2011 Seventh International Conference on Intelligent Environments

Persim – Simulator for Human Activities in Pervasive Spaces

Sumi Helal, Jae Woong Lee,
Shantonu Hossain and Eunju Kim
University of Florida,
Gainesville, FL 32611 USA
helal@cise.ufl.edu, jwlee@cise.ufl.edu,
shantonu_hossain@yahoo.com, ejkim@cise.ufl.edu

Hani Hagras
University of Essex,
Colchester CO4, UK
hani@essex.ac.uk

Diane Cook
Washington State University,
Pullman, WA 99164 USA
cook@eecs.wsu.edu

***Abstract*— Activity recognition research relies heavily on test data to verify the modeling technique and the performance of the activity recognition algorithm. But data from real deployments are expensive and time consuming to obtain. And even if cost is not an issue, regulatory limitations on the use of human subjects prohibit the collection of extensive datasets that can test all scenarios, under all circumstances. A powerful and verifiable simulation tool is needed to accelerate research on human activity recognition. We present Persim, an event driven simulator of human activities in pervasive spaces. Persim is capable of capturing elements of space, sensors, behaviors (activities), and their inter-relationships. We focus on presenting the five main use cases for Persim addressing dataset synthesis, reuse and extension of existing datasets, sharing of data and simulation projects, as well as data validation.**

Keywords: Pervasive system simulation, sensory dataset synthesis, human activity simulation, activity recognition algorithms, standard sensory dataset representation, simulation model.

I. INTRODUCTION

Access to meaningful collections of sensory data is one of the major impediments in human activity recognition research. Researchers often need data to evaluate the viability of their models and algorithms. But useful sensory data from real world deployments of pervasive spaces are very scarce. This is due to the significant cost and elaborate groundwork needed to create actual spaces. Additionally, human subjects are not easy to find and recruit. Even in real deployments, human subjects cannot be used extensively to test all scenarios and verify multitudes of theories. Rather, human subjects are used to validate the most basic aspects of the pervasive space and its applications, leaving many questions unanswered and theories unverified.

It is thus necessary to develop alternative and practical approaches to studying human activity recognition. Powerful and realistic simulation tools could be used to support the growing demand for test data. Simulations enable researchers to create focused synthetic replications of important events and activities under study. It can be easily changed and refined allowing researchers efficiently to experiment, analyze and fine-tune their models and associated algorithms. Simulation also allows a wider community of researchers to engage and collaborate to solve a specific problem. Hence, a design based on preliminary simulation studies would most likely to be a more robust and inclusive design. Also, a simulation model that mimics an existing real world pervasive space is most likely to answer more questions (and generate much more data) than the target actual space. This early stage simulation can help researchers evaluate their ideas and algorithms quickly, cost-effectively, and with reasonable accuracy.

In this paper, we present Persim – a simulator for human activities in pervasive spaces. Persim is a multi-scale, event-driven simulator used primarily to synthesize datasets for human activities in standardized format. Depending on the goals of the study, Persim can be configured and used to capture micro- or macro-scale events and artifacts.

Persim is based on a set of Sensory Dataset Description Language specifications (SDDL) [1][2][14] intended for effective sharing of existing datasets among researchers. SDDL, initiated by the University of Florida is intended as a community based effort. It provides a tool to convert existing datasets to the SDDL format.

By design, Persim promotes for sharing of datasets and simulation studies and their generated data, among research groups working on activity recognition and smart environments. It uses a community repository to store data and simulation projects. The repository and Persim itself are accessible from the Persim project web site [2].

In addition to presenting Persim and its architecture simulation loop, we focus in this paper on the five main use cases of Persim summarized below.

- Converting dataset into an SDDL format
- Simulating a smart environments including its activities and generating a synthesized dataset
- Sharing of datasets and Persim projects using a community resource repository
- Extending existing datasets by modifying the smart environments in which they were collected or generated. We refer to this as "stem-celling" a dataset, because the resulting dataset generated by Persim carries much of the realistic nature of the original dataset, even though by extension the new dataset is different.
- Validating a generated dataset against a real or a reference dataset.

The rest of this paper is organized as follows. Section II reviews related research on simulation and human activity modeling. In section III, Persim and its architecture and

This work was supported in part by an NIH Grant number 5R21DA024294.

DOI 10.1109/IE.2011.34

simulation algorithm are presented. Section IV presents the five use cases of Persim. An approach to validate a simulated dataset is introduced and an illustrative analysis of the dataset is presented section V. Discussion of work in progress and future work is presented in Section VI. Finally, conclusions are presented in section VI.

II. Related work

In the field of pervasive computing and wireless sensor network (WSN), several simulation concepts and tools have been explored and introduced, each with its own focus and goal. SENSORIA [9], is a simulator focusing on traffic generation, energy consumption and inherent protocols of WSN. In [6], a detailed simulation model was presented which also focuses on accurate model for battery, processor power consumption and in network traffic. In [8], Discrete-Event system Specification (DEVS) was proposed to define asynchronous discrete-events occurring in WSN. Eventually, routing and communication becomes non-trivial factors for them. Unlike Persim, none of the above simulation models focus on human activities performed in a ubiquitous computing environment.

On the other hand, some approaches such as [10][11] simulate a specific system at a very low level. For example, TOSSIM [11] simulates the TinyOS operating system in action under some workload (application). However, it is impractical to use TOSSIM to simulate what Persim can simulate easily without making any assumptions about the underlying platform.

The DiaSim [7] simulator executes pervasive computing applications by creating an emulation layer and developing simulation logic using a programming framework. Under DiaSim, a pervasive computing environment is described in terms of stimulus procedures (any change in the environment that can be consumed by sensors) and simulated services (sensors and actuators) in a specification language called DiaSpec. It also generates simulation data in raw format. Though Persim's definition of pervasive computing space is similar to DiaSim, the objective of the two simulators is different. Persim aims to simulate human activities performed in pervasive spaces and generate corresponding sensor data in a standardized format for further analysis, whereas DiaSim simulates applications such as fire situations, intrusions, etc. to identify potential conflicts. The validation of DiaSim is performed only from the application perspective, ensuring the pervasive application does what it is intended to do. On the other hand, Persim verifies itself as discussed in this paper via fuzzy based verification to measure the level of fidelity between simulated system and its real counterpart in terms of their corresponding datasets. It also generates simulated data in SDDL format [1][2][14], which makes it suitable for collaborative research.

Studies about modeling human activity from sensor data and simulating such models have been carried out using different approaches and aiming at different goals. In [19], activities of nurses caring for patients in a hospital setting have been studied with the goal of maximizing scheduling efficiency and minimizing down-time to patients. In such simulations, computation was intensive, and classification based reduction techniques were used to improve simulation scalability. In [20], a simulation tool was used to model movement of users in outdoor environments. The goal of the tool was to study how best to route ad-hoc traffic based on user mobility. The tool is useful but has limited applicability in indoor environments where multiple activities such as eating and cleaning are of interest. In [21], elements such as "fact" and "belief," are proposed as 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. Although this approach is useful for dynamic and dangerous contexts such as offloading on a lunar surface, it may impose unnecessary complexity and require excessive computation in simulating simple activities of daily life such as having breakfast and watching TV.

III. Overview of Persim

In this section we present the Persim architecture and its main simulation model and algorithm.

A. What is Persim?

Persim is an event-driven, human activity simulator capable of capturing the physical elements of a space including its sensors and human activities. Persim users typically build a simulation "project" over multiple sessions before they are ready to generate data or make a multitude of changes to the sensory elements, the activities or even the structure of the output dataset. Data generated by Persim follows the Sensory Dataset Description Language (SDDL) proposed standard [1][2][14]. There are five essential steps to creating a simulation project.

- First, a space and its various areas must be defined. Currently, we have rudimentary graphical/spatial space definitions, but work is well underway to create a higher level of realism through the use of space templates (e.g., single family home, apartment, assisted living facility, kindergarten, etc).
- In the second step, different types of sensors and actuators are added into the various areas of a space. Each sensor is defined in terms of type, function, sensor event generator, domain value generator, among other important information.
- In the third step, activities to be performed (e.g. turn on oven, sit on chair, walk from area A to B etc.) are expressed.
- In the fourth step, *activities-to-sensors mapping* is prescribed by the user. The mapping specifies which sensors are relevant to the detection of each activity. The user may repeatedly perform the four steps to make changes or to fine-tune the simulation project to her/his satisfaction. Persim invalidates any step that becomes inconsistent with earlier steps that have been updated.

- In the fifth and final step, simulation of the finalized simulation project is started and the corresponding dataset is generated.

Based on simulation results, the user may go back to the project steps again and again to make any necessary modifications.

Persim promotes sharing of efforts by allowing one user to start a simulation project by "uploading" a dataset originally generated by another user (through Persim), modifying the design and fine-tuning the experiments to achieve a specific research-goal. Such sharing is made possible by using SDDL. Persim also allows a user to convert a real world deployment data into SDDL and use it to generate the corresponding simulation model which can then be extended far beyond the original actual deployment. This allows for a greater level of sharing. Further, it allows the same researcher and the owner of a real dataset to continue work on a past project, albeit in a virtual simulation mode.

Thus the simulator is intended to open a new dimension of collaborative and extensible research in the area of human activity recognition and other pervasive computing applications.

B. Architecture

The architecture of Persim is shown in Fig. 1. Users interact with Persim through a web based control panel (GUI) to build and maintain a simulation project. The component builder assists the user in configuring all components that enter into a simulation project (space, sensors, events, and activities). An event-driven simulation engine uploads Persim projects, runs the simulation and generates output through an SDDL generator. A text to SDDL conversion tool is used by Persim to enable the conversion of existing datasets to SDDL format. An SDDL parser is also used to parse an existing SDDL dataset and capture through its metadata the project definition that could be later updated and/or simulated.

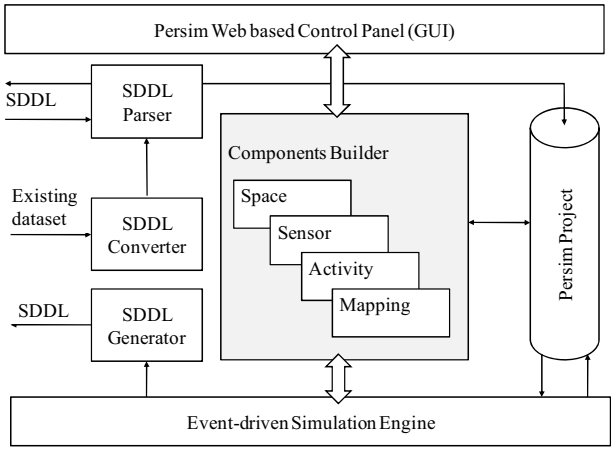

Fig. 1. Persim architecture.

We now examine the major components of a Persim simulation project in greater details.

Space: This component captures the spatial aspects of the target simulation. As mentioned earlier, it is currently in a primitive stage, capturing only space location ID and space name, giving no details as to layouts, walkable and non-walkable areas, and to the precise relative locations among objects and sensors. Work is underway to further develop this component and to offer editable templates of common spaces.

Sensor: Sensor component denotes the physical sensors deployed in the space areas. It maintains information for each sensor including the following attributes: (a) Name, (b) Id, (c) Location Id, (d) Type, (e) Functional Type, (f) Minimum Value, (g) Maximum Value, (h) Initial Value, (i) Value Generation Function, and (j) Distribution Parameters. Persim supports two types of sensors: independent and dependent.

Independent Sensor: is a sensor whose output is not triggered by any external activity or behavior. Temperature sensor is an example of an independent sensor since it generates temperature values regardless of any human activities around it. To simulate independent sensor, Persim requires the following additional attributes related to the process of generating sensor values (not the generation of values themselves): (a) Process Generation Type, (b) Interval Size, (c) Process Generation Function and (d) Distribution Parameters.

Dependent Sensor: the output of this type of sensor is triggered by specific external activities or behaviors. For example, pressure sensor is a dependent sensor since it only generates output when an external pressure is applied on the sensor.

Activity: Is an independent event that captures a human behavior (e.g., walking to kitchen, getting up, leaving house, etc.) Activities are generally sensed by several sensors and are characterized by: (a) Activity Name, (b) Activity Type, (c) Interval Type, (d) Interval Size, (e) Inter-arrival Function and (f) Distribution Parameters.

Activity-Sensor-Mapping: Recognizing an activity usually depends on one or more sensors. In order to capture this dependency in the simulation, an Activity-Sensor mapping must be specified for each activity. For each sensor-activity mapping there are three attributes: *min*, *max* and *seq*. *Min* and *max* define the range of the sensor values required for the detection of the activity. *Seq* (sequence number for sensor trigger) assigns priorities to the various mappings to set an order for the effect of a given sensor on the various activities. This *seq* number is utilized in the simulation loop and provides additional control for the researcher over how various activities are recognized by overlapping sets of sensors. A don't care value is allowed ("#") and is used when appropriate. In this case the simulation loop chooses randomly. Sensors of identical *seq* numbers are triggered simultaneously (at same simulation clock value).

Persim Configuration: Persim manages all configuration parameters needed by the event-driven simulation engine. These are (a) Simulation Start Time, (b) Simulation End Time, (c) Simulation Mode, and, (d) Activities to be simulated.

Persim supports two simulation modes – *Activity-driven* and *State-space*. In the *Activity-driven* mode, the user specifies the set of activities that the simulation should focus on. This mode activates the set of dependent sensors for all desired activities, based on the *Activity-Sensor* mapping. In

State-space mode, Persim generates a time-stamped sequence of events reporting on all sensors (dependent and independent) deployed in the space irrespective of any activity. In this mode, the simulator records the state of the entire space based on time-driven events and/or activity-driven events.

C. Implementation

We have adopted the discrete-event simulation model [5] using a next-event, time-advance approach to capture the dynamic nature of the pervasive space. In this simulation model, our target space state changes whenever any event (activity-driven or time-driven) occurs and the system variables of the space are updated, based on the simulation logic. The simulation model consists of the following major variables:

Simulation Clock (simClk): A variable that keeps track of the current value of the logical simulated time. The domain of the clock value is the set of timestamps of all events. This is because simClk is always advanced only to the time of the next soonest event.

List of Events (eventList(e, t)): A sequence of tuples of the form (e, t) where e denotes the event and t denotes the time of the occurrence of that event. Each tuple represents an event trigger, which needs to be processed by the simulator.

List of Activities (activityList): A list of activities defined by the user for the specific simulation project.

List of contexts (contextList): A list of independent sensors that act as "contexts" for the simulation.

Initialization Routine: A method to initialize simClk and eventList before the simulation loop begins.

Timing Routine: A method that determines the next event from the eventList and advances the simClk to the time of occurrence of the selected event.

Event Routine: A method to schedule the next event to occur. It also determines the time of occurrence of the event based on the library routine.

Library Routine: A method to generate random variables drawn from the probability distribution of the inter-arrival time of an event and generation function of sensors (as specified by the user and configured by Persim).

Simulation Loop: The main program that invokes the initialization routine, timing routine and event routine and that propels the simulation of the target space by processing and generating/scheduling new events for the entire lifecycle of the simulation.

Pseudo code of the main simulation loop is shown in Fig. 2. The simulation starts from the empty and idle state. Then it will loop in finding the most imminent (next) event from the *eventList*. It'll then advance the *simClk* to the time of occurrence of the next event. Then it will process the event according to the type of the simulation, whether it is activity-driven or state-space. The simulation will continue until the *simClk* exceeds the designated simulation time at which point the simulation loop exits. Finally, simulated data in SDDL format is generated.

```
function simulationLoop
1. simClk← 0
2. eventList←initialize events from activityList
3. while(simClk<= simulationEnd)
4.   currentEvent← find the most imminent event from  eventList
5.   simClk←currentEvent.eventTime
6.  if (simMode == activity_driven)
7.    processActivityDrivenEvent
8.      Determine dependent sensors from activityMapper
9.      Find next sensor to be triggered based on sensor seq#
10.     Get sensor output from valueGenerationFunction
11. else    //simMode = state_space
12.   processStateSpaceEvent
13.     Find independent sensors from contextList
14.     Get sensor output from valueGenerationFunction
15.     if (currentEvent.type == activity_driven)
16.       processActivityDrivenEvent
17.         Perform same steps as 9-11
18.     endif
19.   generateEvent
20.     Get newEventTime for currentEvent from the inter- arrival distribution
21.   Schedule next event newEvent based on  currentEvent.endTime
22.   Add newEvent to the eventList
23.  endif
24. endwhile
25. generate SDDL file as output
```

Fig. 2. Pseudo code of Persim simulation algorithm.

Persim utilizes a web-based interface, which is platform independent and universally accessible. The Persim interface is developed in Java, JavaScript, and Java Server Pages (JSP), and the application is hosted by an Apache Tomcat server. In Persim, simulation information and standardized output data are generated as XML files. The Persim simulator will be made available for download from the Persim project web site [12]. The Persim download package includes: (a) a war file, which can be run within Apache Tomcat, (b) user manual, and (c) sample projects and SDDL files.

IV. PERSIM USE CASES

In this section, we discuss the various use cases of Persim utilizing real life scenarios. Fig. 3. shows an overview of the different ways in which Persim can be used. The main use scenario is as a simulation design tool and a synthesized data generator (the B cloud). In this role, Persim enables researchers to create a simulation environment and simulate complex human activities using its web-based interface. Persim can also be used as a support tool to convert an arbitrary formatted dataset (e.g., data collected from an actual space) to an SDDL counterpart using an SDDL converter (the A cloud). Converting an arbitrary dataset to SDDL achieves a little more than just adhering to standards and enhancing the sharing of the dataset itself. The conversion allows Persim to glean a partial simulation model corresponding to a real dataset converted into SDDL format. By extracting a simulation model from actual data, Persim empowers its users (be they the original owners of the dataset or other researchers) to virtually recreate the smart environment in which data was collected, and allows them to extend data collection and experimentation work previously done in a real deployment (the D cloud). This is a significant and powerful feature of Persim, which we refer to as dataset "*stem-celling.*" In addition to creating projects from actual datasets, Persim enables additional sharing – utilizing a repository- by allowing one simulation project designed by one researcher to

be picked up and extended by another (the C cloud). Finally, Persim can be used as a validation tool capable of contrasting and measuring the similarity of two datasets (the E cloud). This can be used to compare a synthesized dataset with an actual one. This validation is also helpful in calibrating Persim itself in terms of realism, and in incremental testing by the researcher during the construction of a simulation project.

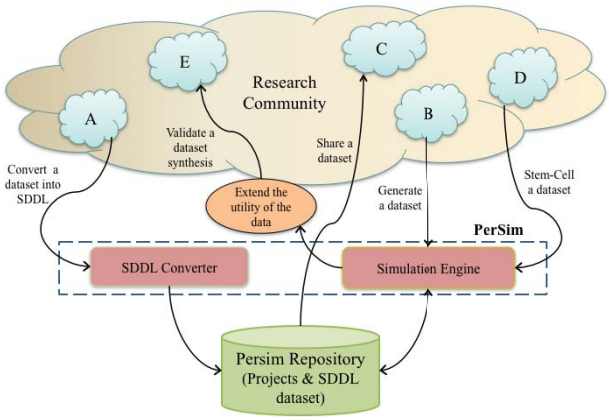

Fig. 3. High level view of intended uses of Persim.

To illustrate all five use cases of Persim in more details, we use a simple scenario from an activity dataset contributed by the University of Amsterdam [23], which records activities of daily living performed by a 26-year-old man living in a three-bedroom apartment for 28 days. The scenario consists of two activities: preparing dinner and getting a drink. The man (we call him Charlie) prepares dinner in the evening and then gets a drink before eating what he has prepared. The activities are detected by 9 sensors deployed in the kitchen. They are attached to the microwave, freezer, cupboard for cups, pan, fridge, cupboard for plates, cupboard for groceries, dishwasher, and hall-bathroom door. Even though the sensors detect his activities by sensing various methods (e.g., sensing motion, sensing sound, and so on), detection by any sensors infers that there is a movement involved with an object (e.g., passing by the fridge, or turning on the microwave). Therefore, for simplicity, we assume that the sensors deployed in the space are all motion sensors.

A. Converting dataset into an SDDL format

Persim offers SDDL conversion as a useful tool to convert non SDDL datasets to XML encoded SDDL format. It is accessible from Persim's web-based interface and utilizes a wizard type interface gadget. The conversion process consists of two steps. In the first step, the user has to specify the meta-data of the dataset such as layout of the space from where data was collected, types of sensors deployed, performed activities, date and duration of data collection. This step is very important since meta-data plays a significant role in characterizing the dataset. After the meta-data is specified, the user points the tool to the actual data file. Then, the user can click 'Generate SDDL' button, which will store the whole dataset along with the meta-data in SDDL format in the root directory where Persim is installed. Subsequently, the user can optionally edit the SDDL file for any annotation needs.

The SDDL converter can be utilized as an auxiliary tool to accelerate the sharing and reuse of datasets by a community of users. For an example, the BoxLab [22] project is an example of a funded project aiming to enable sharing among researchers working on smart environments. Our SDDL tool can be very helpful for the BoxLab repository, as it is a key enabler to our own repository described in section IV.C.

B. Generating a synthetic SDDL dataset

To generate a dataset, a Persim project must be constructed (as per the five steps described in section III.A) or a shared project is used. To start simulation and generate data, the user has the choice to simulate the entire state space generating a dataset that reports on the state of all sensors and actuators at fixed or variable time intervals. This type of simulation, which we refer to as "state space simulation" is similar in structure to the iDorm dataset contributed by the University of Essex [4]. The user may instead decides to zoom into the space and simulate only a selected set of activities of interest, generating a pre-labeled dataset in which only events and state changes related to the activity or activities are reported as they happen (in variable time intervals). This type of simulation, which we refer to as "activity-focused simulation" is similar to the WSU Smart Apartment dataset [3] and the CARE project data [23] contributed by the University of Amsterdam.

In activity-focused simulation, users may structure activities into sub-activities or tasks. For example, the activity "preparing dinner" in our example scenario can be split into three smaller actions: taking out groceries, cooking on a pan, and microwaving food. This structuring capability helps achieve more accurate simulation, because each sub-activity can be configured differently (e.g., different intervals and distribution of event arrival in each).

A generated dataset is saved in an SDDL file which contains sensor and event data as well as other information such as the pervasive space parameters, defined activities, non-structured (text) descriptions of the dataset, its owner, purpose, history, etc.

C. Sharing a dataset

One goal of Persim is to facilitate the collaboration and sharing of ideas, efforts and resources across the smart environments research groups. Community resources such as the BoxLab [22] collect datasets and make them available as annotated archrivals. Persim features a similar community resource repository [24], but unlike BoxLab, it requires all datasets to be in SDDL format. Additionally, the Persim repository stores real and simulated datasets. It stores generated datasets along with their corresponding Persim simulation projects. Real or generated datasets may be shared without having to use or learn Persim. However, a simulation project may also be shared (along with its generated data), which will require the use of Persim. Fig. 4 describes the range of resources maintained by the repository.

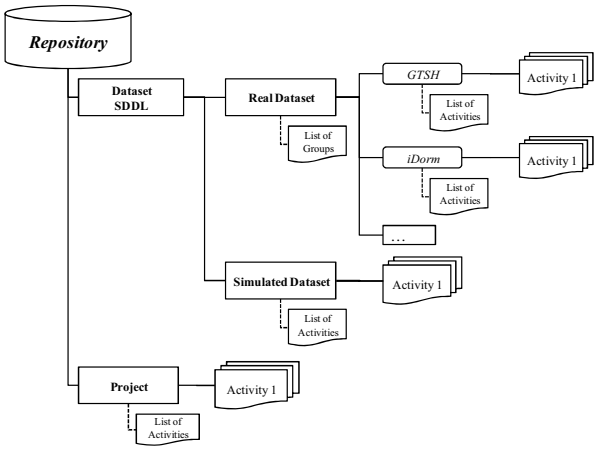

Fig. 4. Scheme of Persim repository.

D. Stem-celling a real dataset

Persim allows for a powerful feature in which a simulation project can be gleaned out of a real dataset (in SDDL format). The resulting project as demonstrated in is a partially specified one, requiring the user to complete its remaining specifications. Even though the gleaned project file may be missing important environmental information for simulation, it does carry important and realistic data values that empower Persim in selecting the project model parameters (e.g., sensor value generator and event interval generator distributions).

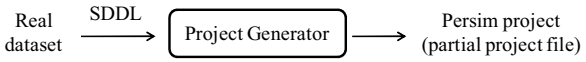

Fig. 5 Gleaning a Persim project out of a real dataset.

Suppose that a user stem-cells a real dataset section describing drinking activity from the Amsterdam dataset. The project generator in Persim recognizes that Charlie drinks in the kitchen when nine sensors deployed in the kitchen area detect his motions. Then it analyzes sensory data and figures out the mapping between activities and sensors. For example, if there is event data with a value from a sensor attached to a cup, the sensor on the cup is mapped to the activity of drinking.

Parameters such as generation functions for sensory values and event intervals are calculated mathematically. Because sensory data is well defined within labels of activities in SDDL, Persim can calculate an average of a sensory value per activity and set it as a parameter. For the interval generation function, we utilize an arithmetic average and standard deviation by using (1) and (2), respectively, and then set them as initial values of exponential distribution function. Additionally, we initialize a generation function to generate sensor values using a uniform distribution function because we are often interested in values generated within some range (the sensing or detection range). For the range, we consider maximum and minimum of sensor values.

$$Int(avg)A = \sum_{i \in A} \frac{U(i+1) - U(i)}{numberof(events)} \quad (1)$$

$$Int(dev)A = \sum_{i \in A} \sqrt{\frac{(U(i) - Int(avg)A)^2}{numberof(events)}} \quad (2)$$

(A: activity, $U(i)$: time at event i)

As mentioned, a Persim project stem-celled from an actual dataset is a partial and tentative project, whose parameters should be inspected and modified as needed by the user.

E. Validating a data synthesis

The fifth use case of Persim is dataset validation. Often it is needed to ensure that a synthesized or shared/extended dataset is useful and representative of reality. Validating a dataset against another "reference" dataset will therefore be required. The main mechanism for such validation is finding a measure of distance between the two datasets (the one being validated and the reference set), and verifying that the distance is within acceptable margins.

Since we had to validate Persim itself to verify its accuracy and usefulness [13], we leveraged the validation code and packaged it into Persim's tool set to allow users to validate synthesized data.

V. Validation of Dataset Simulated by Persim

We validate a simulated dataset by computing its similarity with a real dataset. We selected the Amsterdam dataset as the real dataset for demonstration. We selected 12 days, covering only "preparing dinner" and "drinking water" activities. The order of occurrence of sensor events and their frequencies are different each day. For example, it is unnecessary to turn on a dishwasher every time Charlie prepares dinner, whereas he may open a fridge many times a day. Therefore we collect data to cover the various cases. The duration of activities in each day is no more than 40 minutes. TABLE I shows the information of the real dataset such as start time and end time. Persim simulated the two activities using this information.

Both micro and macro level similarity were computed. First, for micro level similarity analysis, we examined the type and temporal position similarity of every individual sensor event data. For example, for sensor event s_i as in (3), we determined the most similar real sensor event r_j and their type and position similarity. Second, we examined the features of the two datasets rather than individual sensor event. To illustrate, we compared the number of sensor events for each sensor type in each dataset. If two datasets are similar, their sensor types and number of sensor events for each sensor type will also be similar.

To compute the position similarity, we used Fuzzy Logic. In Fuzzy Logic, a fuzzy set A of the Input I is defined by its membership function (denoted also by A) $A: I \rightarrow [0,1]$ [25]. In our analysis, there are two datasets R and S. R is a real dataset and S is a simulated dataset. We regard S as an input I and computed the membership of S corresponding to R.

$$R = \{r_1, r_2, r_3, \ldots, r_n\}$$
$$S = \{s_1, s_2, s_3, \ldots, s_m\} \quad (3)$$

We use the notation $A_{ij}(s_i) = (r_j, \alpha, \beta)$ to define membership of s_i to r_j where, α and β define the left and right maximum possible distance from r_j respectively. In our analysis, they are calculated using start time and end time of each dataset. We found the fuzzy set for sensor type (A_{ij}^{type}) and sensor position ($A_{ij}^{position}$) using fuzzy membership functions in (4) [25][26]. We then computed the membership of s_i using min-max fuzzy inference rule as in (5) [26].

$$A_{ij}(s_i) = \begin{cases} 1 - \frac{r_j - s_i}{\alpha} & \text{if } r_j - \alpha \le s_i \le r_j \\ 1 - \frac{s_i - r_j}{\beta} & \text{if } r_j \le s_i \le \beta \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$A_{ij} = \min\left(A_{ij}^{type}, A_{ij}^{position}\right) \quad (5)$$

$$A_i = \max\left(A_{i1}, A_{i2}, A_{i3}, ..., A_{in}\right)$$

To compute similarity *A(S),* we used hamming distance, which is commonly used as distance measure between fuzzy sets *A* and *B* as shown in (6) [25]. In our analysis, we assign 1 to the fuzzy set *B* of a real dataset *R* because we assume that the membership of real dataset is always *true*.

$$d(A, B) = \frac{1}{m}\sum_{i=1}^{m} |A_i - B_i| \quad (6)$$

$$Similarity(A, B) = 2 - d((A \cap B),[1]) - d((A \cup B),[0])$$

We used a similar method for computing the similarity of the number of events for each sensor type. First, we count the number of event for each sensor type of both real dataset and simulated dataset. Then we compute similarity using (4). For this analysis, α and β are the maximum number of events. The analysis result shows 78.5% similarity in average for individual event position. The average similarity of the number of sensor events is 74.9%. Total average of both similarities is 76.7%. TABLE I and Fig. show the similarity between simulated datasets and real datasets for each day. In TABLE I, when both activities were performed, the similarity is higher than either one of them were performed. When single activity like "Drink" is performed, the number of involved sensor and the frequency of sensor events are small. In the case, even small difference of values results in big similarity difference. It means Persim performs better when there are frequent sensor events with more sensors.

TABLE I
SIMILARITY OF SIMULATED DATASET

Data Set	Preparing Dinner	Drinking	Comparison of Similarity	
	Start Time End Time	Start Time End Time	Event Position	Event Number
1	19:40:26 20:22:58	20:13:12 20:23:35	0.82	0.81
2	No activity	21:39:23 21:40:12	0.77	0.67
3	18:59:55 19:19:08	19:19:52 19:20:11	0.74	0.82
4	19:44:50 21:05:13	No activity	0.80	0.76
5	20:57:26 21:09:27	No activity	0.69	0.58
6	No activity	23:52:12 23:52:48	0.79	0.83
7	19:51:29 20:27:00	No activity	0.77	0.63
8	18:54:04 19:38:33	19:40:03 19:40:26	0.81	0.83
9	19:33:36 20:10:42	20:10:53 20:11:24	0.91	0.68
10	17:58:06 18:12:51	18:16:03 18:16:30	0.80	0.93
11	18:15:12 18:39:46	18:55:11 18:55:46	0.85	0.77
12	19:43:51 20:15:37	No activity	0.67	0.68
AVG			0.785	0.749

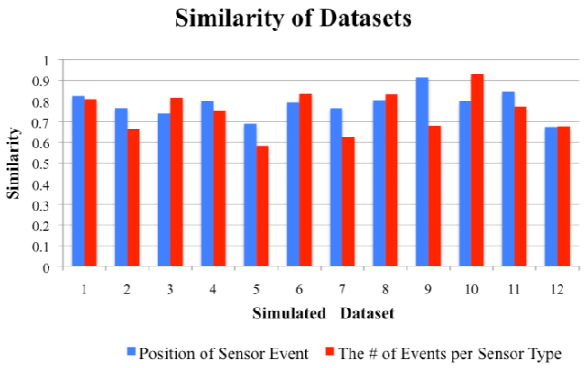

Fig. 6. Similarity of between Simulated Datasets and Real Datasets.

VI. DISCUSSION

In this section, we discuss a few issues in Persim, which are subject of work in progress or future research. First, Persim may bring some questions about how actual activity is simulated using the current user interface. Persim assumes that entities involved in activities are well-defined; for example, sensors are assumed to be deployed virtually at the optimal places in a designed pervasive space. Under this assumption, the Persim user is neither concerned nor able to control sensor placement. To address this limitation we are working jointly with Dongguk University, Korea, on a high fidelity 3D realization of designed space, with high-precision grid that allows for full control and deliberate placement of sensors. We will also employ a programming by demonstration (PbD) technique [28] within the 3D user interface, which will offer a totally different approach to activity specification.

Another limitation of the current version of Persim is that its domain of target activities is limited to indoor spaces (e.g. homes), leaving many important outdoor spaces unaddressed. As the need for recognition of large-scale outdoor activities and collecting their dataset grows, Persim will need to add simulation capabilities to cover outdoor environments and simulation. Mobility recognition [27] is one example of a

dataset collected in a city street using GPS. Unlike indoor activities, outdoor activities need large space and dense deployment of sensors. Also function generation for indoor activities might not work well for outdoor activities. Addressing simulation for outdoors is planned as future development of Persim.

VII. CONCLUSION

In this paper we presented Persim - a human activity simulator capable of capturing physical elements of a space including sensors and the behavior of the residents (activities). Persim can be used to assist in studying human activities and evaluating the accuracy of existing or newly proposed algorithms such as activity recognition. It can also examine the suitability of a tentative sensor set used to recognize the activities. It empowers the researcher to design a space and modify the design in an incremental fashion over multiple persistent sessions, and to simulate/change the space repeatedly until it satisfies the requirements or the research goals. It also allows the owner of a dataset to go back in time and explore slight variations in the pervasive system without actually repeating the experiments and avoiding huge time to collect data. We presented the Persim architecture and its simulation algorithm and described its five main use cases. Persim helps not only in modeling and generating synthesized datasets, but also in sharing them with the research community.

REFERENCES

[1] A. Helal, A. Mendez-Vazquez, S. Hossain, "Specification and synthesis of sensory datasets in pervasive spaces," IEEE Symposium on Computers and Communications, Sousse, Tunisia, July 5-8, 2009.

[2] *Persim: A Simulator for Pervasive Spaces* - project official web site: http://www.icta.ufl.edu/projects_persim.htm

[3] D. J. Cook and M. Schmitter-Edgecombe, "Activity profiling using pervasive sensing in smart homes," *IEEE Transactions on Information Technology for Biomedicine*, 2008.

[4] V. Callaghan, J. Woods, S. Fitz, T. Dennis, H. Hagras, M. Colley, I. Henning, "The Essex iDorm: A testbed for exploringintelligent energy usage technologies in the home," Proceedings of the International Conference on Intelligent Green and Energy Efficient Building and Technologies, Beijing, China, April 2008.

[5] A. M. Law, W. D. Kelton, Simulation Modeling and Analysis, 2nd edition, pp. 1-103.

[6] M. Varshney and R. Bagrodia, "Detailed models for sensor network simulations and their impact on network performance," Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, ACM, pp. 70-77. October 2004.

[7] W. Jouve, J. Bruneau and C. Consel, "DiaSim: A parameterized simulator for pervasive computing applications," Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications - Volume 00, Computer Society, pp. 1-3. IEEE March 2009.

[8] T. Antoine-Santoni, J. F. Santucci, E. De Gentili and B. Costa, "Modelling &simulation oriented components of wireless sensor network using DEVS formalism," Proceedings of the 2007 spring simulation multiconference - Volume 2, pp. 299-306. March 2007.

[9] J. N. Al-Karaki and G. A. Al-Mashaqbeh, "SENSORIA: A new simulation platform for wireless sensor networks," Proceedings of the 2007 International Conference on Sensor Technologies and Applications, IEEE Computer Society, pp. 424-429, October 2007.

[10] M. C. Huebscher and J. A. McCann, "Simulation model for self-adaptive applications in pervasive computing," Proceedings of the Database and Expert Systems Applications, 15th International Workshop, pp. 694-698. IEEE Computer Society, August 2004.

[11] S. Nath, P. B. Gibbons, S. Seshan and Z. Anderson, "TOSSIM: accurate and scalable simulation of entire tinyos applications," *Transactions on Sensor Networks (TOSN)*, Volume 4 Issue 2. ACM, March 2008.

[12] *Persim: A Simulator for Pervasive Spaces* - Project information and download site : http://www.icta.ufl.edu/persim/persim.

[13] Amr Elfaham, H. Hagras, A. Helal, S. Hossain, J. W. Lee, D. J. Cook, "A fuzzy based verification agent for Persim human activity simulator in ambient intelligent environments", 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain, July 18-23. 2010.

[14] *SDDL: Sensory Dataset Description Language* - specifications and resources web site: http:// www.icta.ufl.edu/persim/sddl

[15] M. Patterson and J. Mack, "The cleveland scale for activities of daily living (CSADL): Its reliability and validity," Journal of Clinical Gerontology," pp. 15-28. November 2001.

[16] B. Reisberg and S. Finkel "The Alzheimer's disease activities of daily living international scale (ADL-IS)," International Psycho geriatrics, vol. 13, no. 2, pp. 163-181. 2001.

[17] M. Schmitter-Edgecombe, E. Woo, and D. Greeley, "Memory deficits, everyday functioning, and mild cognitive impairment," Proceedings of the Annual Rehabilitation Psychology Conference, Tucson, Arizona, 2008.

[18] K. Cameron, K. Hughes, and K. Doughty, "Reducing fall incidence in community elders bytelecare using predictive systems," In Proceedings of the International *IEEE-EMBS* Conference, October 1997, pp. 1036-1039.

[19] D. Sundarmoorhi, V.C.P. Chen, S.B. Kim, J.M. Rosenberger, D. F. Buckley-Behan, "A data-integrated nurse activity simulation model," Proceedings of the 38th conference on Winter simulation, Monterey, California, 2006.

[20] I. Stepanov, P. J. Marron, K. Rothermel, "Mobility modeling of outdoor scenarios for MANETs," Proceedings of annual Simulation Symposium, April, 2005.

[21] M. Sierhuis, W.J. Clancey, R.V. Hoof, R.D. Hoog, "Modeling and simulating human activity," AAAI 2000 Fall Symposium on Simulating Human Agents, North Falmouth, MA, 2000.

[22] Shared Inventory of Home-Activity Dataset- BoxLab Wiki Page, http://boxlab.wikispaces.com/Mission

[23] T. V. Kasteren, A. Noulas, G. Englebienne, B. Krose, "Accurate activity recognition in a home setting," Proceedings of the 10th international conference on Ubiquitous computing (Ubicomp '08), Seoul, South Korea, 2008.

[24] *Persim: A Simulator for Pervasive Spaces* - Dataset/Project Repositroy, Persim - pervasiave human activity simulator, web site: http://code.google.com/p/persim-pervasivehumanactivitysimulator

[25] D. Park, S. Lee, E. Song and D. Ahn, "Similarity Computation of Fuzzy Membership Function Pairs with Similarity Measure," Advanced Intelligent Computing Theories and Applications. 2007, Volum e 4682/2007, 485-492

[26] A. Abraham, "Rule-based Expert Systems," in Handbook of Measuring System Design, edited by P. Sydenham and R. Thorn John Wiley & Sons 2005.

[27] Y. Zheng, Q. Li, Y. Chen, X. Xie, W. Ma, "Understanding mobility based on GPS data," Proceedings of the 10th internatinoal conference on Ubiquitous computing (Ubicomp '08), Seoul, South Korea, 2008.

[28] D. Jin, Y. Sung, K. Cho, K. Um, "A Framework Structure For Generation of Action Primitives Using Programming by Demonstration," Proceedings of Artificial Intelligence and Applications/Modelling, Identificaton, and Control(AIA,MIC 2011) 717-045, February 14 – 16, 2011, Innsbruck, Austria