates a reasoning process by comparing predefined context models and the current situation. Based on an RDF triple database consisting of subject, predicate, and object, the context model is built with additional contextual information, such as time and location. All the descriptive resources are organized in an ontology-oriented model and abridged in an XML syntax. By applying well-organized syntactic structures, the necessary information for describing contexts is instantiated. The syntactic context models systematically facilitate reasoning process and are easily translated into humans' natural language to achieve more human-centric, smartspace systems. However, an ontology-oriented modeling approach still needs the efforts of human users to configure each context model, which remains burdensome and ambiguous, especially when defining context models. Our idea does not approach syntactic models for contexts. Instead, it aims to ensure a faster process, to be aware of the current context, and to provide a proper and convenient service without any syntactic analysis of contexts. In our prior study [13], we proposed a context model which works in the simulation of smart spaces. In this paper, we focus on the automation of defining contexts by analyzing sensor data collected from smart spaces. Our proposal aims to build context models from the sequence of space states observed through human senses or electronic sensors attached to objects or humans.

2.2. Applicable and Algorithmic Approaches

Even though the syntactic analysis of contexts in the first category is an advantage in the described contexts, it does provide entities which are needed to reason a context in a given situation. In the second category, modeling contexts is primarily dependent on the reasoning context. In other words, entities in a context model are determined through the reasoning context process. Thus, it is critical to appropriately build rules to catch contextual space states and determine contexts. These approaches are mostly found in the research for context-awareness computing or activity recognition learning. The context-driven simulation approach [13] is an example. Contexts defined in this approach were defined by related context entities, such as sensors and their statuses. Once context models are manually defined, the causality in between contexts also needs to be configured so that they can describe an entire daily living scenario. Another example, the context-aware simulation system for smart home (CASS) [22] adapts the mechanism of a rule-based system, in which the system detects the conflicts of certain rules to control a preconfigured character to move it. The rules need to be defined to describe certain conditions of entities, causing burdens in modeling contexts.

To reduce the burden and increase automaticity, research into activity recognition has derived high-level information, which is meaningful for activity, from low-level information, which generally consists of sensory data. The research assumed that the collected sensor datasets triggered by human activities could be divided into multiple clusters, each of which contains highly relevant sensors. For instance, in cluster-based classification for activity recognition systems (CBARS) [26], supervised learning models were first built by clustering training datasets, and then unsupervised learning was applied with new testing data to recognize an activity. The challenge was that a supervised learning model was needed for the recognition process. This was addressed in activity recognition using active learning in the overlapped activities (AALO) [27]. It proposed an active and dynamic recognition system, which enables an accurate classification of specific activities according to locations and times, without training data. With cluster-based classifier ensemble (CBCE) [28], an approach was proposed that combines multiple classifiers including Naïve Bayesian (NB) models, hidden Markov models (HMMs), and conditional random fields (CRFs). Even though this ensemble of classifiers enabled an unsupervised method to recognize an "activity", it is hard to define abstract information for a "context", which represents other state spaces in a given cluster, due to lack of contextual entities.

As context-aware computing is increasingly integrated into end-user service applications, the development of more applicable and practical methods has been the subject of relevant research. In mobile cloud services [34] and recommender systems [35,36], for instance, historic computational methods such as cosine similarity were applied for the context recognition process. However, it is not their main purpose to find a present context. The primary goal of the approaches is to suggest proper service(s) for the present context found by the recognition process and, thus, an extra process for finding proper services should be developed. As a result, the computational complexity of their proposed methods will greatly increase.

2.3. Domain-Specific Approaches

The approaches in the second category proposed context models for smart homes. For generic context models, studies in the third category have proposed methods for modeling activities and contexts from sensory data in specific domains. They observed that there were different contextual conditions per domain, and thus developed domain-specific methods for modeling and reasoning. Since some domains have environments that are difficult to access, their approaches propose performing a simulation in a virtualized space. The authors of [21] assumed that elements such as fact and belief are an integral part of a model for human astronauts' activities in outer space. The activity recognition process was simulated based on real data collected from space experiments in a NASA project. The authors of [37] presented a simulation tool used to model common behaviors of human subjects in outdoor environments. The goal of the tool was to study how best to route ad hoc traffic based on human mobility. The authors of [38] focused more on the activities of nurses caring for patients in a hospital setting, with the goal of maximizing scheduling efficiency and minimizing downtime for patients. In such simulations, computation was intensive, and classification-based reduction techniques were used to improve simulation scalability. Although these approaches are useful for dangerous and dynamic situations, they may impose unnecessary complexity and require excessive computation when simulating simple activities of daily life.

3. Design of Context Model

In our definition, context represents a meaningful state space, which is described by the status of entities such as objects and residents in a space. To decide that a state space is a context, changes in the status of the entities should be persistently tracked and evaluated. Before we introduce the main process for recognizing contexts, we define the necessary models, which are state space, context, and context graph. The principles of the reasoning algorithm will be explained in the following section.

3.1. Context Model

3.1.1. State Space

In order to model a context, we must first define a state space, since a context is considered a representative state space. In a smart space, sensors attached to objects or worn by human residents provide the status of the space—state space, for short. In our state space model, all the sensors' statuses are collected in an ordered set, as shown in Equation (1). Thus, a state space, denoted by *S*, in which $\boldsymbol{\omega}$ sensors are attached, is defined as follows:

$$S = \{s_1, s_2, \dots, s_{i_\ell}, \dots, s_{\omega}\},\tag{1}$$

where s_i contains a status (usually indicated by a single value in any type, such as numbers) of sensor s_i . During a daily living scenario in the real world, the status of sensors is affected

as human residents perform activities or objects move in a space. In other words, state space *S* continuously changes, and a multitude of different state spaces will be defined, as illustrated in Figure 1. The universal set of state spaces is a collection of all *S* and denoted by Equation (2):

 $\hat{S} = \{S_1, S_2, \dots, S_c, \dots, S_{\zeta}\}.$



Figure 1. State spaces
$$\hat{S}$$
 describes all states of a given daily living scenario in order. Note that each state space indicates the status of every individual sensor. Contexts \hat{C} are defined as meaningful and representative states out of \hat{S} .

In Equation (2), the total number of state spaces obtained is ζ and the current state space is noted by S_c . In a given daily living scenario in Figure 1, S starts from S_1 , and then changes to any other state in \hat{S} as time passes. Note that \hat{S} is recursively generated, and S may return to S_1 when the daily living scenario ends.

3.1.2. Context

Context is a meaningful state space, which is representative enough to describe what is happening in the space [13]. In a context model, it is important to determine which sensors are related, and which sensor status is required. Suppose that we are reasoning a context, "making breakfast", in a kitchen in which an electric stove, microwave, refrigerator, spoons, knives, ceiling light, and windows are placed. The sensors attached to the objects will change their status when the corresponding objects are used. Recall that any change in sensor status triggers the state space and generates a different one. In the kitchen, many state spaces may be generated; however, some of them will not be related to the context. For instance, the ceiling light and windows are not needed to make breakfast; thus, state spaces triggered by sensors attached to them are not meaningful for that context. It can be recognized that the context only begins, or is in progress, if state spaces are triggered by the sensors attached to cooking tools, such as an electric stove, microwave, refrigerator, spoons, and knives. To reason a context, however, a sufficient number of relevant sensors

(2)

are triggered. In the example, it is hard to reason the context if one touch sensor attached to a refrigerator is triggered since the refrigerator can be touched only for drinking water. The state space generated by this sensor event is not representative; hence, it will be ignored. As shown in Figure 1, brown-colored state spaces with enough information are considered contexts, while others remain normal state spaces. Thus, it is critical to locate all sensors that contribute to modeling and reasoning a context. Once a context is chosen, the state spaces after it belong to the context since the context lasts until a new context begins.

Context has a fundamentally identical structure to state space *S* since it is a selected *S*, which is a collection of sensor statuses, as follows:

$$C = \{s_1, s_2, \dots, s_i, \dots, s_{\omega}\}$$
(3)

Note that sensors that are not relevant to any context will be ignored. These sensors have a null state value, so they will not be considered in the reasoning process.

As a multitude of contexts can be selected from a collection of state spaces \hat{S} , the collection of the contexts is noted by Equation (4):

$$\hat{\mathbf{C}} = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_{\rho}, \dots, \mathbf{C}_{\eta}\}$$
(4)

In Equation (4), the total number of contexts is η , and the present context is denoted by C_{ρ} . As contexts are selected from the collection of state spaces \hat{S} , the collection of contexts \hat{C} is a subset of \hat{S} ($\hat{C} \subseteq \hat{S}$), which infers that $\eta \leq \zeta$. The status of these sensors in a given context is considered as a condition for being the context since it distinguishes itself from non-context state spaces or other contexts. In other words, when the sensors in a given state space have the same status for the corresponding sensors in a target context, it is said that the state space represents a context.

3.2. Context Graph

The set of contexts \hat{C} is used to find a current context during the reasoning process. To improve reasoning methods, \hat{C} needs to be structured into a context graph. We observed that particular pairs of contexts were occurred by a certain causality or time order. Consider two contexts: "sleeping at night" and "having breakfast". The two contexts should happen in a certain order, that is, "sleeping at night", followed by "having breakfast". However, the opposite case, which is, "having breakfast", followed by "sleeping at night", cannot occur. This infers that contexts may be implicitly described by the hidden property of time—night and morning in this example. The time property can define an order between contexts. However, there is an example in which it does not affect pairs of other contexts, for example, the context "going to bathroom". It is meaningless to consider the time property or find a specific order because humans may go to the bathroom at any time while they are sleeping or making breakfast. The two cases support an idea, which is that the time property is not used to model an individual context but defines the relationships between contexts. The context graph reflects the relations by using the features of a directed graph, as shown in Figure 2.

Figure 2 indicates all transitions between contexts and illustrates how contexts are driven. A causal relation can be either determinative or non-determinative. The contexts under a determinative relation have a single possible transition from a predecessor context. In Figure 2, for instance, C_1 and C_2 have a determinative relation since C_1 is followed by C_2 . In this case, C_2 is only one successor context, which comes next. On the other hand, transitions from C_2 have three options, that is, C_2 followed by C_3 , C_2 followed by C_5 , and C_2 followed by C_6 . It is not determinative since the predecessor C_2 can transition to any possible successor: C_3 , C_5 , and C_6 . These successors are considered as the next possible contexts, which are followed by C_2 . We generalize this and define the set of the next possible contexts C_N , which follow a given context C_i in Equation (5):

$$C_N = \{C_1, C_2, \dots, C_{\tau}, \dots, C_{\chi}\},$$
 (5)

where $C_N \subseteq \hat{C}$. Note that every C_j has its own C_N . For example, the C_N of C_1 is $\{C_2\}$, and the C_N of C_2 is $\{C_3, C_4, C_5\}$ in Figure 2. The C_N of contexts under determinative relation is a singleton set with one possible next context.



Figure 2. Context graph C_G is created from \hat{S} . Note that the dashed edges are transactions, which are not illustrated in Figure 1; however, they might occur as the state space changes. For simplicity, we assume that it returns to C_1 .

In daily living scenarios, context graph C_G allows for a context sequence to be repeated as a daily routine. Recall the two contexts, "making breakfast" and "sleeping at night". The context "making breakfast" literally happens every morning, while the context "sleeping at night" generally starts every night. The contexts can be repeated daily, as they are cycled. However, the allowance of cycled contexts in C_G causes an issue, that is, it is difficult to designate a root context, which may be naturally and originally defined. Even though C_1 looks like a root in Figure 2, it might be a next context, which follows a certain context in the real world.

4. Principles of Context Reasoning

In context awareness computing, reasoning a context is functionally the same as finding the next context in the present context. Thus, our approach aims to discover the next context with respect to the present context, C_{ρ} , and the current state space, S_c . The main procedure for reasoning is that, in a given C_{ρ} , S_c happens, and then it becomes a next context, C_v , which is found from the next possible context, C_N , if S_c is sufficiently similar to C_v . Our approach follows this procedure, and all the methods in our approach are designed to find the next context.

4.1. Overall Approach

The reasoning process is fundamentally event-driven since it proceeds whenever a sensor event occurs. The sensor event is caused by resident activities or the movement of objects in the space, and then changes a state space. Assuming that the current context is C_{ρ} , if the reasoning procedure qualifies the changed state space as the next context C_{v} , the transition into C_{v} is indeed performed, as shown in Algorithm 1. Otherwise, if the algorithm fails to find such C_{v} , the current context remains, and the algorithm waits for the next change in the state space. Algorithm 1 runs in two phases: (1) *MODELING* and (2) *REASONING*.

Algorithm 1. Context Reasoning Main **INPUT:** a training set of state spaces \hat{S} **OUTPUT:** a next context C_{ρ} Context Graph $C_G \leftarrow$ Build a context graph with (\hat{S}) by Algorithm 2 1. 2. State Space $S_c \leftarrow$ Initialize the current state space 3. Context $C_{\rho} \leftarrow$ Initialize the present context with (S_c, C_G) 4. Context $C_{\nu} \leftarrow NULL$ //selected next context 5. while *S*_c changes by any events $C_{\nu} \leftarrow$ Reason context with (S_c, C_{ρ}, C_G) by Algorithm 3 6. 7. if $C_{\nu}! = NULL$ $C_{
ho} \leftarrow C_{
ho}$ //transition to $C_{
ho}$ 8. 9. //output C_{ρ} as the new present context

10. end if

11. end while

4.1.1. MODELING

The initial step for reasoning contexts is to establish a context graph C_G with a given set of state spaces \hat{S} which is formed from training datasets. C_G is built by applying the statistical analysis method of sensor datasets [4]. The statistical analysis method runs in three main phases. First, the dataset is analyzed by a conditional probabilistic table (CPT) based on Bayesian networks and estimates the number of contexts, that is, *K*. Then, the dataset is clustered into *K* contexts by applying *K*-means clustering. Thirdly, conditions for entering a context are discovered by utilizing principal components analysis (PCA). PCA is a way of finding important components and reducing the dimensions of a given dataset by projecting it into the lower dimension [39,40]. The use of PCA helps to find the most important sensors for a given context and optimizes both the context modeling and reasoning, which will be explained in detail in the reasoning step.

Once C_G is established, the current state space S_c and the present context C_ρ are initialized. S_c is simply configured by the current status of sensors. Initialization of C_ρ is performed by the Algorithm 2. The purpose of the algorithm is to find an initial present context C_{ν} which has the highest similarity with S_c .

Algorithm 2. Initializing Context

INPUT: current state space S_c ; context graph C_G **OUTPUT:** the highest similarity scored contexts C_{ν}

- 1. Next context $C_{\nu} \leftarrow NULL$
- 2. **for** each context C_{τ} in C_G
- 3. **if** similarity (S_c, C_τ) > similarity (S_c, C_ν)
- 4. then $C_{\nu} \leftarrow C_{\tau}$ //replace C_{τ} as a next context
- 5. end if
- 6. end for

4.1.2. REASONING

When the current state space S_c changes due to a sensor event, an attempt is made to reason a context that is most similar to S_c , as shown in Algorithm 3. If a proper context model is found, it becomes the new present context C_{ρ} . As in Algorithm 2, the similarity between S_c and each target context C_{τ} is calculated. The proposed algorithm can reduce the number of target contexts when calculating the score due to C_G . By the definition of C_G , since C_{ρ} is followed by C_N of C_{ρ} , only contexts in C_N are considered in the reasoning process. If the reasoning process is unable to find any context model, that is, C_v is empty (line 7 in Algorithm 1), it does not transition to the next context, and the current context C_{ρ} continues. Algorithm 3. Reasoning Context

INPUT: current state space S_c ; present context C_ρ ; context graph C_G					
OUTPUT: the highest similarity scored contexts C_{ν}					
1.	Next Contexts $C_N \leftarrow$ find all next contexts of C_ρ in C_G				
2.	Set of Candidates Contexts $\hat{C}_N \leftarrow NULL$				
3.	<i>for each context</i> C_{τ} in C_N				
4.	if similarity condition between S_c and C_{τ} is matched				
5.	then $\hat{C}_N \leftarrow \ \hat{C}_N \cup C_{ au}$ //add $C_{ au}$ to \hat{C}_N as a candidate				
6.	end if				

- 7. end for
- 8. $C_{\nu} \leftarrow$ the highest similar scored C_{τ} in \hat{C}_N .

Differing from Algorithm 2, which selects any context and declares it as C_{ρ} , more consideration is needed to discover the possible next contexts C_N . If a context is too different from S_{c_r} it must not have a chance to become C_{ρ} . In some cases, contexts in C_N have too low a similarity and should not be any context. To address the cases, a context requires filtering by a threshold θ , which stands for a particular similarity grade. We will introduce the reasoning methods and define θ in the next section.

4.2. Context Graph Building Steps

The approach proceeds in three steps, each of which utilizes a statistical method. First, the number of contexts is decided by a conditional probability table (CPT), and then meaningful and representative state spaces are defined as contexts by K-means clustering. Lastly, important sensors, which are related to each context, are discovered by using principal component analysis.

4.2.1. Deciding the Number of Contexts

To find the number of contexts, we first capture the probability of the consecutive occurrence of each pair of different sensors in the datasets. The idea is that sensors in one context are related and, thus, the occurrence probability is fairly high [41]. In other words, if the occurrence probability of a pair of sensors is low, the sensors are not in a context. This probability can be accurately calculated from the frequency of occurrence of consecutive sensor events. These conditional probabilities are arranged as a $\zeta \times \zeta$ table (ζ being the number of sensors), which is called the conditional probability table (CPT). Figure 3 shows how to decide on a pair of consecutive sensor events and how to build a CPT.



Conditional Probability Table

Figure 3. A conditional probability table (CPT) consists of conditional probability of each pair of sensor events. Conditional probability reveals how frequently a sensor event occurs in a given scenario. A conditional probability of 0 means the sensor event never happened.

CPT is used as the probabilistic fingerprint of the entire dataset. A pair of sensor events with high conditional probability usually contains sensor events that are related and associated with each other. They could belong to the same context and, therefore, are highly likely to occur together in this order. On the other hand, if the pair has low conditional probability, its sensor events are considered to be unrelated and would rarely occur together. A pair of sensor events with low probability indicates the end of one context and the start of another. Therefore, we divide the dataset between sensor events e_i and e_{i+1} if the conditional probability of e_i and e_{i+1} satisfies the condition $P(e_{i+1}) \leq \theta \times P(e_i)$, where θ is a parameter that represents the extent to which e_{i+1} relates to e_i . We set θ to 0.5 in the experiments. Using this method, we divide the dataset into *k* groups, which are considered the number of contexts.

4.2.2. Defining Contexts

We observe that a meaningful state space is sufficiently distant from other meaningful state spaces but could be close to other relevant, yet non-meaningful, state spaces. To find out which state spaces are meaningful, all spaces are partitioned into *k* clusters, in which each state space belongs to the cluster with the nearest mean. Therefore, the universal set of state spaces $S_{U} = \{S_1, \ldots, S_i, \ldots, S_{\zeta}\}$ is divided into $\{\hat{S}_1, \ldots, \hat{S}_i, \ldots, \hat{S}_{\zeta}\}$, where S_i is a state space and \hat{S}_i is a cluster of state spaces. Each cluster \hat{S}_i minimizes the sum of distances between the within-state space and the mean, according to the following formula:

$$\arg\min_{S_{u}} \sum_{i=1}^{k} \sum_{S_{t} \in \hat{S}_{i}} \|S_{t} - \mu_{i}\|^{2},$$
(6)

where S_i means a state space in cluster S_i . After S_U is classified into k clusters, cluster centroids are considered meaningful state spaces and are candidates for contexts. Note the number of contexts obtained from the previous step. Figure 4 illustrates an example of how contexts are clustered and chosen from a set of state spaces.



Figure 4. State spaces are clustered to capture contexts where the number of sensors ω is 3. In this example, the number context *k* is 7. The centroid of each cluster is defined as context and highlighted by red spots.

4.2.3. Discovering Context Conditions

The centroid in context is representative and meaningful but has insufficient information to define the context. We observed that multiple sensors usually contribute to beginning a context. Principal components analysis (PCA) can discover these important sensors. Once we find the relevant sensors via the stochastic analysis of the sensors' high-dimensional data, the original dataset can be projected onto lower-dimensional data. The process is repeated for each cluster, and the remaining data are used to build context conditions.

Principal components are sensors that show definite variance patterns that explicitly express a change of states. To discover these sensors, we should understand the pattern in which the dataset is scattered. For this, a matrix of covariances (*cov*) is calculated first. In a ζ -dimensional dataset, covariance *cov* is calculated by Equation (7):

$$cov \ (\hat{s}^{i}, \hat{s}^{j}) = \frac{\sum_{k=1}^{\omega} (\hat{s}^{i}_{k} - \mu_{i}) (\hat{s}^{j}_{k} - \mu_{j})}{(\zeta - 1)},$$
(7)

where \hat{s}^i and \hat{s}^j stand for the set of sensor values in *i*th state space and *j*th state space, respectively. *i*th state space and *j*th state space belong to the same context. Since the maximum number of state spaces in a context is ζ , the total covariances establish a $\zeta \times \zeta$ covaraince matrix *R*, shown in the Equation (8):

$$\boldsymbol{R}(\hat{\boldsymbol{S}}) = \begin{bmatrix} cov\left(\hat{\boldsymbol{s}}^{1}, \hat{\boldsymbol{s}}^{1}\right) & \dots & cov\left(\hat{\boldsymbol{s}}^{1}, \hat{\boldsymbol{s}}^{\zeta}\right) \\ \vdots & \ddots & \vdots \\ cov\left(\hat{\boldsymbol{s}}^{\zeta}, \hat{\boldsymbol{s}}^{1}\right) & \dots & cov\left(\hat{\boldsymbol{s}}^{\zeta}, \hat{\boldsymbol{s}}^{\zeta}\right) \end{bmatrix}$$
(8)

From the covariance matrix R, we calculate the eigenvectors, which characterize the square matrix as linear equations, each of which can transform the matrix onto a new axis. Eigenvalues then measure how well the sensor data are scattered. Eigenvectors and eigenvalues are calculated by the following condition: $R\nu = \lambda\nu$, where ν is the eigenvectors and λ is the eigenvalues. As R is a $\zeta \times \zeta$ matrix, ν is also a $\zeta \times \zeta$ matrix, that is ζ eigenvectors exist. Each eigenvector has its own eigenvalue; therefore, there are ζ eigenvalues in λ :

$$\mathbf{v} = \begin{bmatrix} \vec{elg_1} & \dots & \vec{elg_l} & \dots & \vec{elg_\zeta} \end{bmatrix}, \tag{9}$$

where elg_l is an eignevetor with ζ components.

Next, we choose the eigenvectors with high eigenvalues as principal components. The challenge is in determining the threshold for which eigenvalues are high enough to be acceptable. We propose the threshold θ_e for the average of eigenvalues of selected eigenvectors. In our approach, the eigenvectors are sorted by eigenvalues in descending order; then, eigenvectors with values that exceed θ_e are chosen. Eigenvectors satisfying this condition establish a feature matrix *F*:

$$F = \begin{bmatrix} \vec{elg_1} & \dots & \vec{elg_\kappa} \end{bmatrix},$$
(10)

where $\kappa \leq \zeta$, $\sum_{i=1}^{\kappa} eiv_i \leq \theta_e$, eiv_i is an eigenvalue of elg_l . With a feature matrix *F*, a given dataset is transformed into lower-dimensional data with κ sensors. The conditions for a context are a collection of sensor values in the lower-dimensional data.

4.3. Context Reasoning Methods

To reason a context, the similarity between S_c and each possible next context C_{τ} ($\in C_N$) is considered by calculating $sim(S_c, C_{\tau})$. Since the state space and context conditions are vectorized by the status of sensors, $sim(S_c, C_{\tau})$ can be computed by two methods, as shown below.

4.3.1. Similarity Based on Euclidean Distance

When the distance is within a fuzzy threshold, $\theta_d^{C_{\tau}}$, which specfies a range for the condition of C_{τ} , signifies no difference between S_c and C_{τ} ; hence, C_{τ} becomes the next context. Note that $\theta_d^{C_{\tau}}$ is context-specific; thus, each context has its own value. This is obtained by averaging the distances from each state space to context during the initialization step [4]. Some conditions may be defined with a range of values, instead of one value. In this case, to apply Euclidean distance, the range is represented by a median of the range. For instance, if the range of sensor's values is from 0 to 5 in an integer, its median, which is 3, is used to compute the Euclidean distance. The Euclidean distance is calculated by the following Equation (11):

$$dist(S_{c}, C_{\tau}) = \|S_{c} - C_{\tau}\| = \sqrt{\sum_{i=1}^{\omega} (s_{i}^{S_{c}} - s_{i}^{C_{\tau}})^{2}}$$
(11)

In Equation (12), $s_i^{S_c}$ stands for the status of sensor s_i in the current state space S_c , and $s_i^{C_{\tau}}$ stands for the status of sensor s_i in the possible next context, C_{τ} .

If the distance is below $\theta_d^{C_{\tau}}$, the C_{τ} is considered to possibly be the next context. On the other hand, when the distance is greater than $\theta_d^{C_{\tau}}$, there is no chance for S_c to become C_{τ} . The similarity is calculated and compared for all C_{τ} which satisfy the distance condition, that is, $dist(S_c, C_{\tau}) \leq \theta_d^{C_{\tau}}$:

$$sim_{dist}(S_{c}, C_{\tau}) = 1 - \frac{dist(S_{c}, C_{\tau})}{\|S_{c}\|} = 1 - \frac{dist(S_{c}, C_{\tau})}{\sqrt{\sum_{i=1}^{\omega} (S_{c}^{S_{c}})^{2}}}$$
(12)

If none of C_{τ} verifies the condition, the current state space cannot not be any of the contexts, therefore no change in context occurs.

4.3.2. Similarity Based on Cosine Similarity

Dissimilar to the Euclidean distance, which values the distance between the end points of two vectors, cosine similarity verifies the appropriateness of the angle between two vectors. The $sim_{cos}(S_c, C_{\tau})$ is calculated as follows:

$$sim_{cos}(S_{c}, C_{\tau}) = \frac{S_{c} \cdot C_{\tau}}{\|S_{c}\| \cdot \|C_{\tau}\|} = \frac{\sum_{i=1}^{\omega} s_{i}^{S_{c}} \cdot s_{i}^{C_{\tau}}}{\sqrt{\sum_{i=1}^{\omega} (s_{i}^{S_{c}})^{2}} \cdot \sqrt{\sum_{i=1}^{\omega} (s_{i}^{C_{\tau}})^{2}}}$$
(13)

As the cosine of two vectors increases to 1, the vectors are considered more similar. All of C_{τ} in $sim_{cos}(S_c, C_{\tau})$ must be within the threshold $\theta_a^{C_{\tau}}$ for the condition of C_{τ} to be considered the possible next context. Failure to find such C_{τ} delays the transition to any possible next context.

 $sim_{dist}(S_c, C_{\tau})$ fits when finding the similarity of state spaces with Boolean-stated sensors, such as sensors generating two statuses of 0/1 or On/Off. In other words, when the change in sensor status is monotonous and, thus, the magnitude of vectors S_c and C_{τ} is stable, the Euclidean distance can be applied. However, this is not applicable when the status in the sensors is non-Boolean and generates values within certain ranges since the magnitude of vectors can significantly change. For the state spaces containing those sensors, $sim_{cos}(S_c, C_{\tau})$ is used.

In the reasoning methods above, each vectorized element is individually calculated. As the given space is extended and the state space is scaled with more sensors, the computational complexity greatly increases and affects the performance in reasoning contexts. Since the number of sensors matters, it is critical to reduce the size of the sensors. To achieve this goal, we utilize PCA and reduce the dimension of the dataset for faster computation when calculating similarity. Note that the PCA is applied regardless of the similarity method.

5. Experimental Validation

Automation reduces time and effort for modeling the structures of a process and avoids potential errors when manually configuring a process. However, an automated process might downgrade performance due to the lack of an ability to catch and handle unexpected exceptions. The validation point for an automation-enabled approach is to consistently show results that are as good as those produced by non-automated approaches in various cases. Therefore, we performed experimental validation with case studies to show whether these contexts are reasoned by our approach without violation. In the experiment, we conducted a reasoning process for a day-long scenario and evaluated whether a set of appropriate contexts was sequentially recognized. The contexts located within the set are considered as a contextual schedule, notated by $C_S \subseteq C_G$. Thus, the validation aimed to analyze the C_s and decide whether this is reasonable and feasible. For our validation, we defined two measurements: Fidelity and Feasibility. Fidelity is used to measure the degree of reasonability of a C_{s} , and *Feasibility* is used to terminologically describe whether the $C_{\rm S}$ is at all possible by *Fidelity*. In short, *Fidelity* generates numeric measurements, and *Feasibility* divides the range of *Fidelity* to particular categories. The categories of *Fidelity* will be introduced after explaining how to calculate *Fidelity*. If a C_{s} has a high Fidelity, it could be said to happen with high probability and be categorized as "good" *Fidelity*. We verbally describe it as being feasible or possibly feasible.

Our approach was validated by showing how many C_S are feasible. For this goal, we conducted the following steps:

- 1. Preparing datasets: we built context models and context graph C_G by analyzing sensor datasets. In this step, we obtained datasets from real-world scenarios, which are used as training sensor datasets for forming C_G . Note that each sensor dataset is formed in a collection of state spaces, \hat{S} . Then, we prepared more datasets as testing sensor dataset also follows a form of \hat{S} .
- 2. Testing an approach: we reasoned the testing sensor datasets and obtained context schedule C_S . Note that we had to test each dataset; thus, there were multiple C_S , which were used to calculate fidelity.
- 3. Evaluating context schedules: we evaluated whether each context schedule C_S obtained in the testing step was feasible. In the evaluation, we analyzed that a C_S could be found along with context graph C_G built in the preparing step. If the C_S could be located, we could validate that the reasoning process was flawless. If we failed to find it, we considered two reasons. First, the C_S was not feasible, and this meant the reasoning process had flaws. Second, the C_G was not accurately established, which inferred that the modeling process was errored. Next, we will explain the conditions for evaluation algorithms.

First, the training sensor datasets were collected for 11 days in the Gator Tech Smart House (GTSH) [42]. The datasets captured the status of 19 sensors, which were deployed on 18 objects, while 8 activities were performed by 2 volunteer testers. The floor plan describes where the objects and sensors were placed, as shown in Figure 5. A daily living scenario in which 8 activities were performed was proposed in [43].

Figure 5. In our experiment, there were 19 sensors (indexed by Q#) deployed in the Gator Tech Smart house (GTSH). These sensors were attached to 18 objects (indexed by O#).

Next, we used the Persim 3D simulator [43], which generates the testing sensor datasets. Persim 3D simulates pre-defined context models and a context graph C_G based on a context-driven approach, and then synthesizes different datasets. We obtained 11 testing sensor datasets and generated 11 context schedules C_S using our reasoning process. We then evaluated the 11 C_S and proved that they could be discovered in C_G . For a more accurate evaluation, we proposed the following conditions and proved that they all were met in order:

Condition 1. $|C_S| \ge 2$, where $|C_S|$ stands for the number of elements (that is contexts) in the set C_S .

Condition 1 allows us to filter out cases of single contexts in which there is no transition of contexts; thus, the reasoning approach fails to find another context. In this case, a daily living scenario starts and ends in the same context without any transitions. This is considering the fact that the reasoning approach does not work properly. In our experiment, there was no testing sensor dataset that could not pass this condition. However, this does not guarantee that C_S is feasible, even though it consists of more than two contexts, and thus passes this condition. This condition is not enough to catch exceptionally occurring contexts, which may be accidentally sensed by sensor errors. For this reason, C_S needs to satisfy the next condition:

Condition 2. $P_{CS}(C_S) \ge OPT$, where $P_{CS}(C_S)$ stands for the occurrence probability of a given context schedule C_S , and OPT is threshold of the occurrence probabilities.

Condition 2 restricts to accept only the most likely occurring context schedules C_S . This proceeds with context graph C_G , which was made in the preparation step. Recall that C_G is built in a directed graph, which describes all transitions between contexts. This means that C_G can reveal various context paths, each of which is considered a C_S . Each C_s is evaluated by its probability of occurrence, denoted by $P_{CS}(C_S)$. Note that the low probability of occurrence of an event means that it rarely or never occurs. Thus, we can infer that a C_S with a low probability rarely occurs and, hence, it should rarely be reasoned.

The *Fidelity* and the *Feasibility* of C_S is measured based on the probability. For this goal, statistic models were defined as follows: First, we calculated the join probability

distribution of C_S based on use of the Bayesian network (BN) and conditional probability table for contexts. BN-based probability P_{BN} of C_S is computed by Equation (9):

$$P_{BN}(C_S) = \prod_{C_i, C_i \in C_S} P(C_i, C_j)$$
(14)

In Equation (9), $P(C_i, C_j)$ calculates the conditional probability of two distinct contexts, C_i followed by C_j . $P_{BN}(C_S)$ presents the probability for occurrence of a given C_S ; however, it is not enough to measure the *Fidelity* and the *Feasibility*. Even if a C_S has a high occurrence probability but it rarely occurs, it could be considered as an exception or an error, and thus should have a low *Fidelity*. To resolve this, *Odds* for C_S is calculated by Equation (10). $P(C_S)$ stands for the probability of a C_S , and denotes how often the C_S occurs. In this experiment, it can be computed by the frequency of occurrence of a C_S over 11, the number of total training (or testing) cases.

$$O_{BN}(C_S) = \frac{1 - P(C_S)}{P(C_S)} \tag{15}$$

The $P_{BN}(C_S)$ and the $O_{BN}(C_S)$ are used for computing occurrence probability, $P_{CS}(C_S)$. $P_{CS}(C_S)$ is used for determining *OPT*. *OPT* is obtained by calculating an arithmetic average of $P_{CS}(C_S)$ for all given C_S . In our experiment, *OPT* was 0.301 for the training datasets.

$$P_{CS}(C_S) = P_{BN}(C_S) * O_{BN}(C_S)$$
(16)

In the evaluation step, we repeated the entire process with testing datasets, which are generated by Persim 3D, and we obtained $P_{BN}(C_S)$, $O_{BN}(C_S)$, and $P_{CS}(C_S)$. Then, we conducted the validation process by computing *Fidelity* and deciding *Feasibility*. *Fidelity* is measured by the ratio of the $P_{CS}(C_S)$ of the testing sensor datasets over *OPT*. If the $P_{CS}(C_S)$ is below *OPT*, it means that the C_S is unreasonable and irrational to occur. The maximum *Fidelity* is 100%, which implies that the C_S of the testing dataset occurred in the same probability.

However, even though the *Fidelity* of a C_S is not 100%, it is hard to decide whether it is not yet reasonable. Since $P_{CS}(C_S)$ and *OPT* were calculated by statistical analysis based on probability distribution, this may be adjusted as more training datasets are collected and used. Therefore, we categorized the *Fidelity* into three states, which increased the flexibility of evaluation. We defined the *Feasibility* states as falling into the three *Fidelity* categories, as shown in Table 1. A C_S with 100% *Fidelity* is classified into the category *Feasible*. If the *Fidelity* of a C_S is below 100%, yet considerably low, we assume that the C_S may be feasible in certain conditions, and thus accept it even though it is not guaranteed to occur. In our experiment, we analyzed the training datasets to determine the proper range for the category *Possible* by using Equation (17) and obtained 64%:

$$F_{ratio} = \frac{\left| \left\{ C_{S_i} \mid P_{CS}(C_{S_i}) \ge OPT \right\} \right|}{\left| \left\{ C_{S_i} \right\} \right|} \tag{17}$$

Table 1. Three categories for Fidelity by Feasibility states.

Category	Range	Perception about a Given C_{S_i}	
Feasible	100%	Feasible and solid	
Possible	$\geq 64\%$, and <100%	Probably feasible and acceptable	
Infeasible	< 64%	Not feasible	

In Equation (17), C_{S_i} stands for the context schedule obtained from an *i*th dataset, and F_{ratio} is a ratio representing how many context schedules are beyond the average. This is calculated by the number of C_{S_i} whose occurrent probability is above *OPT*, over the total number of C_{S_i} .

The *OPT* of the 11 testing sensor datasets is shown in Table 2. The *Fidelity* score of six datasets shows 100%, which means that they are *Feasible*, and those of the other three datasets are categorized in the *Possible* state $\geq 64\%$, and <100%), which means they might be acceptable. Hence, we consider that nine datasets are feasible. On the other hand, the *Fidelity* score of two datasets (no. 4 and 9) are not in the feasible or possible range, showing that they are below the minimum possible score, which is 64%. This means that the contextual schedules C_S of the datasets have very low conditional probability and thus very rarely occur. In summary, the proposed approach's overall success in reasoning contexts is 81%.

Table 2. Decision table by *Fidelity* and *Feasibility* based on occurrence probability. When *Feasibility* is between 64% and 100%, we declare that it is possible according to the three categories defined in Table 1.

Dataset No	$P_{BN}(C_S)$	$P_{CS}(C_S)$	Fidelity	Feasibility
1	0.048	0.214	71.11%	Possible
2	0.154	0.412	100%	Feasible
3	0.206	0.360	100%	Feasible
4	0.005	0.051	16.91%	Infeasible
5	0.048	0.214	71.11%	Possible
6	0.048	0.214	71.11%	Possible
7	0.154	0.412	100%	Feasible
8	0.206	0.360	100%	Feasible
9	0.015	0.153	50.72%	Infeasible
10	0.206	0.360	100%	Feasible
11	0.206	0.360	100%	Feasible

6. Discussion

In this paper, we proposed an approach for the automation of modeling and reasoning of contexts in smart spaces by stochastically analyzing the sensor data. Traditionally, and still in many applications, contexts are semantically and manually defined and configured using ontology-based modeling methods. Those methods enable well-organized structures for contexts and can be applied in human-centered machine and human-oriented smart spaces since they are readable and associable for human users. However, manual configuration requires human efforts and cannot be an optimal solution in this pandemic, which demands fewer physical contacts or assistance. The main contribution of our approach is to reduce humans' efforts in defining contexts and recognizing a current context and enables automation of the process. To achieve this goal, we employed a mathematical and statistical analysis of the data collected from sensors attached in smart spaces. First, we utilized a conditional probability-based analysis, K-means clustering, and PCA to build context models. In the processing used to determine a present context, we applied Euclidean distance, cosine similarity, and PCA. In all processes, when defining and recognizing a context, PCA was used to filter out sensors that had no contribution or minimal contributions and, eventually, to reduce the computational complexity.

As shown by experiments with positive performance results, our approach was shown to be validated and applicable. In the future, we plan to enhance the approach, which can dynamically adjust context models, which were already built while running a reasoning process. The current approach is generally based on the supervised learning method, and thus establishes a static model, which will be used to reason a present context. It is possible to fail to determine any context if the current state is very new or unexpectedly occurs due to forces external to the smart space, such as a natural disaster or the occurrence of unknown diseases. To address this issue, our future research will examine unsupervised and more dynamic learning methods and assesses their success in recognizing present contexts without training, and then modify the context models. The enhancement will include research to discover unknown contexts in the reasoning process, define them, and update the context graph. A related approach, which is equally important, is avoiding "impermissible contexts", which are known a priori but are not permitted to occur [44]. We will also develop methods to optimally map the present context to the most relevant services for the human residents in the space. This would improve real-world applications so that they work without training steps.

Author Contributions: Conceptualization, J.W.L. and A.H.; methodology, J.W.L. and A.H.; software, J.W.L.; validation, J.W.L. and A.H.; formal analysis, J.W.L. and A.H.; investigation, J.W.L.; resources, J.W.L.; data curation, J.W.L.; writing—original draft preparation, J.W.L. and A.H.; writing—review and editing, J.W.L. and A.H.; visualization, J.W.L.; supervision, J.W.L.; project administration, J.W.L.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data supporting the reported result are available from the corresponding author on request.

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

References

- 1. Li, R.Y.M.; Li, H.C.Y.; Mak, C.K.; Tang, T.B. Sustainable Smart Home and Home Automation: Big Data Analytics Approach. *Int. J. Smart Home* **2016**, *10*, 177–198. [CrossRef]
- Ernst, E.; Ostermann, K.; Cook, W.R. A virtual class calculus. In Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Charleston, SC, USA, 11–13 January 2006; pp. 270–282. [CrossRef]
- Bühler, D.; Küchlin, W.; Grubler, G.; Nusser, G. The Virtual Automation Lab-Web based teaching of automation engineering concepts. In Proceedings of the Proceedings of IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Edinburgh, UK, 3–7 April 2000; pp. 156–164. [CrossRef]
- 4. Baker, S.B.; Xiang, W.; Atkinson, I. Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities. *IEEE Access* 2017, *5*, 26521–26544. [CrossRef]
- 5. Kang, H.S.; Lee, J.Y.; Choi, S.; Kim, H.; Park, J.H.; Son, J.Y.; Kim, B.H.; Noh, S.D. Smart manufacturing: Past research, present findings, and future directions. *Int. J. Precis. Eng. Manuf. Green Technol.* **2016**, *3*, 111–128. [CrossRef]
- D'Alessandro, D.; Gola, M.; Appolloni, L.; Dettori, M.; Fara, G.M.; Rebecchi, A.; Settimo, G.; Capolongo, S. COVID-19 and Living space challenge. Well-being and Public Health recommendations for a healthy, safe, and sustainable housing. *Acta Biomed* 2020, 91, 61–75. [PubMed]
- 7. Chen, T.; Lin, C.-W. Smart and automation technologies for ensuring the long-term operation of a factory amid the COVID-19 pandemic: An evolving fuzzy assessment approach. *Int. J. Adv. Manuf. Technol.* **2020**, *111*, 3545–3558. [CrossRef]
- Taiwo, O.; Ezugwu, A.E. Smart healthcare support for remote patient monitoring during COVID-19 quarantine. *Informatics Med.* Unlocked 2020, 20, 100428. [CrossRef] [PubMed]
- Murphy, L.; Eduljee, N.B.; Croteau, K. College Student Transition to Synchronous Virtual Classes during the COVID-19 Pandemic in Northeastern United States. *Pedagog. Res.* 2020, 5, em0078. [CrossRef]
- 10. Maalsen, S.; Dowling, R. COVID-19 and the accelerating smart home. Big Data Soc. 2020, 7. [CrossRef]
- Abowd, G.D.; Dey, A.K.; Brown, P.J.; Davies, N.; Smith, M.; Steggles, P. Towards a Better Understanding of Context and Context-Awareness. In Proceedings of the International Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, 27–29 September 1999; Volume 1707, pp. 304–307. [CrossRef]
- Yau, S.; Wang, Y.; Karim, F. Development of situation-aware application software for ubiquitous computing environments. In Proceedings of the Proceedings Annual International Computer Software and Applications, Oxford, UK, 26–29 August 2002; pp. 233–238. [CrossRef]
- Lee, J.W.; Helal, A.; Sung, Y.; Cho, K. A context-driven approach to scalable human activity simulation. In Proceedings of the ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, Montreal, QC, Canada, 19–22 May 2013; pp. 373–378. [CrossRef]
- 14. Chattoraj, S. Smart Home Automation based on different sensors and Arduino as the master controller. *Int. J. Sci. Res. Publ.* **2015**, *5*, 736–739.
- Singh, H.; Pallagani, V.; Khandelwal, V.; Venkanna, U. IoT based smart home automation system using sensor node. In Proceedings of the International Conference on Recent Advances in Information Technology, Dhanbad, India, 15–17 March 2018; pp. 1–5. [CrossRef]
- 16. Lin, H.; Bergmann, N.W. IoT Privacy and Security Challenges for Smart Home Environments. Information 2016, 7, 44. [CrossRef]
- 17. Strang, T.; Linnhoff-Popien, C. A context modeling survey. In Proceedings of the International Conference on Ubiquitous Computing, Nottingham, UK, 7–10 September 2004. [CrossRef]

- 18. Bettini, C.; Brdiczka, O.; Henricksen, K.; Indulska, J.; Nicklas, D.; Ranganathan, A.; Riboni, D. A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput.* **2010**, *6*, 161–180. [CrossRef]
- 19. Fischer, G. User Modeling in Human–Computer Interaction. User Model. User-Adapted Interact. 2001, 11, 65–86. [CrossRef]
- 20. Obrenovic, Z.; Starcevic, D. Modeling multimodal human-computer interaction. Computer 2004, 37, 65–72. [CrossRef]
- Sierhuis, M.; Clancey, W.J.; Hoof, R.V.; Hoog, R.D. Modeling and simulating human activity. In Proceedings of the AAAI 2000 Fall Symposium on Simulating Human Agents, North Falmouth, MA, USA, 3–5 November 2000; pp. 100–110.
- Park, J.; Moon, M.; Hwang, S.; Yeom, K. CASS: A Context-Aware Simulation System for Smart Home. In Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management & Applications, Busan, Korea, 20–22 August 2007; pp. 461–467.
- 23. Bolchini, C.; Orsi, G.; Quintarelli, E.; Schreiber, F.A.; Tanca, L. Context Modelling and Context Awareness: Steps forward in the Context-ADDICT project. *Bull. IEEE Tech. Comm. Data Eng.* **2011**, *34*, 47–54.
- 24. Salber, D.; Dey, A.K.; Abowd, G.D. The context toolkit. In Proceedings of the Conference on Human Factors in Computing Systems, Pittsburgh, PA, USA, 15–20 May 1999; pp. 434–441. [CrossRef]
- 25. Hasan, M.; Roy-Chowdhury, A.K. Context Aware Active Learning of Activity Recognition Models. In Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 4543–4551. [CrossRef]
- Abdallah, Z.S.; Gaber, M.; Srinivasan, B.; Krishnaswamy, S. CBARS: Cluster Based Classification for Activity Recognition Systems. In Proceedings of International Conference on Advanced Machine Learning Technologies Applications, Cairo, Egypt, 8–10 December 2012; pp. 82–91. [CrossRef]
- Hoque, E.; Stankovic, J. AALO: Activity recognition in smart homes using Active Learning in the presence of Overlapped activities. In Proceedings of the International Conference on Pervasive Computing Technologies for Healthcare and Workshops, San Diego, CA, USA, 21–24 May 2012; pp. 139–146. [CrossRef]
- 28. Jurek, A.; Nugent, C.; Bi, Y.; Wu, S. Clustering-Based Ensemble Learning for Activity Recognition in Smart Homes. *Sensors* 2014, 14, 12285–12304. [CrossRef] [PubMed]
- 29. Choi, W.; Shahid, K.; Savarese, S. Learning context for collective activity recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011; pp. 3273–3280.
- Wang, M.; Ni, B.; Yang, X. Recurrent Modeling of Interaction Context for Collective Activity Recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3048–3056.
- Ejigu, D.; Scuturici, M.; Brunie, L. An Ontology-Based Approach to Context Modeling and Reasoning in Pervasive Computing. In Proceedings of the Annual IEEE International Conference on Pervasive Computing and Communications Workshops, New York, NY, USA, 19–23 March 2007; pp. 14–19. [CrossRef]
- 32. Aguilar, J.; Jerez, M.; Rodríguez, T. CAMeOnto: Context awareness meta ontology modeling. *Appl. Comput. Inform.* 2018, 14, 202–213. [CrossRef]
- Wang, X.H.; Zhang, D.Q.; Gu, T.; Pung, H.K. Ontology based context modeling and reasoning using OWL. In Proceedings of the IEEE Annual Conference on Pervasive Computing and Communications Workshops, Orlando, FL, USA, 14–17 March 2004; pp. 18–22. [CrossRef]
- 34. Wu, X. Context-Aware Cloud Service Selection Model for Mobile Cloud Computing Environments. *Wirel. Commun. Mob. Comput.* **2018**, 2018, 1–14. [CrossRef]
- 35. Shin, D.; Lee, J.-W.; Yeon, J.; Lee, S.-G. Context-Aware Recommendation by Aggregating User Context. In Proceedings of the IEEE Conference on Commerce and Enterprise Computing, Vienna, Austria, 20–23 July 2009; pp. 423–430. [CrossRef]
- 36. Madadipouya, K.; Chelliah, S. A literature review on recommender systems algorithms, techniques and evaluations. *Brain: Broad Res. Artif. Intell. Neurosci.* 2017, *8*, 109–124.
- 37. Stepanov, I.; Marron, P.; Rothermel, K. Mobility Modeling of Outdoor Scenarios for MANETs. In Proceedings of the 38th Annual Simulation Symposium, San Diego, CA, USA, 4–6 April 2005; pp. 312–322. [CrossRef]
- Sundaramoorthi, D.; Chen, V.P.; Kim, S.B.; Rosenberger, J.M.; Buckley-Behan, D.F. A Data—Integrated Nurse Activity Simulation Model. In Proceedings of the Winter Simulation Conference, Monterey, CA, USA, 3–6 December 2006; pp. 960–966. [CrossRef]
- 39. Eklundh, L.; Singh, A. A comparative analysis of standardised and unstandardised Principal Components Analysis in remote sensing. *Int. J. Remote Sens.* **1993**, *14*, 1359–1370. [CrossRef]
- 40. Chang, C.-I.; Du, Q. Interference and noise-adjusted principal components analysis. *IEEE Trans. Geosci. Remote Sens.* **1999**, 37, 2387–2396. [CrossRef]
- Sung, Y.; Helal, A.; Lee, J.W.; Cho, K. Bayesian-based scenario generation method for human activities. In Proceedings of the ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, Montreal, QC, Canada, 19–22 May 2013; pp. 147–158. [CrossRef]
- 42. 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. [CrossRef]
- 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 Trans. Autom. Sci. Eng.* 2015, 12, 1243–1256. [CrossRef]
- Chen, C.; Helal, A.; Jin, Z.; Zhang, M.; Lee, C. IoTranx: Transactions for Safer Smart Spaces. ACM Trans. Cyber-Physical Syst. 2022, 6, 1–26. [CrossRef]