HAL
open science

IMHuS: Intelligent Multi-Human Simulator

Olivier Hauterville, Camino Fernández, Phani-Teja Singamaneni, Anthony Favier, Vicente Matellán, Rachid Alami

To cite this version:

Olivier Hauterville, Camino Fernández, Phani-Teja Singamaneni, Anthony Favier, Vicente Matellán, et al.. IMHuS: Intelligent Multi-Human Simulator. IROS2022 Workshop: Artificial Intelligence for Social Robots Interacting with Humans in the Real World, Oct 2022, Kyoto, Japan. hal-03865622

HAL Id: hal-03865622

https://laas.hal.science/hal-03865622

Submitted on 22 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IMHuS: Intelligent Multi-Human Simulator

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

***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 face-to-face. 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. *Index Terms*— RI, simulation, social roboticsRI, simulation, social roboticsH**

I. 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 [1] 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.

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.

II. State of the art

There are several simulators commonly accepted in the robotics research community such as Gazebo [2], Unity [3], Coppelia (V-REP)[4], Morse [5], 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.

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 [6], Pedsim [7], or Menge do not simulate robots. Some adaptations have been proposed to include robots in these crowd simulators. The best known is Pedsim-ROS [8], 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) [9].

MengeROS[10] is a 2-D crowd simulator that integrates multiple robots and a crowd. Menge's crowd scenario includes the static map (walls and obstacles), the simulated people and their initial locations, along with one navigation algorithm (A^* or potential fields) and one collision-avoidance (SFM, ORCA or PedVO) methodology for all the simulated pedestrians. Some severe restrictions in Menge are that all robots have a circular shape and that laser scan and robots' actions are noise-free.

Another recent simulator, SEAN [11] (Social Environment for Autonomous Navigation), is a high-fidelity visual simulator based on Unity that provides only two realistic scenes. The Trial Runner tool included in the toolkit performs the simulations by setting up the scene and assigning start and goal positions for human agents who use Unity's built-in path planning algorithm to generate their routes. However, no tools are provided to define the social behaviour of the human avatars.

A more recent related work is SOCIALGYM [12], 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

[1] RIS group - LAAS (CNRS), Colonel Roche, 31400 Toulouse (France) {Olivier.Hauterville, phani-teja.singamaneni, anthony.favier, Rachid.Alami}@laas.fr,

[2]Robotics Group, Universidad de León, 24007 León (Spain),{cferl,vmato}@unileon.es

[3]ANITI, Univeristé Fédérale Toulouse Midi-Pyrénées, B612, Rue Tarfaya, 31400 Toulouse (France)

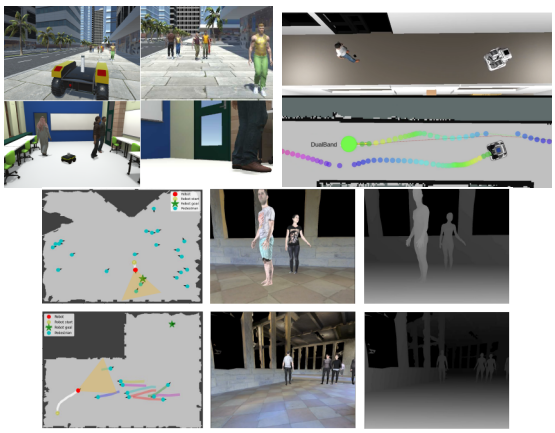

Fig. 1. Collage of different human-robot simulators. From left to right, SEAN, InHuS and SocNavBench.

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) [13] which contains images taken by medium size robots. SocNav1 [14] 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[15] 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.

DeepSocNav [16] proposes a similar approach, starting from the ETH BIWI Walking Pedestrians dataset (RGB images from a bird's eye view of two real scenarios). It recreates the scenes of this dataset in Unity extracting depth images to create first-person views of the trajectories and with an added ability to include new agents. In this case, these systems can go beyond pre-recorded behaviours, but no social behaviours can be defined for the new agents.

Liu et al. [17] 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) [18], 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[19] 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 planning community, there are simulators as PDSim[20], 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.

III. 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.

The definition of the choreography 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.

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*: 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.

IV. Implementation

The prototype has been implemented as a new version of the InHus tool ([21], [18]) 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 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 2 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).

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

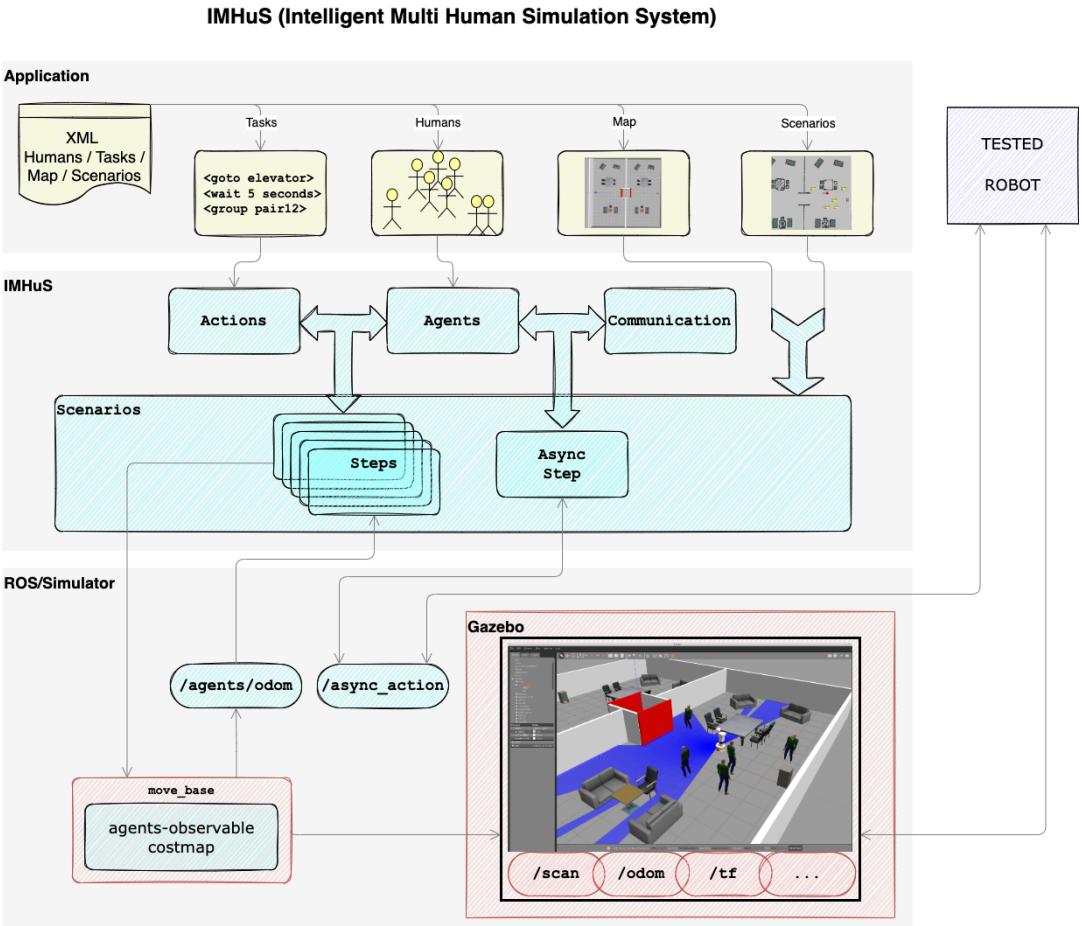

Fig. 2. Structure of the new IMHuS tool

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.

V. 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). In this scenario, the robot has to be able to take an elevator in the presence of people.

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 3 shows the initial situation where the

tested robot is approaching the elevator, human 2 is walking and the rest of humans are idle.

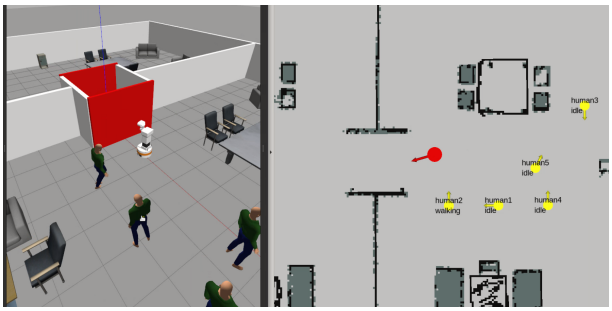

Fig. 3. Initial situation.

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 2). 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 4 shows.

Fig. 4. Request dropped.

The *tested robot* repeats the request every five seconds until human 2 enters an idle state and responds to the request. Figure 5 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 6 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.

Fig. 5. Request accepted by human2.

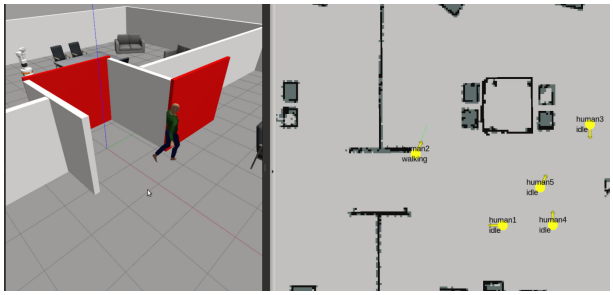

Fig. 6. Final situation. *Tested robot* in second floor.

VI. 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 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 III can be easily migrated to other simulators, such as MORSE or Unity. The tool could be used for benchmarking of competitions such as SciRoc 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 [19], 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.

ACKNOWLEDGMENT

This work has been partially funded by the Kingdom of Spain under grant PID2021-126592OB-C21.

REFERENCES

[1] Y. Kobayashi, T. Sugimoto, K. Tanaka, F. J. A. G. a. C. H. K. Yuki Shimomura, H. Yabushita, and T. Toda, "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.

[2] N. Koenig and A. Howard, "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, 2004, pp. 2149–2154 vol.3.

[3] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," 2020.

[4] E. Rohmer, S. P. Singh, and M. Freese, "CoppeliaSim (formerly V-REP): a versatile and scalable robot simulation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, 2013, p. 1321–1326.

[5] S. Lemaignan, G. Echeverria, M. Karg, J. Mainprice, A. Kirsch, and R. Alami, "Human-robot interaction in the MORSE simulator," in *Human-Robot Interaction Conference (HRI'2012)*, 2012.

[6] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Trans. Graph.*, vol. 25, no. 3, p. 1160–1168, jul 2006. [Online]. Available: https://doi.org/10.1145/1141911.1142008

[7] C. Gloor, "PedSim: Pedestrian crowd simulation," pedsim.net.

[8] B. Okal and T. Linder, "PedSim: Pedestrian crowd simulation," github.com/srl-freiburg/pedsim_ros.

[9] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[10] A. Aroor, S. L. Epstein, and R. Korpan, "MengeROS: a crowd simulation tool for autonomous robot navigation," *ArXiv*, vol. abs/1801.08823, 2017.

[11] N. Tsoi, M. Hussein, J. Espinoza, X. Ruiz, and M. Vázquez, "SEAN: Social environment for autonomous navigation," in *Proceedings of the 8th International Conference on Human-Agent Interaction*, ser. HAI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 281–283. [Online]. Available: https://doi.org/10.1145/3406499.3418760

[12] J. Holtz and J. Biswas, "SOCIALGYM: A framework for benchmarking social robot navigation," 2021.

[13] A. B. R. Kamal M O, "SRIN: A new dataset for social robot indoor navigation." *Global Journal of Engineering Sciences*, vol. 4, no. 5, 2020.

[14] L. J. Manso, P. Nuñez, L. V. Calderita, D. R. Faria, and P. Bachiller, "SocNav1: A dataset to benchmark and learn social navigation conventions," *Data*, vol. 5, no. 1, 2020. [Online]. Available: https://www.mdpi.com/2306-5729/5/1/7

[15] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, "SocNavBench: A grounded simulation testing framework for evaluating social navigation," *CoRR*, vol. abs/2103.00047, 2021. [Online]. Available: https://arxiv.org/abs/2103.00047

[16] J. P. de Vicente and A. Soto, "DeepSocNav: Social navigation by imitating human behaviors," in *RSS Workshop on Social Robot Navigation*, 2021.

[17] Y. Liu, S. Li, C. Li, C. Perez-D'Arpino, and S. Savarese, "Interactive pedestrian simulation in iGibson," in *RSS 2021 - Workshop on Social Robot Navigation*, Jul 2021.

[18] A. Favier, P.-T. Singamaneni, and R. Alami, "Challenging Human-Aware Robot Navigation with an Intelligent Human Simulation System," June 2022, working paper or preprint. [Online]. Available: https://hal.laas.fr/hal-03684245

[19] R. A. Phani-Teja Singamaneni, Anthony Favier, "Human-aware navigation planner for diverse human-robot interaction contexts," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2021.

[20] E. D. Pellegrin and R. P. Petrick, "PDSim: Planning domain simulation with the unity game engine," 2021, https://icaps21.icaps-conference.org/demos/demos/387.pdf.

[21] R. A. Anthony Favier, Phani-Teja Singamaneni, "Simulating intelligent human agents for intricate social robot navigation," in *RSS 2021-Workshop on Social Robot Navigation*, Jul 2021.