Linac: A Smart Environment Simulator of Human Activities

Gemma Di Federico(✉), Erik Ravn Nikolajsen, Mamuna Azam, and Andrea Burattin

Technical University of Denmark, Kgs. Lyngby, Denmark
gdfe@dtu.dk

Abstract. The identification and construction of datasets of human activities is an extremely time-consuming and resource intensive task, yet researchers cannot refrain from such datasets. The publicly available datasets may not reflect all the researchers' requirements and are not scrupulously documented. In addition, these datasets can cope with just a limited and predefined set of behaviors. To address these challenges, we developed an instrument that allows to simulate the behavior of agents interacting with an environment. The environment is a customized configuration, equipped with sensors. The simulation generates as output a stream of events stemming from activated sensors. In addition, the agents behavior is not fully deterministic, so as to reflect the dynamic nature of human beings and to be as realistic as possible.

1 Introduction

In this work we describe *Linac*, a smart environment simulator. The simulator combines the non-deterministic behavior of human beings with a controlled simulation system, with the aim of generating data streams for research purposes. During the last decade there has been a notable diffusion of sensor systems. These systems allow the collection and analysis of data in real time, gaining the attention of researchers in the field of process mining [11,14]. As a result, the application of sensor systems has spread to many fields with the aim of collecting data, opening up the opportunity to derive new processes. One of the most attractive and innovative application is the derivation of processes related to human behavior [13], paving the way into the world of industry and healthcare [9]. In order to include human beings in process analysis, algorithms need to consider all the specific characteristics derived from this new application. To evaluate, extend, develop and test process mining algorithms, data is needed. There are several ways to collect real-life data, but they are expensive and time consuming. As well as there are scenarios that cannot be replicated (such as accidents or borderline situations). In addition, a thorough understanding of the underlying data as well as its underlying execution is required, and the most appropriate way to do the work is by generating *ad hoc* data (i.e., where the ground truth is known beforehand). The simulation and data generation instrument proposed in this paper allows the configuration of a custom and controlled

J. Munoz-Gama and X. Lu (Eds.): ICPM 2021 Workshops, LNBIP 433, pp. 60–72, 2022.
https://doi.org/10.1007/978-3-030-98581-3_5

simulation. Different agents populate a smart environment, and the process to be carried out is defined by the user. The scenario is equipped with sensors. The simulation consists of the behavior of the agents interacting with the system.

The rest of the paper is organized as follows: existing solutions, the problems of testbeds, public datasets and simulators are presented in Sect. 2. In Sect. 3 we describe our proposed solution, and in Sect. 4 its implementation. In Sect. 5 an evaluation of the approach is presented, and Sect. 6 concludes the paper.

2 Existing Solutions

To evaluate process mining algorithms, data is needed [18]. In particular, when the process comprises the analysis of human behavior, the construction of a dataset becomes an extremely time-consuming and resource intensive task [20]. Three solutions can be considered: testbeds, public datasets and simulators.

The first option consists of constructing a physical testbed. Testbeds are physical environments equipped with sensors, controllers and network components, capable of capturing the state of the environment. Once the testbed has been constructed, a participant performs a predefined list of activities. The acquisition of such datasets is subject to limitations related to the cost and configuration of the actual environment. The layout of the environment must be carefully studied and verified, then all the necessary materials must be acquired, configured and installed. After the construction, the real execution process could start. This process is usually long-lasting, since participants need time to carry out the activities. Large datasets acquired with these techniques are therefore very complicated and expensive to obtain.

The second option to obtain data consists of using publicly available datasets. The main issue is to find a dataset that describes exactly the scenario needed. Additionally, the understanding of a dataset is limited to the documentation provided by authors. Since datasets are usually made up of thousands of events, it is hard to have a detailed description of their content. Consequently, it takes a long time to understand it. Orange4Home and CASAS are two datasets widely used in the literature for the analysis of routines and daily activities. The Orange4Home dataset [7] reports the activities of daily living in a smart apartment, for 4 weeks of recording. The dataset contains recording from 236 data sources. The log is not annotated, and the documentation is limited to the routine plan followed by the occupant during the experiment. The CASAS project [6] provides 66 different datasets, describing labeled and unlabeled activities performed in a smart environment such as assisted care apartments or box offices. The data refers to both single and multi-resident. The documentation provides the list of activities performed and the list of sensors. For the labeled datasets, authors do not specify how the activities were recognized. In addition to what has been said so far, datasets publicly available usually describe common scenarios, characterized by the execution of regular activities. However, we may be interested into processes related to extreme or anomalous cases both for a case study and to include all possible scenarios. Based on the experiment under investigation,

there might be the need to examine specific cases that are difficult or expensive to replicate in reality or that may rise ethical concerns: an accident in an industry or an elderly person falling to the ground. Additionally, to verify the correctness of an algorithm we might want to evaluate it against as many cases as possible. A ready-to-use dataset hardly includes borderline cases, limiting the testing possibility.

The most effective way to obtain data is by using a simulator that replicates each specific use case. In this way we could have a complete control over the environment and the actions, a complete knowledge of the activities carried out and the data produced, i.e., the ground truth. Synonott et al. [19] distinguish between model-based and interactive approaches. Model-based approaches [3, 12,17] consist of the specification of the reference model that the simulation should follow. The abstraction level in the definition of the activities describes the accuracy of the modeled behavior. Renoux et al. [17] propose a model-based approach in which inhabitants interact with a "sensorized" apartment. The user does not script out the actions that need to be carried out but rather provides an idea of how the simulated world works. The abstraction of the model does not allow to represent subtle but significant differences in the behaviors execution. To one hand they are suitable for long running simulations, on the other hand they do not provide clock simulation. Interactive approaches [2,5] consist of a virtual environment (2D or 3D scenario) where the user can operate: the agent interacting with the environment is an "avatar" guided by the user who can interact with each individual sensor. The simulation is precise and realistic, since there is a real human behind the movements. However, the generation of large amounts of data is very expensive as it requires a great effort by the user: these approaches are suitable for testing single activities or short runs. An example is the work of Buchmayr et al. [5] which presents a 2D floor map equipped with sensors. The main objective of the tool is to generate and visualize sensors behavior, in particular the simulation of faulty or unexpected cases. The simulation is performed by the user interacting with the sensors via the mouse.

The majority of the simulators cannot be adapted to simulate several different environments. For example, it is challenging to represent a smart factory scenario with a 3D smart home simulator as it would become very complex in terms of dimensions and objects to have integrated [2]. Or it would be impossible to be able to replicate exact behaviors when the simulator only requires a general behavioral model as input [17]. The tools provided to draw the environment are often limited to the intended objective [3]; as well as they are not always oriented to the generation of datasets. Furthermore, there is a lack of simulators capable of reproducing, in a realistic way, the behavior of a human being [8] interacting with an environment. For this reason, we have developed a simulator capable of representing different environments and scenarios. The agent's behavior is the central focus of the tool. The behavior of the agent is as realistic as possible by introducing different walking speeds and non-deterministic movements. The process carried out by the agent is defined by the user, and it controls the expected result of the simulation.

3 Proposed Simulation Solution

To study processes related to human beings, process mining algorithms need to be evaluated. The data used in the evaluation phase should faithfully represent the reality and the human behavior. During the development of Linac, we have identified several key features that a simulator must consider. The most relevant is that human beings are flexible in their movements [8]: they do not perform movements in a fixed way but introducing variability. Furthermore, we cannot assume that human beings are all equal, i.e. elderly are slower in the movements, while young people are faster. Another factor that gains the attention of our analysis is the amount of data generated: sensors systems tend to produce large amount of data. In the following sections we explain how these challenges have been addressed.

3.1 Configuration of the Smart Environment

The simulation platform allows the configuration of a smart environment. The application is not limited to represent specific domains since it offers a blank canvas where to build up the floor plan, in form of a grid. The drawing tools are walls, entities and sensors. *Walls* are used to physically constrain the environment, while *entities* are objects that are part of the environment (that are not sensors). The agent can interact with these objects. An example of entity is a chair: the agent can move around such an object, but if she is instructed to go to it, then she can "stand" on top of it. Both sensors and entities have a physical and an interact area. Fig. 1a shows an example of floor plan designed as two rooms and two agents (agents represented with green tiles). The purple tiles are non-walkable sensors, while the blue tiles are entities. The walkable sensors are not shown on the map, but they can be inspected on the application. A pre-build selection of sensors is included, but new sensors could easily be defined as Java classes. *Sensors* can be active or passive. While active sensors are activated by the direct interaction by the agent, passive sensors are continuously running to detect changes in their statuses, producing an output at fixed time intervals. How the agent interacts with the sensor is defined via a command in the agent instructions. The trigger frequency of each sensor is configurable.

3.2 Configuration of the Agents' Behavior – AIL Language

The simulation consists of human beings moving and interacting with the objects of the smart environment. The simulation is carried out by one or more agents, each of them representing a person. Sensors and entities are shared among the agents. During the configuration it is possible to define a specific movement speed (i.e. meters per second) for each agent. The definition of the speed allows the simulation of the behavior of people with different ability levels. Furthermore, each agent has a specific set of activities to perform. The list of instructions is specified by the user during the configuration phase, using an application specific

script language called Agent Instruction Language (AIL). AIL has been implemented in order to facilitate the definition of a list of activities to be performed. In fact, the definition of a long set of instructions is a time consuming and complex task. In addition, human behavior is characterized by the repetition of activities, always performed following the same set of actions. As a consequence, the list of instructions is composed by redundant code, e.g., the procedure for preparing a cup of tea will always be executed in the same way. To facilitate the task, we introduced primitive instructions that and can be grouped into macros. The AIL language comprises four primitive instructions which describe basic behaviors:

- **goto**(x,y): instructs the agent to move to a specific position;
- **goto**(name), where name refers to an entity or an active sensor: instructs the agent to move to a random tile in the interaction area of the entity/sensor;
- **wait**(seconds): instructs the agent to remain stationary for the specified amount of seconds;
- **interact**(activeSensor, command): instructs the agent to move to a random tile within the interact area of sensor activeSensor, and interact with it as specified by the command. The command of interaction is reflected in the data produced when the sensor triggers.

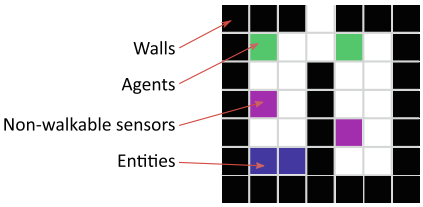
Macro instructions describe complex behaviors. These instructions include a sequence of primitive and/or macro instructions. Macros are a powerful tool to avoid errors, limit redundant code and establish groups of activities that form richer behavior. These instructions are defined as follows:

- **macro**(m) {list of primitive instructions}: defines the macro;
- m(): executes the macro m.

Figure 1b shows an example of primitives and macro. For each agent, a list of instructions is reported, and both agents share the macro called makeTea. Then the macro is then used only by agent a1. The language is designed to be intuitive: an external application can be used to automatically generate instructions starting from an ideal behavior.

3.3 Simulation Execution

The simulation models human behavior, which consists of movements and interactions with the objects. The movements, in turns, comprise journeys between the agent's current position and the target position. To find the path the agent must follow, we implemented a path-finding algorithm. To comply with the flexibility and stochasticity of human behavior, we extended the A* path-finding algorithm [10], constructing a sub-optimal and non-deterministic version of it. Actually, we started from an optimal and deterministic implementation of A*: since we defined the floor plan as a grid of tiles, it is easy to translate the grid of tiles into a graph of nodes needed for the A* algorithm. The A* algorithm uses a heuristic function to calculate a path between two nodes on a graph.



(a) Floor plan example

```
macro(makeTea) {
    interact(faucet,TURN_ON); wait(3);
    interact(faucet,TURN_OFF);
    interact(kettle,TURN_ON); wait(60);
}
agent(a1) {
    makeTea();
    goto(couch);
}
agent(a2) {
    goto(3,4);
    wait(100);
}
```

(b) AIL example

Fig. 1. A floor plan and a possible list of instructions for two agents

The heuristic function can be either admissible or inadmissible: in the first case it always calculates a distance that is shorter than or equal to the actual distance of reaching the goal node (optimal path); in the second case, it can calculate a distance that is longer than the actual distance of reaching the goal node (sub-optimal path). Furthermore, since neither the A* algorithm itself nor the commonly used heuristic functions contains a random variable, the calculated path is deterministic. However, a path-finding algorithm that is deterministic and optimal is not a good model for human movement: *(i)* a human does not take the same path every time between two points, *(ii)* a human does not take a random walk between two points and *(iii)* humans do not always take a shortest path between two points [4]. To tackle these problems, we defined an inadmissible heuristic function that includes a random variable. To obtain this heuristic function H, we considered the Euclidean distance between the two points plus a value $R \sim U(0, n \cdot L)$, randomly drawn from the uniform distribution between 0 and $n \cdot L$. In this case, L is the length of the sides of the tiles in the floor plan and n is a parameter indicating the degree of sub-optimality (the higher the value the less optimal the path). Adding the random variable to an otherwise admissible heuristic, overestimates the distance, thus making the heuristics inadmissible and hence sub-optimal. Furthermore, since it contains a random variable, it will be non-deterministic. We then experimented with increasing the threshold value n as much as we could, while avoiding the agent making too many counterproductive movements. Here we defined counterproductive movements as a move that leaves the agent at the same distance to the goal node or a longer distance away from the goal node.

3.4 Clock Simulation

A fundamental aspect of Linac is the clock simulation. Being able to run the simulation in real time is valuable when evaluating online algorithms and, on the other hand, not practical when simulating multiple days, as this would take multiple days in the real life as well. One way to lessen this impracticality is to scale how fast time progresses in the simulation relative to the time progression

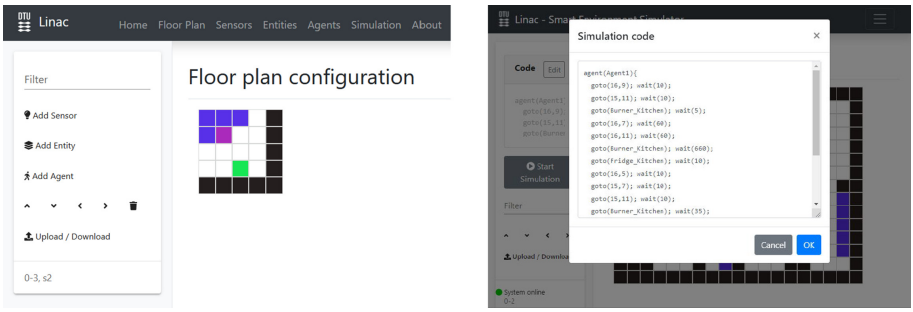


Fig. 2. Screenshots of the Linac floor-plan page (left) and simulation (right)

in the reality. This can be achieved by defining how many real time seconds a simulated second should take.

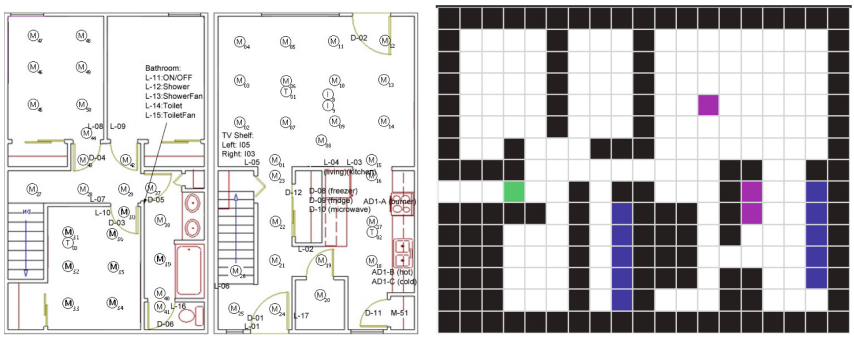
3.5 MQTT Output

The output produced by Linac is in the form of MQTT messages. The MQTT protocol [15] is based on a publish/subscribe model that decouples the publisher that sends the message from the subscribers that receive the message by the use of a broker. This architecture involves one or more publishing clients that publish messages under a topic. In our context each sensor would constitute a publishing client. Using this system, we could also focus only on a specific sensor by subscribing to a specific topic.

4 Implementation

Linac is implemented as a web application[1]. The application comprises a frontend and backend. The latter[2] is a server-side application implemented in Java. The former[3] is a web application implemented in TypeScript and Vue.js. The web application communicates with the server using a restful interface. The backend, in turns, exposes a set of APIs that can be triggered also from other applications. For example, a script can be used to generate and execute multiple simulations by programmatically generating the corresponding AIL code. The web application is organized in three main components: the floor plan, the sensors/entities/agents pages and the simulation page. A screenshot of the main page is shown in Fig. 2. The implementation allows the configuration and simulation of the environment. Once a simulation is running, it is possible to see the movements of the agents on the map in real time. Further details on the implementation are available on the report [16].

[1] See http://linac.compute.dtu.dk.
[2] Source code available at https://github.com/DTU-SPE/linac-backend.
[3] Source code available at https://github.com/DTU-SPE/linac-frontend.



(a) Floor plan of CASAS [1]

(b) Floor plan in Linac (floor sensors not visible)

Fig. 3. The two floor maps

5 Evaluation

To evaluate the behavior of the simulator, we decided to replicate an existing dataset, derived from a real scenario, and compare the results. The dataset chosen is one from CASAS. The dataset represents sensor events collected in a smart apartment testbed. The apartment has two residents performing their normal daily activities. The dataset provides both the raw and the annotated events, but we used the annotated one to recognize activities. Our evaluation comprises three phases: in the first phase we tried to replicate the floor plan, after that we analyzed the annotated dataset to identify how each activity has been performed, concluding with a running simulation. Being able to successfully replicate the CASAS dataset would allow us to show the capabilities of Linac in terms of realism of the data, thus allowing us to derive new datasets where specific situations or behaviors appear.

5.1 Configuration

Floor Plan Design. The Linac tool allows the configuration of an environment by means of a grid of variable size. On the grid, wall entities and sensors are distinguished by colors: black, blue, and purple respectively. The CASAS environment is composed of a 6 rooms apartment, equipped with more than 50 sensors (motion, item, door, water, temperature, electricity sensors). The floor plan was designed by transforming the sensors layout into the form of a grid. Then, all the sensors have been configured, and for each of them the physical area, the interaction area and the trigger frequency have been defined. Once the two maps matched, we moved on to the next phase. Figure 3 shows the two floor plans: Fig. 3a illustrates the original map provided by CASAS while Fig. 3b refers to the grid layout designed using Linac.

Table 1. Datasets structure and results comparison

	Duration		# of sensors		# of events	
	CASAS	**Linac**	**CASAS**	**Linac**	**CASAS**	**Linac**
Simulation1 (`Bed_to_Toilet`)	2 min	2 min	12	13	47	51
Simulation2 (`Meal_Preparation`)	19 min	19 min	14	14	360	440

Agent Instructions. The CASAS dataset is labeled, but no information regarding how the activities were carried out is provided. For this reason, we choose to focus only on two activities, rather than analyzing all the 13 proposed. The activities are `Bed_to_Toilet`, referred as Simulation1, and `Meal_Preparation`, named Simulation2. Starting from the list of triggered sensors in the CASAS dataset, we reconstructed the path followed by the agent. At this point, we drawn up a list of instructions that the agent had to follow to carry out the specific activity. A key feature of our simulator is the **`goto`** primitive statement which allows to instruct the agent to reach a specific tile, entity or active sensor, without having to provide coordinates for each step. In this way, it is easier to define the activity list. Once completed the list of instruction, we moved on the simulation phase.

The Simulation. The simulation tool of Linac allows for defining the date, the relative time and the configuration of MQTT. The choice of date and time let you to place the simulation at a specific moment in time. The simulation could be performed in real time speed or in a specific relative time, that is how many real seconds a simulated one should take. For this evaluation task, we performed the two simulations (`Bed_to_Toilet` and `Meal_Preparation`) on the same floor plan. The first simulation refers to a movement between two rooms, that is the path followed to go from the bed to the kitchen. The second refers to the activity of meal preparation. We used the relative time to run the simulations, which took less than 1 minute to execute. The data for both simulations is available for download[4].

5.2 Results

The four datasets (2 simulations $\times$ 2 datasets) are structured as reported in Table 1. For each couple of simulations, the durations are the same. The total amount of unique sensors activated differs in Simulation1 and this is caused by the non-deterministic path-finding algorithm implemented in Linac. In other words, the path that agents follow is characterized by a certain random variability, which led to the activation of an additional sensor. A gap could be observed in the number of events generated, especially in Simulation2. This discrepancy does not imply differences in the behavior: the index that causes this spread is the trigger frequency of each sensor. In Linac, each sensor has a fixed trigger frequency (for this simulation, configured to 5 seconds). In the CASAS dataset,

[4] See https://doi.org/10.5281/zenodo.5386318.

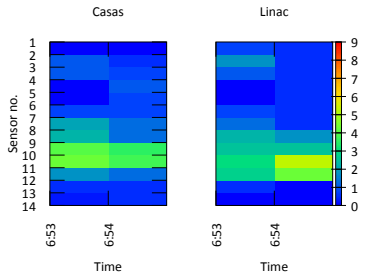


Fig. 4. Sensors triggered in Simulation1 in CASAS and Linac

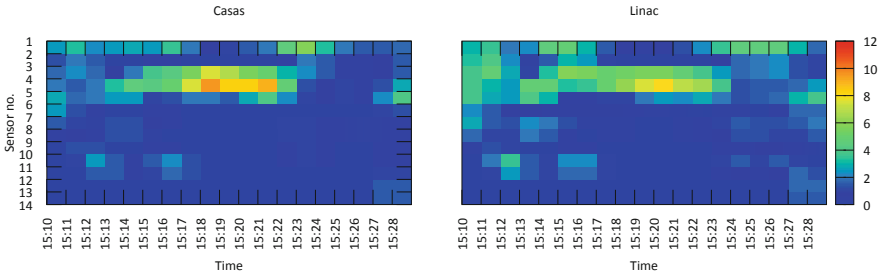


Fig. 5. Sensors triggered in Simulation2 in CASAS and Linac

on the other hand, this information is not provided. To better evaluate the differences between the two simulations, we plot them on heat maps.

The first map, reported in Fig. 4 refers to Simulation1. The maps show the number of times that each relevant sensor is triggered during the simulation period (with a time grouping of 1 min). As we can notice from the color variations, the two maps seem to behave very similarly over the same sensors/time. However, some discrepancies could be observed in the intensity color. Since the simulation is spread over just two minutes, and the number of events generated is small, we cannot consider the impact of sensors triggered only few times. In fact, the agent did not spend long periods in those zones, but was only passing through them, generating a single trigger for each sensor.

The second simulation lasts for 19 min, and this makes the heat maps in Fig. 5 more accurate. In fact, the average amount of triggers for each sensor is higher than the previous simulation. Therefore, as the simulation lasts longer, the total number of events generated is much greater. Looking at the maps we have to consider that the values on the x-axes are one minute units. Therefore, there could be sensors activated a minute before or a minute after others (e.g. 15:11:58 and 15:12:01), which could be considered misalignment in the graphs but, for simulation purposes, are absolutely tolerable. Both the maps in Fig. 5 intensify in colors in the time interval 15:15–15:23, suggesting that the main behaviors occurred during that interval.

To better evaluate the resulting simulations, we computed the Pearson's coefficient for the two scenarios. The Pearson's coefficient is a measure of the strength of a linear correlation between two variables. We computed the correlation for each sensor, and then we calculated the average value for all sensors. Both Simulation1 and Simulation2 resulted in a coefficient of 0.93. These results depict a strong correlation between the sensors activation during the simulations. Therefore, we can state that, in both simulations, the behavior of the simulated sensors (by Linac) and the behavior of the reference sensors (CASAS) agree and hence we can conclude that Linac is capable of effectively mimicking a real dataset.

The objective of the simulations is to replicate the behavior of human beings (in CASAS) as movements of agents (in Linac) through a simulation. The two simulations conducted, and the resulting evaluation, highlighted that Linac is able to replicate a real behavior such as that collected in the CASAS dataset.

6 Conclusions and Future Works

We presented a smart environment simulator for the generation of datasets, in the form of streams. The simulator could be used to configure different environments, thanks to its structure made up of walls, entities and sensors. The behavior of the agents is composed of movements inside the environment and of interactions with entities and sensors. The behavior is dictated by means of a list of instructions that the agent must follow and that can be described using the language AIL, that we created for this purpose. The simulation uses a non-deterministic sub-optimal algorithm to replicate the stochasticity of human behavior. The simulator, additionally, offers the functionality for setting the speed of movement for each agent. A simulated clock is used to solve problems related to the long running of the simulations. The clock is fully configurable, and the emulation consists of running real simulations but in which time passes much faster. The last aspect to be summarized is the output that uses the MQTT protocol to stream the data, that is, sensors readings.

The behavior of the Linac simulator has been compared with a dataset describing the behavior of an actual person inside a testbed environment. The analysis gave very positive results suggesting that Linac is able to reproduce the same movements.

All things considered, it can be said that the Linac simulator is suitable for the generation of realistic datasets referring to the human behavior. The data generated can be used to test and evaluate algorithms, thus resulting in a valuable tool for researchers.

Aspects to improve comprise the definition of the library of sensors available and the extension of the AIL language. For example, considering the `wait` statement, used to instruct the agent to remain stationary for a certain period, there are cases in which we want the agent should not remain exactly in the same tile, but randomly move nearby. These aspects could contribute towards an ever higher level of realism.

References

1. CASAS - Daily Life Spring 2009. http://casas.wsu.edu/datasets/twor.2009.zip
2. Alshammari, N., Alshammari, T., Sedky, M., Champion, J., Bauer, C.: OpenSHS: open smart home simulator. Sensors **17**(5), 1003 (2017)
3. Ariani, A., Redmond, S.J., Chang, D., Lovell, N.H.: Simulation of a smart home environment. In: ICICI-BME 2013, pp. 27–32 (2013)
4. Banovic, N., Buzali, T., Chevalier, F., Mankoff, J., Dey, A.K.: Modeling and understanding human routine behavior. In: Proceedings of CHI, pp. 248–260 (2016)
5. Buchmayr, M., Kurschl, W., Küng, J.: A simulator for generating and visualizing sensor data for ambient intelligence environments. Procedia **5**, 90–97 (2011)
6. Cook, D.J., Crandall, A.S., Thomas, B.L., Krishnan, N.C.: CASAS: a smart home in a box. Computer **46**(7), 62–69 (2012)
7. Cumin, J., Lefebvre, G., Ramparany, F., Crowley, J.L.: A dataset of routine daily activities in an instrumented home. In: Ochoa, S.F., Singh, P., Bravo, J. (eds.) UCAmI 2017. LNCS, vol. 10586, pp. 413–425. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67585-5_43
8. Di Federico, G., Burattin, A., Montali, M.: Human behavior as a process model: Which language to use? ITalian forum on Business Process Management (2021)
9. Seoane, F., Traver, V., Hazelzet, J.: Value-driven digital transformation in health and medical care. In: Fernandez-Llatas, C. (ed.) Interactive Process Mining in Healthcare. HI, pp. 13–26. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-53993-1_2
10. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE SMCS **4**(2), 100–107 (1968)
11. Janisch, C., et al.: The internet-of-things meets business process management. A manifesto. IEEE Syst. Man Cybern. Mag. **6**(4), 34–44 (2020)
12. Kormányos, B., Pataki, B.: Multilevel simulation of daily activities: why and how? In: CIVEMSA, pp. 1–6. IEEE (2013)
13. Leotta, F., Mecella, M., Mendling, J.: Applying process mining to smart spaces: perspectives and research challenges. In: Persson, A., Stirna, J. (eds.) CAiSE 2015. LNBIP, vol. 215, pp. 298–304. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19243-7_28
14. Mandal, S., Hewelt, M., Oestreich, M., Weske, M.: A classification framework for IoT scenarios. In: Daniel, F., Sheng, Q.Z., Motahari, H. (eds.) BPM 2018. LNBIP, vol. 342, pp. 458–469. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11641-5_36
15. Naik, N.: Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In: ISSE, pp. 1–7 (2017)
16. Nikolajsen, E.R., Azam, M.: A platform to simulate agent interactions with IoT devices to facilitate process mining algorithm research. Technical Report, DTU (2021). https://findit.dtu.dk/en/catalog/2691894192
17. Renoux, J., Klugl, F.: Simulating daily activities in a smart home for data generation. In: WSC 2018, pp. 798–809 (2018)
18. Rozinat, A., De Medeiros, A., Günther, C., Weijters, A., van der Aalst, W.: Towards an evaluation framework for process mining algorithms. BPMcenter.org (2007)
19. Synnott, J., Nugent, C., Jeffers, P.: Simulation of smart home activity datasets. Sensors **15**(6), 14162–14179 (2015)
20. Vinciarelli, A., et al.: Open challenges in modelling, analysis and synthesis of human behaviour in human-human and human-machine interactions. Cogn. Comput. **7**(4), 397–413 (2015)