Interactive Social Agents Simulation Tool for Designing Choreographies for Human-Robot-Interaction Research

Olivier Hauterville[1], Camino Fernández[1,2], Phani Teja Singamaneni[1], Anthony Favier[1,3], Vicente Matellán[1,2], and Rachid Alami[1,3]

[1] RIS group - LAAS (CNRS), Colonel Roche, 31400 Toulouse (France)
{Olivier.Hauterville, phani-teja.singamaneni, anthony.favier, Rachid.Alami}@laas.fr, http://https://www.laas.fr/public/
[2] Robotics Group, Universidad de León, 24007 León (Spain),
{camino.fernandez,vicente.matellan}@unileon.es https://robotica.unileon.es
[3] ANITI, Univeristé Fédérale Toulouse Midi-Pyrénées, B612, Rue Tarfaya, 31400 Toulouse (France)

Abstract. Social robotics research requires the replicability of the experiments used to validate new technical or scientific claims. However, humans behavior is difficult to replicate. We have designed and implemented a system to simulate interactive agents that can be perceived by a robot in a robotic simulator. These agents can exhibit social behaviour in human group movement, for example, by talking in pairs. These humans are interactive with the robot and can be perceived so that the robot can react to their behavior. The humans can be choreographed to create high-level social behaviours such as waiting for an elevator, getting in and out of it, or standing in front of a store window and moving together to the next one. The tool would provide metrics to evaluate the robot's performance in defined social situations.

Keywords: HRI, simulation, social robotics

1 Introduction

One of the basic requirements of science is the replicability of the experiments designed to validate any model, theory, or hypothesis proposed. However, when the research topic involves interaction with people, as in social robotics, this replicability is difficult to achieve. Even detailed descriptions like the one in [9] are really difficult to replicate. Furthermore, small variations in the demeanour of the people have a significant impact on the behaviour of the robot. Software simulators cope with this problem by providing the ability to replicate the exact same situation repeatedly. These simulators should be able to provide "simulated" human behaviour for the development of socially acceptable robotic behaviours, but currently offer only limited capabilities in this area.

Likewise, developing skills for autonomous robots also requires testing the skills on real robots, but designing and conducting these user studies are costly

and time-consuming because they typically involve a large number of people to be statistically significant. Usually, tuning and testing tasks are performed on simulators which, as previously mentioned, should provide simulated humans that can actually be useful when designing skills to interact with people.

Simulations are also used for benchmarking the robotic navigation algorithms. If the same environment and conditions are presented to different algorithms, different metrics can be calculated to assess and compare the performance of the algorithms.

The goal of the tool described in this paper is to provide a means to generate scenarios in which predefined social interactions of groups of reactive humans can be used to test the social performance of a robot behaviour under evaluation. In the rest of the paper, we will refer to it as *tested robot*. The aim is to use the system for benchmarking human-robot interaction behaviours.

The rest of this paper is organized as follows. The state of the art of the simulation of robots in the presence of humans is discussed in section 2. The requirements, analysis and tool design are described in section 3. Section 4 shows the implementation details of the tool and section 5 includes some basic use case study of a specific scenario. Finally, section 6 summarizes the contributions and the envisioned future works.

2 State of the art

There are several simulators commonly accepted in the robotics research community such as Gazebo [10], Unity [7], Coppelia (V-REP)[17], Morse [11], Unreal, etc. These simulators have proven to be very useful for testing basic service robotics skills, such as obstacle avoidance, localization, global navigation, etc. However, they are not well suited for human simulation. Some of them, like Gazebo, provide basic mechanisms (actors) to simulate humans, but these simulated humans can exhibit only basic behaviors, such as going to a point, and are not responsive and not even aware of the robot's presence; they are not even fully integrated with the physics of the simulation and are closer to "decorations" than to real simulated entities.

The problem of human social navigation is also an active research question in the crowd simulation research community. Crowd simulation has applications in many areas, from transportation to psychology, the design of video games, or digital effects for movies. But usually, they are more concerned about the management of large crowds than about the intelligent behaviours of the individuals. Unfortunately, classical crowd simulators such as Continuum [18], Pedsim [4], or Menge do not simulate robots. Some adaptations have been proposed to include robots in these crowd simulators. The best known is Pedsim-ROS [14], a set of packages developed at Freiburg University for 2D pedestrian simulation integrated with ROS. However, these simulators usually provide simple behaviours for simulated humans. For instance, Pedsim-ROS only provides local collision avoidance based on the Social-Force Model (SFM) [5].

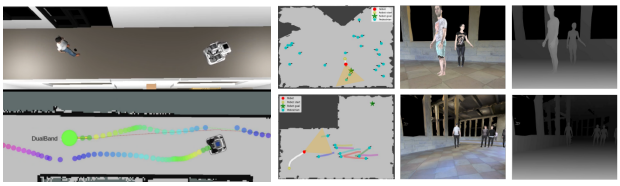

Fig. 1. InHuS and SocNavBench human-robot simulators.

A more recent related work is SOCIALGYM [6], a lightweight 2D simulation environment for benchmarking, training and evaluating social navigation behaviours. Social navigation is modelled as an action selection problem, and the optimal action is selected from a set of four discrete sub-policies: Halt, GoAlone, Follow and Pass. In SOCIALGYM, each simulated human moves back and forth between a starting point and a goal pose by computing a global path of intermediate nodes between them and using a separate local planner (Pedsim-ROS social force model) to handle dynamic obstacles. It also includes a "Navigation Module" responsible for planning the obstacle-free path from the starting point to the goal. Therefore, SOCIALGYM only provides a combined behaviour using the above sub-policies but does not provide a real social interaction.

Regarding benchmarking, currently most proposals to compare social navigation algorithms are based on datasets. Some of them are designed to test perception, such as SRIN (Social Robot Indoor Navigation) [8] which contains images taken by medium size robots. SocNav1 [13] includes 9,280 scenarios containing locations of humans and a robot in an 8x8 area and the subjective level of discomfort generated by the robot's presence in that scenario. Although datasets are essential for machine learning training algorithms, their use as benchmarking tools is not justified, as they are intrinsically static. Therefore, dynamic simulators should be part of the benchmarking process.

SocNavBench[2] is an intermediate testbed that mixes a dataset and some simulation. It comprises a photo-realistic renderer, a set of navigation scenarios based on real-world pedestrian data, and a set of metrics to characterize the performance of robot navigation algorithms. Instead of simulating behaviour, SocNavBench derives real-world pedestrian trajectories from real behaviour datasets. It applies real-world textures on pedestrians and the environment to overcome the inability to obtain sensor inputs for these datasets. Although it is an improvement over mere images, simulated humans have fixed, non-reactive behaviour. They can only exhibit pre-recorded behaviour and only existing behaviors can be reproduced. The creation of new scenes requires the recording new real data.

Liu et al. [12] have also proposed interactive pedestrian simulation in iGibson, sharing some of the ideas with our proposal. They decompose the problem into a "Global Planning" module based on A^* to compute coarse-grained waypoints to guide pedestrians towards their goals and a "Local Planning" module to control pedestrian positions at each time step. The main difference is that they compute

these trajectories for each of the simulated humans focusing on learning socially-aware individual navigation behaviours. Our proposal is not oriented towards learning individual social behaviours; we just want to simulate group behaviours that can be used to test the social skills of robots.

The closest tool to the goals of the proposed system is the Intelligent Human Simulator (InHuS) [3], which proposes a generic architecture for simulating intelligent human agents. It provides a complex "Human Behavior Model" module capable of defining goals, managing relations with other agents and building perception from the data received from the simulator. So, it can be considered as a complete cognitive architecture that could be used to control simulated humans or robots. It has been tested for social navigation, but only with one simulated human, which uses CoHAN[16] as the robot controller, but enhanced with two "attitudes": *Harass* and *StopAndLook*.

Finally, there are other approaches for generating the behaviour of simulated entities. For instance, in the classical planning community, there are simulators as PDSim[15], where the behaviour of objects in planning problems can be defined using PDDL. These "animations" may include moving an object onto another object, letting an agent follow a path between two points on a map, etc.

In summary, existing simulation tools that integrate robots and humans do not allow simulating realistic social behaviours of the simulated people. The only available alternative is the use of teleoperated avatars or basic behaviours. The need for simulation tools that provide high-level primitives to define the behaviour of simulated humans has been identified, and some commercial products are starting to appear, for instance, PedSim Pro V1.2 has incorporated the "queuing behaviour".

The goal of IMHuS (Intelligent Multi Human Simulator), the tool described in this paper, is to provide an open-source toolkit for defining high-level reactive simulated humans with the ability to show the behavior of social groups, but using realistic standard robotics simulators that allow researchers to use models of their real robots, both for debugging their algorithms and for benchmarking and repeatability.

3 Design of a step-based social simulation

The goal of IMHuS is to standardize the validation of the autonomous robot behaviour in the presence of people, allowing researchers to design repeatable human social situations. For example, defining a set of waypoints where different simulated people arrive, meet and move as a group to a different position, or two people facing each other and then moving together to a different location.

Our proposal considers that both individuals and groups have to be included in the simulated environment and that in both cases, each simulated person should exhibit adaptive behaviour. To achieve this goal, we assume that global path planning for the whole set of agents is more suitable for defining fixed social behaviours than the individual path planning approach. This assumption also serves the purpose that humans exhibit a common social behaviour that the

robot must be able to detect in order not to cross, for example, through the middle of a social group.

A central process can have perfect knowledge of the simulated environment, accessing the real and accurate positions of all the elements of the simulations (obstacles, robot, etc.). It is not designed to emulate an embedded agent, since the robot simulation process has to obtain the information through the noise-simulated sensor readings provided by the simulator. This process can use the API of the simulator to get all the information directly, without noise and identification problems (it will know which types of elements are in the simulation and their state). For instance, it does not need to identify if something is "a door" or if it is opened. The door state is obtained directly from the simulator.

3.1 Choreography oriented simulation

The constraints and assumptions made for IMHuS are:

1. The set of persons $P = \{p_1, p_2, ...p_n\}$ in the environment is defined *a priori*, and each person will be unequivocally identified by its ID (p_i) during each execution.
2. The number of locations $L = \{l_1, l_2..., l_m\}$ for these people is also known *a priori*.
3. The number of locations will be greater than the number of people $|L| \geq |P|$.
4. In a given time-step, only one person can stay at an individual location l_i.
5. Group locations L_i^n will have a maximum number n of individual positions $l_1, l_2, ..., l_n$. For instance, an elevator with four positions will be named L^4.
6. The number of actions is also finite and known.

The goal of the tool is to allow researchers to have a high-level definition of a "choreography" of people moving in a social way. For instance, let's consider an example, first a set of people (p_1 to p_4) is defined. In a first step, p_1, p_2 and p_3 have to move from their initial positions to a "group location" (L^3). This joint navigation is represented in figure 2 as a continuous top-opening box grouping the three people at time-step t_0 and a double arrow labelled with the goal destination. In the same way, p_4 will remain in its position but will face a particular direction (30 degrees), indicated by the circle with an arrow.

At t_1, p_1 and p_2 engage in a pair-move to L^4, while p_3 and p_4 initiate individual moves, indicated by a single arrow. A pair-move is a specific type of group move included in the design because it is usually the most common one and because it can be considered as the smallest group unit. At t_2, the two persons moving as a pair would have reached their destination and p_1 will initiate its movement towards l_{10}, while p_2 remains in its current position with no particular orientation (indicated by a circle wit no arrow), and p_3 and p_4 continue to navigate to their targets.

p_1 at t_2 begins to move individually towards l_{10}, which reaches at t_3 while p_3 and p_4 continue to execute their solo movement. p_3 reaches l_{11} at t_4, when p_1 and p_2 begin to face in the same direction. Finally, at t_5, p_4 reaches its destination (l_{12}) and the choreography ends.

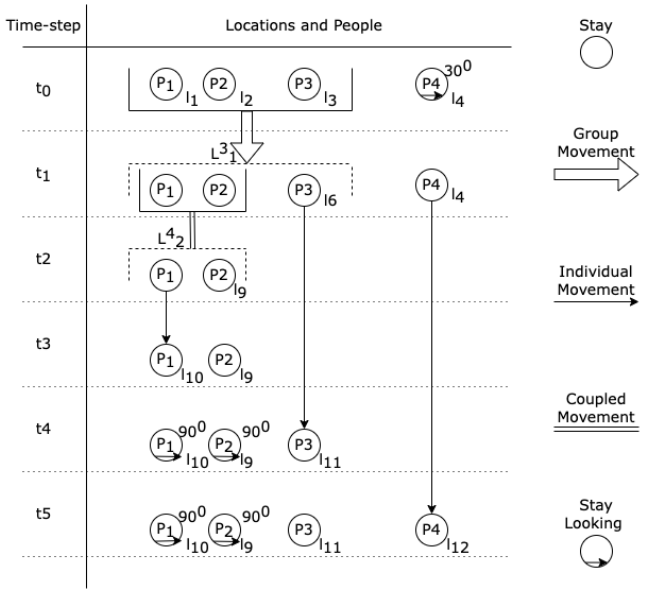

Fig. 2. Schematic representation of the definition of a social navigation choreography

Time-step (t_i) in figure 2 means "*choreography steps*", that is, significant moments for the definition of the simulation, it does not refer to a magnitude measured by a clock. We will refer to them as *steps*, which is an increased ordered sequence of discrete points in time at which a given set of events has to occur. For example, $step_0$ usually specifies the initial state of all the elements of the simulation, i.e., the position of the robot, human', and the other elements. A typical step specifies a set of actions that are initiated at that instant, a navigation task for one of the humans, for instance, at $step_1$ (t_1 in figure 2).

This is the type of simulation that the tool should be able to generate. Figure 3 shows the general architectural framework. This tool receives the definitions of the social scenarios in a definition file (an XML in its current version). It then manages the simulation, obtaining the simulation data, using existing libraries and other tools (such as *move_base* for calculation of trajectories in ROS), and finally generates the log data for evaluation.

The definition of the choreography shown in the upper left of the figure 3 is defined in XML. The design allows five types of *basic actions* that the *agents* (simulated robots or simulated people) can perform:

Navigation actions : Those actions modify the global pose of the agents in the environment. Typical actions in this group are $GoToPose$ and $Wait$.

Turning actions : Actions related to the facing of the agents, such as $LookAt$ and $Turn$ depending on the way the action is specified.

Grouping actions : These actions manage the creation and dismantling of groups.

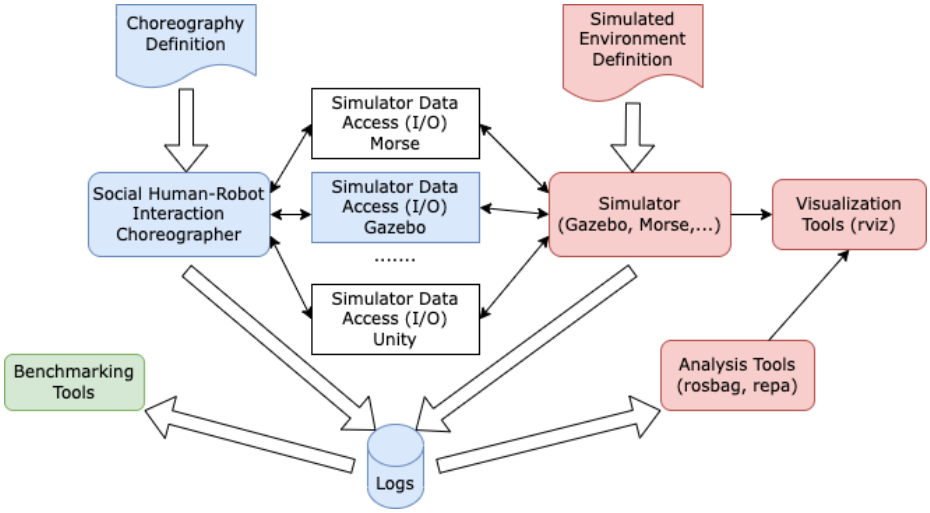

Fig. 3. Global simulation architecture. Red signifies existing tools, blue components are those described in this paper, green signifies on-going development, and uncolored boxes alternative simulators under consideration.

Attitude actions : Actions modelling the high level behavior of the agent. For instance, a simulated human can be ordered to *harass* the robot.

Synchronous actions : Actions related to the environment to be accomplished by an agent during one specific step of the simulation. They have been standardized as *publish* and *subscribe*.

Asynchronous actions : Actions related to the environment and not associated with a specific step in the timeline of the simulation. They have been standardized as *request* and *respond*.

These actions can be applied to a single agent or a group. Basic actions can be integrated into a *compound_task*. For defining these actions, the following components are used:

- *map*: corresponds to the "world" where the simulation will happen. Inside the map a set of *objects* can be defined specifying their individual ID.
- *poses*: correspond to the "location" used in figure 2. They are made up by the x, y position and orientation θ, and a *radius* of tolerance for the motion planner to consider that the goal has been reached.
- *agents*: that can be either *robots* or *humans*, each of them identified by an unique ID. They are given an initial pose where they appear in the world.
- *groups*: they can be created and dismantled during the simulation

The evolution of the simulation is based on steps, as described in the previous section. The regular steps are synchronous, but there is an asynchronous step for interacting with the tested robot:

- A scenario is composed of a set of regular steps and a singular asynchronous step.
- Steps are made up of different actions like navigation, grouping, etc.
- Every action of a step is executed *at the same time*, meaning that the actions are executed in simulated parallelism.
- One step ends when all its actions have finished.
- The asynchronous step runs in parallel to the execution of the synchronous steps.
- The scenario ends when the last regular step ends.

4 Implementation

In the current version choreographies are defined in an XML file that includes four main sections[4]:

- Map = poses and objects. Example: poses definition.
```
<poses> ::= <pose> (<pose>)* ;
    <pose> ::= <poseID> <x> <y> <theta> <radius> ;
```

- Agents = humans/choreographed robots with their initial poses, and groups with their composition. Example: groups definition.
```
<groups> ::= (<group>)* ;
    <group> ::= <groupID> <pair> (<human> (<human>)*) ;
        <pair> ::= <pairID> <human> <human> ;
```

- Tasks = generic actions. Example: compound tasks and look-at action definition.
```
<compound_tasks> ::= <compound_task>* ;
    <compound_task> ::= <compound_taskID> <action> (<action>)* ;
<lookAt_action> ::= <actionID> (<humanID>|<robotID>|<objectID>) ;
```

- Scenarios = step elements of synchronous steps and asynchronous step. Example: scenario definition.
```
<scenario> ::= <name> <step> (<step>)* (async_step);
    <step> ::= <stepID> <stepElement>;
        <stepElement> ::= <agentsID> <pose>|<compound_task>;
        <agentsID> ::= <humanID>|<robotID>|<pairID>|<groupID>;
    <async_step> ::= <respond_event_action>;
```

The prototype has been implemented as a new version of the InHus tool ([1], [3]) which connects to the Gazebo simulator to obtain the information about the world. It uses the *move_base* ROS to generate the global plan for each agent and updates the positions of each agent in the next step of the simulation accordingly. The new version is capable of handling the navigation of multiple agents (humans) in parallel while managing conflicts and completing individual tasks

[4] A full version of the syntax can be accessed here (link)

as in the previous version. It also provides an updated graphical user interface for repeatedly selecting and executing scenarios defined in the XML file.

The IMHuS tool shown in Figure 4 has been structured in three layers: the IMHuS layer (in blue), its configuration in the application layer (in yellow), its communication with the simulator and ROS (in red). The robot whose behavior would be tested in the tool has also been included (in violet).

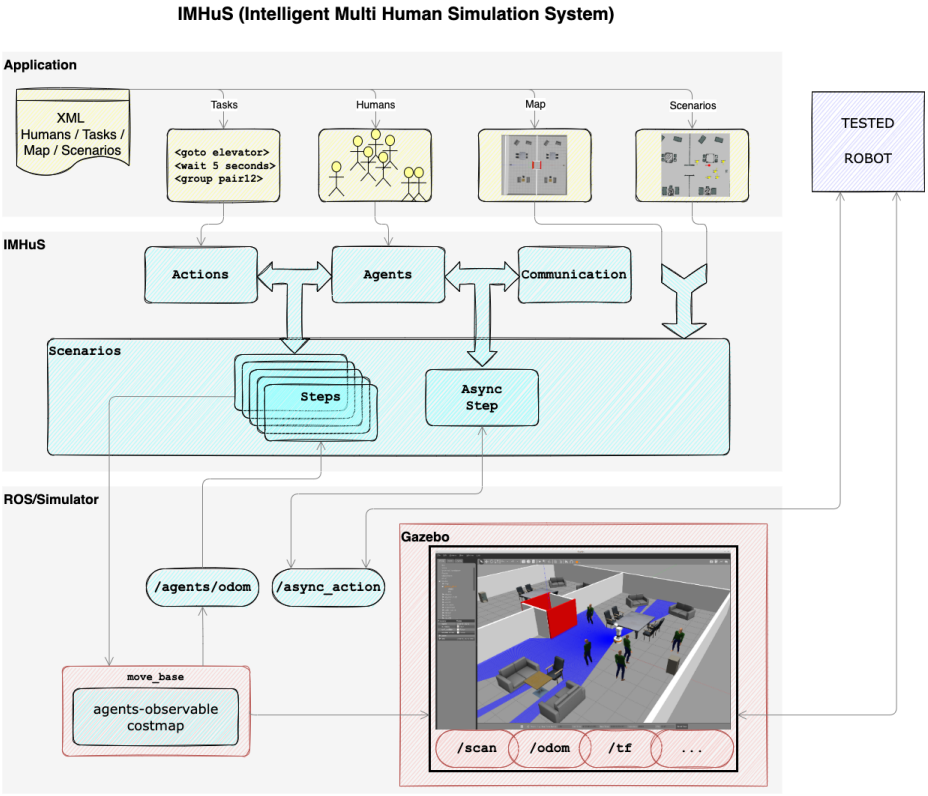

Fig. 4. Structure of the new IMHuS tool.

Application layer This layer includes the XML configuration file needed to use IMHuS by defining its main components: map, agents, tasks and scenarios. The map includes all the locations of the agents to be used during the choreography, as well as those of static objects. These locations are represented as poses and a name is assigned to each of them. The description of the agents includes the name and initial position of all humans and robots choreographed, and also the name and composition of the groups that will appear in any step. Generic tasks are described with no specific subject to

perform them, so that they can be reused by several agents. Last, scenarios describe the steps of the choreography. Each step includes a set of tasks assigned to a particular agent or to a groups. They are the step elements.

IMHuS layer The tool interprets the information contained in the XML configuration file to represent the tasks as actions and the humans/robots as agents. When the scenarios are run, a command combines actions with agents for every step element. The commands concerning each agent are executed in a separate thread inside a step. The step finishes when all the threads are done. This is the way simultaneous movement of agents is achieved. This is the behavior of the tool for the choreography, but taking into account that the tested robot will be present, there must be a way for it to communicate with the agents of the simulation, just as it would happen in the real world. The asynchronous step is in charge of this task. At any point in time, the tested robot can ask something, for example, to press the elevator button. This action would be done through the communication module and would be answered by the agents in the simulation in this asynchronous step by means of a respond action. Not every agent can respond to a request from the tested robot. For instance, if the robot asks for someone to press the button and no one is around, the request will be dropped, counting as a failure.

ROS/Simulator layer This layer is used by IMHuS to place the agents in the simulation and to ask for the trajectories to move them around, as well as to communicate with the tested robot. The agents are placed in the world as obstacles so that when `move_base` is asked for a new path, they are avoided. The costmap obstacle layer has been modified to obtain what we call the *agents-observable costmap* that IMHuS needs.

Interface of the tested robot The way to include the software of a robot in the simulator for its social behaviour to be tested would be through the simulator, Gazebo for now. Outside the simulator, the only communication would be through the asynchronous action thanks to which the robot can send a request action to be answered by a respond action of one agent of the simulation.

5 Use cases

This module is available as Open Source and it has been evaluated in the elevator scenario, as defined for SciRoc competition (IMHuS repository link).

An elevator scenario will be used to show a running example of the behavior of the IMHuS system with a tested robot, and also to explain the communication between the *tested robot* and the agents of IMHuS (video available at this link). The proposed scenario includes five human agents and the *tested robot*, whose goal is to go to the second floor. In order to do that, it has to ask a human agent to push the elevator button. Figure 5 shows the initial situation where the *tested robot* is approaching the elevator, human 2 is walking and the rest of humans are idle.

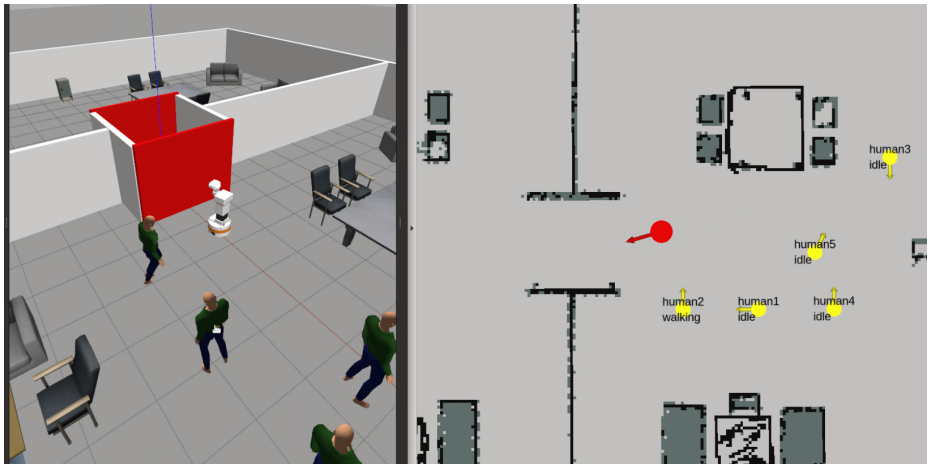

Fig. 5. Initial situation.

Once the *tested robot* gets to the elevator door, it requests an action to be performed by one of the humans. In order for a *tested robot* to trigger an action in IMHuS, it has to publish a specific message type on the `async_action` topic (see Figure 4). By doing so, it triggers an asynchronous action response from IMHuS. This message should contain the time of emission, the name of the agent requesting, the pose of this agent and the name of the task it is requesting. If the IMHuS' scenario includes in its configuration an asynchronous action response corresponding to the requested action, that action is triggered on the humans' side.

At this point, two different things can happen. If there is no human close enough to the *tested robot*, or there is a human but it is not in an idle state, no one will respond to the request and it will be dropped, as Figure 6 shows.

The *tested robot* repeats the request every five seconds until human 2 enters an idle state and responds to the request. Figure 7 shows this moment of the simulation. When an asynchronous action request is triggered, IMHuS periodically checks if one human is able to respond. The requirements for a human agent to respond are to be in the idle state and within a three meters radius from the *tested robot* position when it requested the action. If several human agents are able to accept the task, the task will be assigned to the closest human.

Figure 8 shows the final situation where the *tested robot* has reached the second floor, simulated in the scenario as the room on the other side of the elevator.

6 Conclusions and Future Work

The proposed tool, IMHuS, offers the possibility to create a realistic and challenging simulated environment in which groups of humans can be choreographed

Fig. 6. Request dropped.

Fig. 7. Request accepted by human2.

to evaluate the behaviour of a *tested robot*. Different scenarios can be easily created using an XML configuration file in which social situations can be defined to measure the behaviour of *tested robots* in a replicable environment. Furthermore, human agents are programmed to respond to interactions related to the particular situation of each scenario and their communication with the *tested robot*.

IMHuS's code is available as Open Source in the IMHuS repository. The current version has been implemented for Gazebo simulator, but the design presented in section 3 can be easily migrated to other simulators, such as MORSE or Unity. The tool could be used for benchmarking of competitions such as

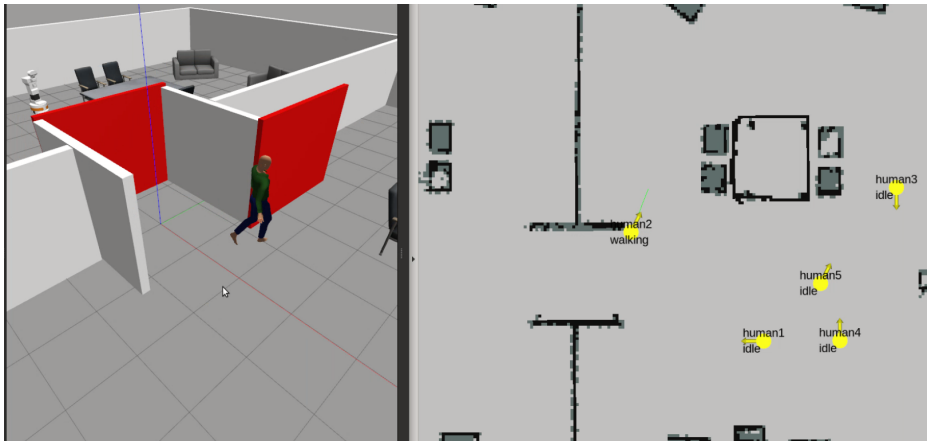

Fig. 8. Final situation. *Tested robot* in second floor.

SciRock or RoboCup as a previous step for the teams before getting to the physical robot challenges. To create a new scenario with IMHuS, all that is needed is a map and the configuration file to choreograph the human agents. The software has been designed to easily support the addition of both new tasks to be performed by the humans, and new interactions between them and the *tested robot*.

One of the ongoing developments is the automatic generation of simulation metrics such as the distance between the robot and the agents, time-to-collision, etc. Another line of work is the extension of basic actions, especially towards the social behaviour of groups of agents. Last, IMHuS is being tested with CoHAN [16], a human-aware robot navigation planner to challenge CoHAN under several human-robot interaction settings. From this, we plan to identify the areas of improvement for human-aware navigation planners and provide a benchmark for testing these planners.

References

1. Anthony Favier, Phani-Teja Singamaneni, R.A.: Simulating intelligent human agents for intricate social robot navigation. In: RSS 2021- Workshop on Social Robot Navigation (Jul 2021)
2. Biswas, A., Wang, A., Silvera, G., Steinfeld, A., Admoni, H.: Socnavbench: A grounded simulation testing framework for evaluating social navigation. CoRR abs/2103.00047 (2021), https://arxiv.org/abs/2103.00047
3. Favier, A., Singamaneni, P.T., Alami, R.: Challenging Human-Aware Robot Navigation with an Intelligent Human Simulation System (Jun 2022), https://hal.laas.fr/hal-03684245, working paper or preprint
4. Gloor, C.: Pedsim: Pedestrian crowd simulation, pedsim.net
5. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Physical review E 51(5), 4282 (1995)

6. Holtz, J., Biswas, J.: Socialgym: A framework for benchmarking social robot navigation (2021)
7. Juliani, A., Berges, V.P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D.: Unity: A general platform for intelligent agents (2020)
8. Kamal M O, A.B.R.: Srin: A new dataset for social robot indoor navigation. Global Journal of Engineering Sciences 4(5) (2020)
9. Kobayashi, Y., Sugimoto, T., Tanaka, K., Yuki Shimomura, F.J.A.G.a.C.H.K., Yabushita, H., Toda, T.: Robot navigation based on predicting of human interaction and its reproducible evaluation in a densely crowded environment. International Journal of Social Robotics pp. 373–387 (2022)
10. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). vol. 3, pp. 2149–2154 vol.3 (2004)
11. Lemaignan, S., Echeverria, G., Karg, M., Mainprice, J., Kirsch, A., Alami, R.: Human-robot interaction in the morse simulator. In: Human-Robot Interaction Conference (HRI'2012) (2012)
12. Liu, Y., Li, S., Li, C., Perez-D'Arpino, C., Savarese, S.: Interactive pedestrian simulation in igibson. In: RSS 2021 - Workshop on Social Robot Navigation (Jul 2021)
13. Manso, L.J., Nuñez, P., Calderita, L.V., Faria, D.R., Bachiller, P.: Socnav1: A dataset to benchmark and learn social navigation conventions. Data 5(1) (2020), https://www.mdpi.com/2306-5729/5/1/7
14. Okal, B., Linder, T.: Pedsim: Pedestrian crowd simulation, github.com/srl-freiburg/pedsim_ros
15. Pellegrin, E.D., Petrick, R.P.: Pdsim: Planning domain simulation with the unity game engine (2021), https://icaps21.icaps-conference.org/demos/demos/387.pdf
16. Phani-Teja Singamaneni, Anthony Favier, R.A.: Human-aware navigation planner for diverse human-robot interaction contexts. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Oct 2021)
17. Rohmer, E., Singh, S.P., Freese, M.: Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013). p. 1321–1326 (2013)
18. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Trans. Graph. 25(3), 1160–1168 (jul 2006), https://doi.org/10.1145/1141911.1142008