Challenging Human-Aware Robot Navigation with an Intelligent Human Simulation System

Anthony Favier, Phani Teja Singamaneni, Rachid Alami

Challenging Human-Aware Robot Navigation with an Intelligent Human Simulation System

Anthony Favier[1,2], Phani Teja Singamaneni[1], and Rachid Alami[1,2] *

[1] LAAS-CNRS, Université de Toulouse, CNRS, INSA, UPS, Toulouse, France
[2] Artificial and Natural Intelligence Toulouse Institute (ANITI)
{anthony.favier, ptsingaman, rachid.alami}@laas.fr

Abstract. The final validation of human-aware social robot navigation schemes requires testing and evaluation with real-life humans. During the prototyping and developmental phase, however, experiments with real humans for debugging and testing may not be a feasible option. Simulators can help overcome this issue, but the lack of available intelligent human avatars with rational behaviors restricts the testing to simple scenarios. In this regard, we propose a system for simulating an autonomous intelligent human agent specifically designed to act and interact with the robot navigating in a simulated environment. Further, the proposed system provides interactive GUI modules to run repeatable scenarios and visualize the interaction data. We conduct a series of experiments that show how the proposed **In**telligent **Hu**man **S**imulation (InHuS) system can help tune and debug social navigation better.

Keywords: Human Simulation · Intelligent Human Avatar · Human-aware robot navigation

1 Introduction

Significant efforts are being dedicated today toward the development of robots that interact, assist or work side-by-side with humans. However, people working in the field of human-robot interactions (HRI) face constraining issues while testing and evaluating their systems. Apart from being mandatory to validate mature systems, experimenting using real humans and robots is burdensome for many reasons: they are slow, hardly repeatable, expensive, etc. Moreover, the system needs to be run extensively for debugging, tuning, and testing various options before it reaches maturity. Doing so with real-life experiments is generally a long and tiresome process where colleagues in the lab and volunteers spend unproductive hours, if not days, interacting with a robot running a system under debugging. Moreover, such methods require exclusive physical access to the robot, a place to run the tests, and cannot run faster than real-time or be parallelized.

* This work has been partially funded by the Agence Nationale de la Recherche through the ANITI ANR-19-PI3A-0004 grant and by the Horizon Europe Framework Programme through the euROBIN Grant 101070596.

Simulations are well suited for such tasks as they allow working without a real robot or a physical space. Further, they allow multiple tests to run simultaneously and faster than in real life. The simulated environment for the tests can be changed very easily in contrast to real-life tests. However, simulating realistic human behaviors and interactions is tough, which could make simulations unreliable. Consequently, HRI researchers face some difficulties such as: "How to test repeatedly and intensively their systems even when they are not sufficiently robust?" and "How to challenge their systems in a large variety of environments and situations?". Therefore, there is a need for an "intelligent artificial human" that would help challenge the robot's interactive and decisional abilities.

1.1 Human simulations in human-aware robot navigation

Being a part of HRI, the field of human-aware social robot navigation inherits all these limitations. One way that is employed for simulating an intelligent avatar in this field is to manually control the human avatar, in real-time [2]. This can be done using a variety of devices like a gaming controller, keyboard, or motion capture. Such approaches require a real human operator only focused on controlling the avatar, which brings back some of the mentioned limitations like human fatigue. Autonomous human avatars on the other hand seem to offer an adequate solution to this, but they often lack intelligence and rationality.

Most of the current autonomous avatars available are either scripted or reactive. A scripted avatar executes a series of predefined actions, like following a fixed path, without being reactive to its environment which makes interactions very limited. Reactive agents use models like social force [4] or optimal reciprocal collision avoidance (ORCA) [15]. These systems are highly scalable and can simulate groups or even crowds composed of a large number of agents. MengeROS [1] and PedSim_ROS[3] are some examples. Despite their number, the generated agents usually fail in intricate social scenarios. Some recent works like VirtualHome [10] and SEAN [13, 14] discuss simulating human agents to challenge robot systems, but the navigation of the agents in these systems is still based on reactive-only models. The work presented in [17] proposes a learning-based method to generate more realistic pedestrian navigation. This ongoing work shows an interesting navigation behavior like waiting and letting the other agent pass embedded in iGibson [11] simulator. However, this work is more focused on motion generation than decision-making to solve conflicts.

We propose the InHuS System to contribute to the lack of intelligent and rational human agents with conflict-resolution skills to challenge the human-aware robot navigation systems. Our contribution includes 1) an intelligent human agent controller, 2) a high-level interface to control the simulated agents, and 3) a GUI to plot execution data and metrics for evaluating the interaction. Such a system could help people working in the field of human-aware robot navigation to test and debug their schemes. Our system is designed to run, analyze and evaluate repeatable and long navigation scenarios involving a robot and an

[3] https://github.com/srl-freiburg/pedsim_ros

autonomous reactive and rational avatar. This work focuses on intricate and narrow scenarios where, in addition to being reactive, rational decisions should be taken in order to solve the conflicts occurring. Note that our contribution is focused on navigation decision-making and not the motion generation part. Throughout this paper, we use the term 'rational' in a meaning close to Goal Reasoning [16, 5], i.e., the ability of autonomous agents which can dynamically reason about and adjust their goals. It enables the agents to adapt intelligently to changing conditions and unexpected events, allowing them to address a wide variety of complex situations.

2 InHuS System

The InHuS System[4] works along with a human operator, a chosen simulator, and the challenged robot controller as depicted in Fig. 1. The system is mainly implemented using ROS. The InHuS System is three-sided. First, the system comes with a high-level interface called Boss that helps to manage the simulated agents. Secondly, there is the main part which is the intelligent human avatar controller itself, called InHuS. Finally, a GUI provides an interactive visualization of the data and metrics computed by InHuS during execution that can help to evaluate interactions. We present below some details for each component.

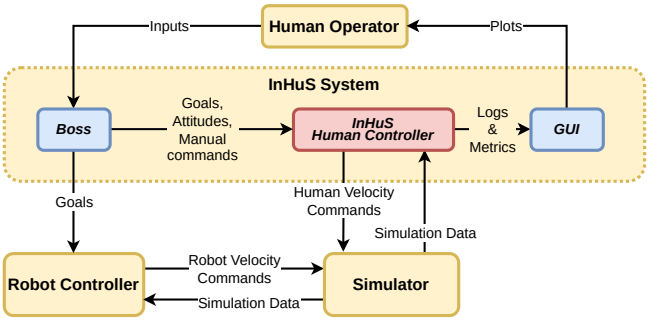

Fig. 1. The InHuS System interacts with three external systems: the simulator, the robot controller, and a human operator. Our system is separated into three parts: the Boss high-level interface gathering inputs from the human operator, InHuS which is the actual human controller, and a GUI to plot the metrics and other data produced.

2.1 Boss

For the human operator to easily control the simulated agents and run repeatable scenarios, we provide a simple graphical user interface component called Boss. Predefined or manually entered goals can be sent to the human, the robot, or both. Goals are by default considered as "Pose goals" that only require one

[4] https://github.com/AnthonyFavier/InHuS_Social_Navigation

navigation action to be achieved. However, the human agent (only) can handle "Compound goals" that need a specified sequence of navigation and waiting for actions to be achieved. This type of compound goal is useful to emulate more complex activities. For example, "Make coffee" could be described as a sequence of three actions: nav(coffeeMachine), wait($15s$), nav(myOffice).

The Boss allows defining scenarios with start positions and goals for each agent to repeatedly generate the same situation. Running a scenario consists of first sending each agent to their respective starting position. Then, the corresponding goals are sent to the human and the robot. A delay can be specified while starting the scenario to delay either the robot's or the human's goal. This is very useful to adjust the timing of a specific situation or conflict. The Boss can also put an agent in "endless" mode where the agent continuously gets a new goal from a given list after completing one.

Each navigation action can specify a radius for the "Pose goal", within which a new "Pose goal" is randomly sampled. This mechanism adds randomness to the execution and diversifies the situations encountered, especially in the "endless" mode. Setting the radius to zero disables the randomization and selects the given goal.

All the goals, scenarios, and endless goal sequences are defined using an XML format. Hence, defining new goals or scenarios is straightforward. There is an XML goal file associated with each map/environment. Thus, it is easy to switch between environments since the corresponding goal file is automatically loaded.

2.2 InHuS

The macro component InHuS is mainly in charge of controlling the avatar and generating rational behaviors. InHuS itself is made of several components as depicted in Fig. 2. However, three components, namely HumanBehaviorModel, Supervisor, and GeometricPlanner, constitute the major functional part of InHuS. We discuss each of these major components in detail.

HumanBehaviorModel: The HumanBehaviorModel is responsible for most of the rational behavior of the agent. The first role of this component is to manage the goals. Goals can either be received from the Boss component or generated

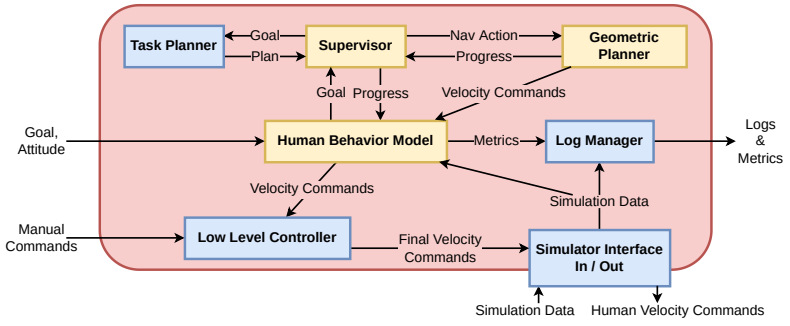

Fig. 2. The human controller InHuS with its components and subsystems.

by the HumanBehaviorModel using the same XML file as the Boss. When a goal is selected, it is sent to the Supervisor for execution.

This component is also responsible for detecting and handling navigation conflicts. Currently, the kind of navigation conflict handled by InHuS is path blockage (e.g. another agent standing in a doorway). While the human agent is navigating, a path to the goal is calculated at regular intervals using Dijkstra's algorithm, and its length is tracked to detect such conflicts. If the tracked path length increases significantly or the path ceases to exist, it could mean that another agent is blocking either the only possible way or the shortest way. When such situations are detected, the plan execution is temporarily suspended, and the agent performs an approach action to get close to the blocking location. This shows the agent's intention to move in a specific direction and might induce the blocking agent to react and clear the way. Eventually, once the avatar is at a specified distance of the blocking location, here set to 1.5 m, the agent stops its approach and actively waits for the path to be cleared.

To generate a lot of different and specific situations, we created what we call *Attitudes*. They are operating modes affecting both goal decisions and reactions toward the other agents. One can activate them through the Boss to generate diversified behaviors of the agent. Some of the *Attitudes* currently implemented in InHuS consist of: 1) randomly picking a new goal, like someone suddenly changing their mind, 2) harassing the robot by constantly going in front of it, like a child would do [8], and 3) stopping close to the robot and looking at it for a few seconds before resuming its goal which emulates a curious behavior.

The final purpose of this component is to build the perception of the human agent based on the map and information about the other agents from the simulator. We build the perception by directly accessing the simulation data rather than adding simulated sensors to the human avatar. Using this perception, we compute the visibility of the human agent and then update the human's knowledge about the robot's position and speed.

Supervisor: The Supervisor is a central component as it coordinates different components to execute the plan and achieve the current goal. When the Supervisor receives a goal from the HumanBehaviorModel, it requests the TaskPlanner component a plan to achieve the goal. For now, the plan generation is quite simplistic. For a "Pose goal", a plan filled with a single navigation action is generated. For a "Compound goal", the sequence of navigation and waiting actions is extracted from the XML goal file and the plan is populated. Despite the simplistic plan generation, this architecture handles complex goals that require several steps to be achieved and emulate human activities.

The execution of each action of the plan is then supervised by the Supervisor by sending requests to other components. When a navigation action needs to be performed, the Supervisor starts by sampling a random position if the given action radius is not zero. Then, it requests the GeometricPlanner to plan for the target position without considering other agents initially. This way, the avatar starts following the shortest path, and we initialize the conflict detection. After

this, the system starts to consider the other agents, and the Supervisor periodically requests the HumanBehaviorModel component to check for potential navigation conflicts. The Supervisor can suspend and resume the plan execution at any time, which can be used to resolve the detected conflicts or to generate specific reactions like the *Attitudes*.

GeometricPlanner: The last major component is the GeometricPlanner. This motion planner component receives a target position to reach from the Supervisor and generates velocity commands to make the avatar move. This component defines how the agent moves around and adapts its velocity to the other agents in the scene. Since the system is implemented in ROS, we use the standard ROS navigation stack for the GeometricPlanner.

The planner used in InHuS is a publicly available human-aware navigation planner called CoHAN [12]. It is built over the ROS navigation stack and uses a local planner based on a modified version of the timed elastic band with human-aware properties. We benefit from the high-level decision-making of InHuS and the enhanced local navigation of CoHAN with trajectory predictions. Moreover, CoHAN is highly tunable which helps to generate different agent behaviors.

2.3 Logs, metrics and GUI

The InHuS system logs the execution data like the positions and speeds of the agents along with some computed metrics. All the logged data is sent to the GUI component, which generates interactive plots. These plots can help evaluate the interaction and thus the performance of the given robot controller. The snapshot of the GUI shown in Fig. 3 shows two kinds of visualizations. On the right side, there is a colored visualization of the paths taken by each agent. These paths are colored over time according to a corresponding legend that helps estimate

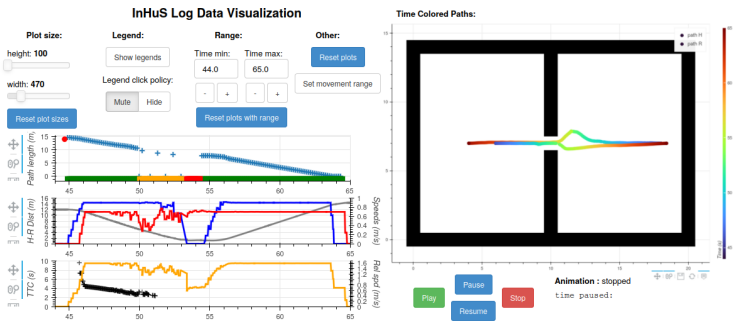

Fig. 3. Overview of the GUI interface which is organized as follows. On the right side are shown the paths taken by the agents and colored over time. On the left side, several metrics and data produced by InHuS are plotted over time on graphs. Additional widgets help to configure the plots.

an agent's position at a specific moment. The left side is composed of several plots showing some computed metrics over time. The first plot is about conflict detection and solving. It shows the length of the path to the goal computed when checking for conflicts. Without any conflict, the path length should decrease linearly over time. If it's not the case, the avatar has been disturbed during the navigation. This plot also shows the conflict state of the agent: Nominal (no conflict), Approach (conflict detected), Blocked (stopped and waiting). The subsequent plots show over time the speeds of each agent, their relative speed, the distance separating them, and a metric called time to collision (TTC). This metric estimates the time remaining before the agents collide with their current velocities. We can argue that TTC corresponds to a "threat feeling" since a low TTC value corresponds to a high threat of collision. Hence, social robots should be tuned to not exceed a minimum TTC value to make humans more comfortable.

3 Experiments

In this section, we show some results through a set of experiments to highlight how our system can help challenge human-aware robot navigation systems. First, we discuss the limits of reactive-only systems to strengthen the need for rational avatars. Then, we present how our system effectively challenges robot navigation systems and we interpret the corresponding plots. Next, we show how the InHuS System can compare the human-aware performances of two different robot controllers. Finally, we present additional experiments showing the diverse behaviors that can be produced using the *Attitudes*, and how "long runs" can benefit the development of a robot controller.

3.1 Limits of reactive-only agents

Most of the current human agent simulations used by the social navigation community rely either on the social force model or ORCA. In order to highlight the limitations of such approaches, we present results obtained with a PedSim_ROS (or simply PedSim) agent. PedSim is a pedestrian simulator that uses the social force model. It is very efficient for generating crowds to test robot navigation. However, at the individual level, the simulated agents are purely reactive and have no decisional abilities like most pedestrian simulators.

Consider the doorway scene shown in the left part of Fig. 4. Both agents have to cross a narrow opening. Here, the robot is blocking the way that the human agent intends to cross. The PedSim agent approaches the robot and tries to push itself through, but it fails due to a very high value of social force. The agent never stops moving and tends to go right or left along the wall before wiggling again just in front of the robot. This confusing behavior can make the agent's intentions unclear to the robot planner. The narrow corridor scenario, shown in the right part of Fig. 4, also exposes some limits. In this scene, there is not enough space for the agents to cross each other, and the only solution is for

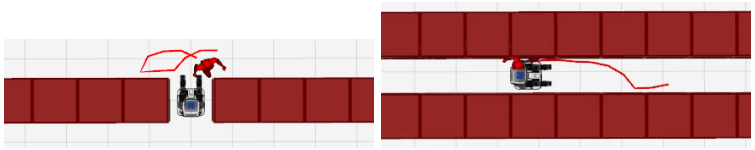

Fig. 4. In the doorway scenario (left), the reactive-only (Pedsim) agent never stops moving while trying to go through the robot even though its path is blocked. In the narrow corridor scenario (right), the agent squeezes itself between the wall and the robot colliding with both.

one of them to back off. Here the path is blocked by the static robot. The PedSim agent slowly gets closer and closer to the robot before squeezing itself between the wall and the robot. For some reason here the social forces allowed the agent to pass, unlike the previous example. It highlights that the PedSim agent doesn't use a defined hitbox or footprint for the agent and relies only on repulsive social forces to prevent collisions. This lack of defined collision shapes makes the agent temporarily pass through the walls and other agents. As a consequence, it breaks many intricate scenarios where a rational decision should be taken and results in unrealistic situations. Despite being efficient for large spaces or crowds, based on the above observation, we can state that in intricate scenarios such approaches can lead to confusing and even unrealistic behaviors.

3.2 Interpretation of plots with human-aware planner

The InHuS System is able to generate challenging situations and associated logs to allow further evaluation. Here, we present one such conflict and a detailed interpretation of the corresponding plots. The plots were produced while challenging the CoHAN system in the doorway scenario.

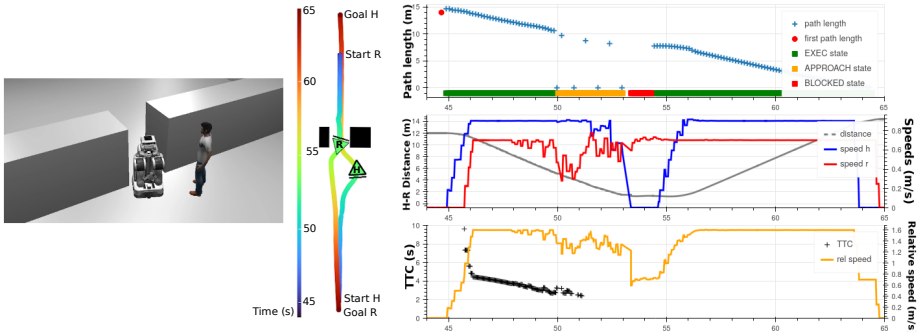

Fig. 5. A condensed view of the InHuS GUI and MORSE simulator for the doorway scenario with a robot running the CoHAN planner. Several plots depict the detection and resolution of the conflict created.

The robot starts closer to the opening and enters the doorway first. The execution can be analyzed with the metric plots and the time-colored paths of the agents in Fig. 5. We notice that the robot's speed (red line on the second graph) goes down around 50 s as it is entering the doorway and creating a conflict. The conflict is detected by InHuS (zero path length = no path), and the agent switches to the approach state (green to the yellow line on the first graph). The non-zero path length in the approach state corresponds to how the approach is performed. In order to keep moving despite the blocked path, the GeometricPlanner is requested at a defined frequency to plan without considering the robot (all non-zero path length). In between these requests, to check if the path is still blocked, the conflict detection plans while considering the robot (zero path length). When the avatar is at a predefined distance from the blocking robot around 53 s, it switches to the blocked state (red line) to stop and wait for the path to be cleared. Further, the time-colored paths show that the GeometricPlanner made the avatar move aside while approaching to avoid blocking the robot. As a result, the agents were no longer moving towards each other, and thus, there was no longer any collision threat (no TTC values). When there is no more collision threat, around 51 s, the robot's speed starts to increase again. Such behavior is a good sign of human-aware properties and might increase human comfort.

From the plots produced by our system, a lot of useful information can be extracted for improving or evaluating the social robot planner's performance like a) finding ways to decrease the blocked state time for the human, b) maintaining a particular threshold for TTC, c) slowing down near the human, or waiting for the human to cross the door without blocking.

3.3 Quantitative comparison between two robot controllers

Our system can be used to run similar scenarios repetitively to produce robust metric values. These values can help to evaluate the human-aware performances of a given robot controller. To show this, we present a comparison between two different robot controllers. The first one is again the CoHAN system, and the second one is the Simple Move Base (or SMB). It uses the *teb_local_planner* and the ROS navigation stack with default parameters. We just add an additional process to consider the human agent as a static obstacle to avoid it, so it is not human-aware. Therefore, we should be able to notice a clear difference through the metrics computed by our system. For this comparison, we used three different scenarios: 1) The doorway scenario where the agents have to cross a narrow opening, 2) the corridor scenario where the agents cross each other with just enough space, and 3) the open space where they cross each other without any environmental constraints. We performed 10 repetitions of each scenario for each robot controller. For each set of 10 repetitions, we extracted the mean values of three different metrics and presented them in Table 1. The metrics are the following. First, the time to goal (TTG) is the time taken by the avatar to reach its goal. Second, the minimum distance between the robot and the human (Min HRDist). And, the minimum time to collision (TTC). Intuitively, we want the

TTG to be as small as possible, the minimum HRDist to be as high as possible, and since a low TTC value represents a collision threat, we want the min TTC to be as high as possible.

Table 1. Mean values of three InHuS metrics over 10 repetitions in three different scenarios and with two different robot controllers. Bold values indicate when the corresponding robot controller has better performance than the other.

	CoHAN			SMB		
Experiment	$TTG(s)$	$Min\ Dist(m)$	$Min\ TTC(s)$	$TTG(s)$	$Min\ Dist(m)$	$Min\ TTC(s)$
Doorway	18.38	**2.32**	**1.33**	**18.26**	2.23	1.16
Corridor	**16.34**	**2.06**	**1.03**	17.05	1.59	0.81
Open space	**9.55**	**2.52**	**1.61**	11.01	2.34	1.18

At first glance, we see in Table 1 that almost all CoHAN values are better than SMB values. Due to the nature of the doorway environment, the execution of the scenario is quite constrained which explains why the values are not too different between the two controllers. However, we notice anyway that, compared to SMB, the CoHAN planner tends to keep a greater distance between the agents and a greater TTC (lower threat of collision). The time to goal of CoHAN is slightly higher because the robot slows down when crossing and moving in the direction of the human. Thus, it is the price to pay in this scenario to maintain adequate TTC values.

In the corridor scenario, The SMB robot tends to wait until the last moment to move aside, which is threatening. On the other hand, the CoHAN robot proactively moves to one side of the corridor. As a consequence, it leaves more space for humans and reduces the threat of collision, which is visible in the obtained values. Also, this pro-activity has the effect of smoothing the trajectory of the avatar, which makes this last one reach its goal faster.

Finally, the open space scenario is a bit similar to the previous one. The SMB robot waits until the last moment to avoid the human, which puts the load of the avoidance maneuver on the human. As a result, the human has to move aside which extends the duration to reach the goal. Also, due to the same behavior, the SMB robot is on average closer to the avatar and more threatening. Since the CoHAN robot moved again aside early, its metric values are noticeably better than SMB.

In summary, the human-aware behavior of the CoHAN controller was captured through significant value differences in the computed metrics compared to a non-human-aware robot controller. This implies that our system can help evaluate and compare human-aware robot controllers.

3.4 Generating different behaviors with Attitudes

By activating *Attitudes*, InHuS is capable of producing more complex behaviors to diversify the conflicts and challenges imposed on the robot. We present the time-colored paths for the execution of two *Attitudes* : *Harass* and *StopAndLook*

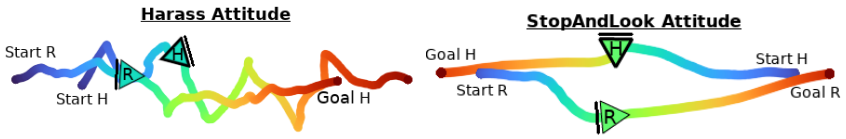

Fig. 6. Behaviors obtained by activating the *Harass* and *StopAndLook Attitudes*. With *Harass*, the human is always in front of the robot. With *StopAndLook*, when close to the robot, the human stops to look at it for a few seconds.

in Fig. 6. Concerning the *Harass Attitude*, by paying attention to the colors, we see that the human is always in front of the robot that continuously tries to avoid the harassing agent causing erratic movements. The robot should be able to detect such non-cooperative behavior from humans and act accordingly. On the same figure we see the execution of the *StopAndLook Attitude*. The color discontinuity behind the human marker shows how the human suspended its goal to stop and briefly stare at the robot before moving again. A robot not pro-active enough could be disturbed by the sudden stop of the human, which could be a situation of interest to handle.

3.5 Long run scenarios

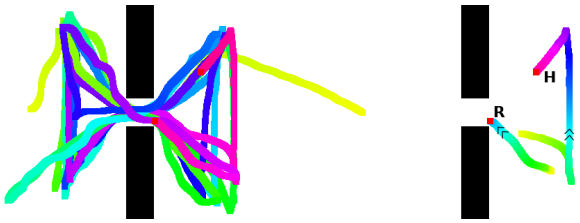

Fig. 7. Execution of the long run scenario using the TDP robot planner and InHuS. We see the complete set of time-colored paths on the left. On the right, the same path is cut, around the moment when the robot got stuck in the wall.

The proposed system can help test the stability and robustness of the robot planner by conducting long randomized runs. Indeed, thanks to the Boss component, possibly randomized goals can be sent autonomously to the agents. This can generate unexpected situations and conflicts that can be of interest. Fig. 7 depicts such a test conducted with InHuS and a human-aware robot planner from Kollmitz et. al. [7] here referred to as TDP. The agents were made to endlessly loop over four goal-positions (each with a 1 m radius) in reverse order to create as many conflicts as possible. After 3 minutes, the robot got stuck in the wall of the doorway, indefinitely blocking the path for the human. In addition to highlighting problematic situations where the robot doesn't act as expected, long runs can expose low-level issues like unexpected crashes or memory leaks.

4 Discussion

Although InHuS provides an autonomous human agent, if needed, the agent can be controlled manually. We do not yet provide a handy controller, but velocity commands generated by any means can be sent to the Boss component to control the human. This extends the usability of InHuS as one can use scripted trajectories or motion capture to control the human agent in the simulator.

The proposed system interacts with an external simulator and robot controller. Since the system is majorly implemented using ROS, switching from one simulator to another is straightforward if it has a ROS interface. InHuS has specific components to abstract the simulation data format. Thus, just by slightly editing these components we were already able to run InHuS on three different simulators: MORSE [3], Stage[5] and Gazebo [6]. Furthermore, any robot controller using the ROS Navigation Stack can be directly used with InHuS.

Simulating intelligent human avatars is a novel field and only few works apart from ours try to address this limitation. A similar work in ROS2 was recently presented in [9]. It is clear that the idea of intelligent human agents is of interest to the community, and it is a necessity to test social navigation effectively. Like any other system, InHuS has limitations too. We claim to generate only reactive and somewhat rational behavior, which is still far from natural or realistic human behavior. We currently handle scenarios with two agents only, the human and the robot. We can run scenarios with other human agents, but they will be treated like robots.

5 Conclusion and Future work

Human-aware social robot navigation is rapidly growing, but the community lacks good human agent simulations to test and debug their systems. The existing reactive approaches offer only limited testing. Through the InHuS system, we proposed a pertinent approach to address this issue. We showed that our system could generate conflicting situations that need resolution by making rational choices. Moreover, all the metrics and data recorded during execution and their visual plots allow us to evaluate the interaction and behavior of the robot. With such evaluation, we showed that we could compare different robot controllers. InHuS can also generate various tunable behaviors that can diversify the situations and conflicts imposed on the robot, and thus, it helps to debug and tune the system. Long runs provide additional potential ways to improve the system.

We already use this system to test our human-aware motion planners and refine them over time using the tests conducted. In the future, we plan to integrate situation detection and diagnosis in the long run to catch the problematic situations that need to be analyzed afterwards to tune, refine or extend a given planner. We also plan to handle scenarios with more human agents, like groups and crowds, using a combination of intelligent and reactive agents.

[5] https://github.com/ros-simulation/stage_ros

References

1. Aroor, A., Epstein, S.L., Korpan, R.: MengeROS: a Crowd Simulation Tool for Autonomous Robot Navigation. arXiv:1801.08823 [cs] (Jan 2018)
2. Echeverria, G., Lemaignan, S., Degroote, A.e.A.: Simulating Complex Robotic Scenarios with MORSE. In: Simulation, Modeling, and Programming for Autonomous Robots, vol. 7628, pp. 197–208. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34327-8_20
3. Echeverria, G., Lassabe, N., Degroote, A., Lemaignan, S.: Modular open robots simulation engine: Morse. In: 2011 ieee international conference on robotics and automation. pp. 46–51. IEEE (2011)
4. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Physical review E **51**(5) (1995)
5. Johnson, B., Floyd, M.W., Coman, A., Wilson, M.A., Aha, D.W.: Goal Reasoning and Trusted Autonomy. In: Foundations of Trusted Autonomy, vol. 117. Springer (2018)
6. 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. IEEE (2004)
7. Kollmitz, M., Hsiao, K., Gaa, J., Burgard, W.: Time dependent planning on a layered social cost map for human-aware robot navigation. In: 2015 European Conference on Mobile Robots (ECMR). pp. 1–6. IEEE, Lincoln, United Kingdom (Sep 2015). https://doi.org/10.1109/ECMR.2015.7324184
8. Nomura, T., Kanda, T., Kidokoro, H., Suehiro, Y., Yamada, S.: Why do children abuse robots? Interaction Studies **17**(3), 347–369 (2016)
9. Pérez-Higueras, N., Otero, R., Caballero, F., Merino, L.: Hunavsim: A ros2 human navigation simulator for benchmarking human-aware robot navigation. arXiv preprint (2023)
10. Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., Torralba, A.: VirtualHome: Simulating Household Activities Via Programs. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8494–8502. IEEE, Salt Lake City, UT (Jun 2018). https://doi.org/10.1109/CVPR.2018.00886
11. Shen, B., Xia, F., Li, C., Martın-Martın, R., Fan, L., Wang, G., Buch, S., D'Arpino, C., Srivastava, S., Tchapmi, L.P., Vainio, K., Fei-Fei, L., Savarese, S.: igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. arXiv preprint (2020)
12. Singamaneni, P.T., Favier, A., Alami, R.: Human-aware navigation planner for diverse human-robot ineraction contexts. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2021)
13. Tsoi, N., Hussein, M., Espinoza, J., Ruiz, X., Vázquez, M.: Sean: Social environment for autonomous navigation. In: Proceedings of the 8th International Conference on Human-Agent Interaction (HAI) (November 2020)
14. Tsoi, N., Xiang, A., Yu, P., Sohn, S.S., Schwartz, G., Ramesh, S., Hussein, M., Gupta, A.W., Kapadia, M., Vázquez, M.: Sean 2.0: Formalizing and generating social situations for robot navigation. IEEE Robotics and Automation Letters **7**, 11047–11054 (2022)
15. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics research, pp. 3–19. Springer (2011)

16. Vattam, S., Klenk, M.E., Molineaux, M., Aha, D.W.: Breadth of approaches to goal reasoning: A research survey. In: Annual Conference on Advances in Cognitive Systems: Workshop on Goal Reasoning (2013)
17. Yige, L., Siyun, L., Chengshu, L., Claudia, P.D., Silvio, S.: Interactive pedestrian simulation in igibson. RSS Workshop on Social Robot Navigation (2021)