Inferring Contexts from Human Activities in Smart Spaces

Jae Woong Lee
Dept. of Mathematics and Computer Science
University of Central Missouri
jwlee@ucmo.edu

Sumi Helal
University of Florida
helal@cise.ufl.edu

Abstract
Modeling and simulation of human activities is becoming a hot research area for validating activity recognition algorithms used to generate useful synthetic datasets for assistive environments and other smart spaces. Context-driven simulation, an emerging approach that utilizes abstract structures of state spaces (contexts), can enhance the scalability and realism of simulations. However, the context-driven approach is demanding of users’ efforts in specifying not only activity models, but also the corresponding contexts and contextual transitions associated with these activities. In this paper, we propose a method to reduce users’ efforts in configuring simulation by using $k$-means clustering and principal component analysis approaches to automate the derivation of contexts from a given set of activities. We validate our approach by comparing the actual sequenced activities with the derived sequenced activities.

Introduction
A key challenge in using smart technology, such as smart homes, to support independent living of the elderly and people with chronic diseases, is to monitor activities through sensors and recognize emergency situations through programmable and automatic recognition. Research in recognition systems requires a large number of datasets and test beds that can evaluate the effectiveness and accuracy of newly developed algorithms or activity models. Unfortunately, relevant datasets are scarce. Simulation of human activities in smart home environments is therefore becoming very important as an alternative source of datasets for researchers (Hadidi and Noury 2010, Hubeischer and McCann 2004, Jahromi et al. 2011).

We addressed these issues and developed Persim 3D (Lee et al. 2015). This simulator is based on a context-driven approach that addresses scalability (Lee et al. 2013). It is also based on a special activity-modeling approach, activity playback, that enhances realism (Liu et al. 2015). In Persim 3D, context was introduced as a new abstract simulation entity that represents consecutively occurring space states and defines a set of available activities that can be performed if such a simulation context is reached.

All contexts of a simulation scenario are formed into a context graph, which can automatically schedule activities. The simulator steps through and drives the contexts instead of stepping through events, as traditional event-driven simulators do. This decreases the computational complexity and achieves scalability. When an activity needs to be performed, an activity playback algorithm dynamically instantiates the activity. The dynamic modeling method increases simulation realism and supports automatic generation of activity scenarios.

However, it is difficult for users to define contexts because they are not only conceptually abstracted from state spaces but also recognized differently. Therefore, it is important to derive contexts from activities algorithmically. This will reduce the burden on human efforts and errors in manually designing contexts.

It is also important to solve the problem of deriving contexts from activities outside our Persim 3D simulation framework. One key application that could help solve this is the enabling of context-aware programming of human-centered systems that rely heavily on activity recognition. By deriving contexts that correspond to activities, it becomes much easier for developers to code the system in the context domain rather than in the sensor domain. Deriving contexts of activities can also help provide additional means to double check recognition, which would raise the accuracy of the overall system.

In this paper, we propose to derive contexts from simulation models of activities. All state spaces obtained by simulating each activity are classified into clusters of relevant state spaces using $k$-means clustering. The centroid in each cluster is defined as a context, which is meaningful and representative of other state spaces in the cluster. Each
context is also built with potential conditions to define it via Principal Component Analysis (PCA). Finally, a context graph including all the contexts is formed and provided for simulation.

This paper is organized as follows. In the next section, we describe existing work related to activity modeling, simulation of human activities, and cluster-based classification that can be used for derivation of contexts. Then, we present the overall approach of derivation and explain detailed steps of the proposed approach. Finally, we discuss computational analysis and experimental validation.

**Related Work**

There are multiple studies available about modeling human activity from sensor data, and simulating such models have been carried out using various approaches. Sierhuis et al. (2000) argued that elements such as “fact” and “belief” are an integral part of a model for human astronauts’ activities in outer space. Simulation was performed based on real data collected from space experiments in a NASA project. Stepanov et al. (2005) 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 humans’ mobility. Sundarmoorhi et al. (2006) focused more on activities of nurses caring for patients in a hospital setting with the goal of maximizing scheduling efficiency and minimizing downtime to 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 in simulating simple activities of daily life, such as having breakfast.

In the context-driven approach (Lee et al. 2013), abstract and representative state spaces are defined as contexts, which facilitate the automated generation of indoor activity scenarios. By specifying related and possible activities in each context, different activities are dynamically scheduled and generated during simulation, leading to automated generation of multiple scenarios using fixed human efforts. There is also research in activity recognition available on deriving meaningful high-level information from low-level information. The goal of that research is to classify clusters from collected sensor datasets. In CBARS (Abdallah et al. 2012), a supervised learning model was built first, and unsupervised learning for new data was applied for new activity recognition. The challenge was that CBARS needed a supervised learning model. AALO (Hoque and Stankovic 2012) addressed that challenge. AALO is an active recognition system that can accurately classify specified activities according to locations and times in which the activities are performed. CBCE (Jurek et al. 2014) proposed a method for combing multiple classifiers including Naïve Bayes (NB) models, hidden Markov models (HMMs), and conditional random fields (CRFs). This ensemble of classifiers can recognize activities in given sensor datasets, however, they do not provide a method to define abstract information of context to represent the other state spaces in a cluster.

**Overall Approach**

Our basic concept was to start with a set of given activities, simulate them to generate a sequence of states that represent their effects, and then utilize clustering and PCA techniques over the state space to infer a set of contexts that correspond to activities, along with their context graphs. We first find all possible state spaces by applying the activity playback algorithm (Liu et al. 2015), which plays all activities defined using the activity playback model. The proper state spaces are generated and then formed into a state space graph $S_G$. That is converted to a context graph $C_G$, which consists of only contexts. $C_G$ illustrates causalities of all contexts and hence is capable of generating activity scenarios. Once contexts are identified, context conditions, which govern how to transition between contexts in the graph, are also derived. This is done through a PCA to find all potential context conditions for each context.

Before we dive further into the details of our approach, it would first be helpful to define context. It is an abstracted state space envelope that represents consecutively occurring state spaces (Lee et al. 2013). A context is intended to represent an important state space in the group of relevant state spaces with respect to activities. Figure 1 shows the process of deriving such contexts by building clusters of state spaces, finding the important state space in each cluster using a $k$-means clustering technique, and declaring the centroid of each cluster as context.

![Figure 1. Clustering state spaces to capture seven contexts. The centroid of each cluster is defined as context (red spots).](image)

**Activity-Context Interplay**

In context-driven simulation, the correlation between activities and contexts is important. In a given context, certain activities that are related to that context can be selected and scheduled (Lee et al. 2013). When the scheduled activities...
are performed (played back), they can change the state space and eventually transition into the next context.

The characteristics of correlation between contexts and activities are used in designing context-driven simulation scenarios and in generating contexts from activities. The remainder of this section introduces the necessary modeling and structures for activities, contexts, and their interplay.

**Context modeling**

Context is defined by three properties: context conditions, which express conditions to enter the context, context activities, which are activities available in the context (for play back of some of them), and next contexts, which can be transitioned to after activities are performed.

**Activity modeling**

To run context-driven simulation properly, a specialized activity model is needed to focus on the realism of visually simulating the activity. It must be cognizant of the interplay between activities and their corresponding contexts. The activity playback model (Liu et al. 2015) is designed to generate dynamically varying and more realistic activities while reflecting the effect of the playback on the space states, hence assisting in navigation between contexts. As we will see, activity playback enables the generation of state spaces, which we use to create the context graph.

**Context graph**

All defined contexts can be formed into a context graph $C_G$, in which a node presents a context, and an edge specifies a transition condition between two contexts. The transition may occur when an activity is executed in one context, and therefore, an edge condition must contain the activity that would go to another context. Multiple activities may be required in a single transition condition.

**State space graph**

The state space graph $S_G$ is an intermediate structure to obtain the desired $C_G$. It consists of all available state spaces, whereas $C_G$ is formed by only meaningful state spaces (contexts). Thus, a node in $S_G$ is a state space. In $S_G$, an edge contains only one activity performed, and activity execution immediately moves to the next node (state space) that it is connected to. However, an edge in $C_G$ can contain multiple activities, and transitioning between nodes (contexts) may not happen after execution of an activity. For instance, at the moment a resident walks to the bathroom after a meal, there is no explicit change of context of personal hygiene begins. Lee et al. (2013) showed that transitioning could happen anytime while activities are being performed if activities change the state space to become similar enough to another context.

Despite these differences, $S_G$ is related to $C_G$ because a context is a substitute for a meaningful state space. Therefore, $S_G$ could be converted into $C_G$ by finding the meaningful state spaces and pruning other state spaces. Such state spaces can be discovered through $k$-means clustering: the mean of each cluster is the most meaningful state space, which is the context. The next section will describe all the phases to build $C_G$ using $k$-means clustering and PCA.

**Derivation Phases**

**Phase 1: Constructing a State Space Graph**

In our approach, which is based on activity-context interplay, the first phase is to construct an $S_G$ from given activities. To obtain all the state spaces $S$, we must simulate all activities. When an activity is simulated, the state space changes, and all possible state spaces are discovered. $S_k$ is the state space after activity $A_k$ is performed; e.g., $S_{12}$ is the state space after activity $A_1$ and $A_2$.

When available state spaces can be derived, the same state spaces can be derived from different activities. This causes a potential redundancy problem. To avoid duplication of state spaces, we can run the following rule:

Rule 1: if $\exists S_{k}, s \in S \forall A_{f} \left( A_{f} \rightarrow S_{k} \in S \equiv A_{f} \rightarrow S_{f} \in S \right)$ then merge $(S_k, S_f)$.

Figure 2 shows Rule 1 merging $S_{12}$ into $S_2$. After the merge, the edge from $S_2$ to $S_{12}$ is connected to $S_2$.

**Phase 2: Conversion to Context Graph**

Once $S_G$ is constructed, it should be converted to $C_G$. The first step in the conversion is to find meaningful state spaces, which eventually will be declared contexts. By our observation, 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 which state spaces are meaningful, all are partitioned into $k$ clusters, in which each state space belongs to the cluster with the nearest mean. Therefore, $S_G = \{S_{i_1}, ..., S_{i_o}, ..., S_{i_o}\}$ is divided into $\{\hat{S}_{j_1}, ..., \hat{S}_{j_1}, ..., \hat{S}_{j_1}\}$, where $\hat{S}_{j}$ is a state space and $\hat{S}_{j}$ is a cluster of state spaces. Each cluster $\hat{S}_{j}$ minimizes
the sum of distances between the within-state space and the mean according to the following formula:

\[
\arg\min_{S_G} \sum_{i=1}^{k} \sum_{S_i \in \hat{S}_j} \|S_i - \mu_i\|^2,
\]

where \(S_i\) means a state space in cluster \(\hat{S}_j\). After \(S_G\) is classified into \(k\) clusters, cluster centroids are considered meaningful state spaces and are candidates for contexts.

The main procedure of conversion is as follows: declare meaningful state spaces as context, prune non-meaningful state spaces in \(S_G\), merge all activities included in every edge between contexts, and declare as scheduled activities what will be performed in a context. Note that the merged activities are executed in order.

Figure 3 illustrates the conversion from \(S_G\) to \(C_G\), which consists of 7 contexts such as breakfast and personal hygiene. Nodes highlighted in blue are meaningful state spaces and contexts. An edge in \(C_G\) describes how a context can transition to the next context in terms of activities. In \(C_1\) (see Figure 3b), for instance, two activities \(\{A_1, A_4\}\) are executed in order, and then, the next context \(C_2\) is reached. Some edges may have multiple sets of activities. For instance, the edge from \(C_2\) to \(C_5\) contains \(\{A_1, A_5, A_7\}\), \(\{A_1, A_6, A_7\}\), and \(\{A_1, A_5, A_6, A_7\}\). This illustrates that there could be multiple choices available for scheduling activities.

**Phase 3: Building Context Conditions**

To define contexts and \(C_G\) fully, context conditions must be built for each context. Context conditions allow for contexts’ evaluation and transition to the next context. We find state spaces that are meaningful and similar to the context and build proper context conditions using their status information. To find important state spaces, we need to investigate individual sensors. Each sensor’s effect is unique and has different meanings according to the context. So, we need a filter to identify relevant sensors. In order to do this, we propose an algorithm that utilizes PCA to find the sensors’ principal components (Lawrence 2005). Once we find the relevant sensors via a stochastic analysis of sensors’ high-dimensional data, the original dataset can be projected onto lower-dimensional data. This method returns less data and decreases computational complexity. The remaining data is used to build context conditions. The algorithm then repeats the whole process for each cluster to find the relevant sensors.

**How to find principal components**

Principal components are sensors that show definite variance patterns that explicitly express the change of states. We want to know in which pattern the dataset is scattered. For this, a matrix of covariances (cov) is calculated first. In a \(\xi\)-dimensional dataset, covariance cov is calculated as

\[
\text{cov} (\vec{s}_i, \vec{s}_j) = \frac{\sum_{k=1}^{\xi} (\vec{s}_{i_k} - \mu_i)(\vec{s}_{j_k} - \mu_j)}{(\xi - 1)}
\]

where \(\vec{s}_i\) and \(\vec{s}_j\) are the set of sensor values in dimensions \(i\) and \(j\), respectively; \(i\) and \(j\) are the sensor values in each dimension. The total covariances establish a \(\xi \times \xi\) covariance matrix \(R\):

\[
R (\vec{s}) = \begin{bmatrix}
\text{cov} (\vec{s}_1, \vec{s}_1) & \cdots & \text{cov} (\vec{s}_1, \vec{s}_\xi) \\
\vdots & \ddots & \vdots \\
\text{cov} (\vec{s}_\xi, \vec{s}_1) & \cdots & \text{cov} (\vec{s}_\xi, \vec{s}_\xi)
\end{bmatrix}
\]

After the covariance matrix is created, we calculate the eigenvectors, each of which can conduct linear transformations of sensor data and characterize its variance; the eigenvalues then measure how well the sensor data is scattered. We choose the eigenvectors that show the most variant spread of data as principal components. If data is evenly scattered with an axis transformed by an eigenvector (i.e., the data pattern is recognized explicitly), it is an important eigenvector, which means it’s the desired principal component. However, if data is narrowly spread around the axis and doesn’t show any clear pattern, the corresponding eigenvector should not be considered in building context conditions.

Eigenvectors that are evenly scattered have high eigenvalues. Eigenvalues determine how the corresponding eigenvectors scatter data—higher eigenvalues spread more...


The complexity of constructing $S_G$ depends on the number of rooms and activities in the space. When there are $\zeta$ number of activity and $h$ areas, there are $\zeta h$ nodes. Then, $h\zeta^2$ edges are inserted onto the graph. Therefore, the complexity is

$$\text{complexity} = O(\zeta^2 h + \zeta h).$$

In our research, we followed the activity models of Lawton and Brody (1969), which identify activities of daily living (ADLs) and instrumental activites of daily living (IADLs). ADLs are basic, physical activities such as feeding, dressing, bathing, walking, toileting, and grooming. IADLs include social behaviors such as shopping, communicating, transporting, transacting, taking medications, housekeeping, and doing laundry. Lawton and Brody provided broad categories of activities with 30 ADL activities and 31 IADL activities. Therefore, the maximum $\zeta$ is 61.

In converting $S_G$ to $C_G$, complexity will mostly depend on $k$-means clustering. The complexity of conversion to $C_G$ is calculated by the number of state spaces ($t$), the number of sensors ($\xi$), and the number of clusters ($k$):

$$\text{complexity} = O(\xi^{(k+1)} \cdot \log \xi h).$$

The complexity involved in building context conditions mostly depends on the complexity of the PCA. In the previous section, we described PCA algorithms for only one context. For the complete configuration of the simulation, we must perform PCA per context. With $k$ as the number of contexts, and $\tau$ as the number of principal components, complexity can be expressed as follows:

$$\text{complexity} = O(\tau^2 \xi^k \xi^3 k).$$

All variables are bounded during scenario design. However, the number of activities and contexts increases as we seek higher accuracy in the simulation, which increases complexity. Therefore, there is a trade-off between complexity and accuracy.

**Experimental Validation**

To evaluate the performance of the proposed approach, we conducted a few experiments. We focused on evaluating the fidelity of $C_G$ derived from activities, which produces all possible schedules of activities that are feasible in the real world. Therefore, our validation goal was to compare the generated schedules with real-world data. We collected schedule datasets for 10 days and used them for validation. (The scenario description and datasets are available online.) However, it is hard to find comparable generated schedules of activities because there is no efficient way of filtering the large number of schedules to find those that are representative of the data we’re studying. Instead of searching for such schedules, we therefore use statistic analysis of the schedules to validate $C_G$ feasibility.

For this purpose, we built statistic models of schedules that apply a Bayesian network. First, we created the Bayesian network from schedules of activities and populate conditional probability table (CPT), which consists of occurrence probabilities for a pair of activities.

We calculate the joint probability distribution on the entire schedule scenario after the CPT has been applied to all possible pairs of activities. If the occurrence probability is not high enough, it says that the scenario did not occur, even if the scenario actually happened. Thus, the probability must be high enough to correct this contradiction. For validation, we created and compared two CPTs, one from actual schedules and one from $C_G$, shown in Figure 5.

If they have a high degree of similarity, then the CPT based on $C_G$ can be considered identical to the actual, and we can say that $C_G$ has high actuality. Note that the CPT based on $C_G$ is enhanced because $C_G$ contains all possible schedules, which makes each conditional probability lower.

---

1 [http://www.icta.ufl.edu/persim/datasets/](http://www.icta.ufl.edu/persim/datasets/)
than those in the CPT based on the actual schedules. For comparison, we applied actual schedules to each CPT and found the similarity of each occurrence probability (P and $P'$). We also created three different iterations of $CG$ by changing the number of contexts (k). Figure 6 shows the occurrence probabilities of each schedule.

![Context Graph](image)

**Figure 5. Validation based on Bayesian Network.**

![Occurrence Probability Chart](image)

**Figure 6. Occurrence probability of actual scenarios based on the derived context graph. The probability when k = 5 shows the best performance.**

The actual schedules show very low probability on days 3, 4, and 8, because the schedules contain certain activities which rarely happen. The generated schedules similarly follow this pattern. When k = 4, it has a slightly different pattern, which is why day 3 has a higher probability than day 2, as well as a lower overall probability. When k = 6, most of the probabilities are too low and minor variances indicate too great of an effect. When k = 5, all probabilities are relatively high, and $CG$ also follows the pattern of actual schedules. Probabilities on days 3 and 4 show different patterns than those of actual schedules, but this could be because there were too few actual schedules available when the CPT was created. On the other hand, $CG$ is a complete model that includes all scheduled scenarios. Hence, we suggest that users modify $CG$ as derived through our approach to meet their own needs and specifications.

**Conclusion**

A context, or a high-level simulation entity, enables scalable simulation with high realism. However, it is hard for users to configure such high-level conceptual models. The user is required to specify a lot of information and many parameters about contexts, which could burden and decrease simulation performance. This shows a need for more user-friendly initiated models.

In this paper, we proposed an approach to derive contexts and context graphs from given activities without additional specifications. Derivation of contexts from activities can reduce user efforts and make simulation easy. Considering the performance of our approach, we can conclude that it is a good alternative to manual specifications of activity scenarios and simulation configurations.

**References**


