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

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

***Abstract*— Human-aware and so-called social navigation abilities of a robot need to be tested and evaluated in real-life experiments with real humans. Such tests are mandatory to validate a mature system. But they are heavy to set up and can become tiresome when debugging the early stages of a system. Simulation could help solve this issue, but the lack of available intelligent human avatars that can exhibit a rational behavior restricts this to simple scenarios. To address this issue, we propose a system providing first an autonomous intelligent human agent specifically designed to act and interact with a robot navigating in a simulated environment. Then, the system provides a GUI to visualize the paths taken by the agents as well as the produced metrics in order to help evaluate the interaction. Moreover, the system also provides a high-level interface to control the agents and run repeatable scenarios. We show through a set of experiments how the proposed system can help answer the lack of intelligent avatars for tuning and debugging social navigation systems before their final evaluation with real humans.**

I. Introduction

Significant efforts are being dedicated today towards 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. However, 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. Simulation would be well suited for such tasks but 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 sufficiently 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.

The field of human-aware or social robot navigation is also subject to the same issues and lacks intelligent avatars to challenge their systems. Researchers mostly rely on systems using reactive models like social force [1] or optimal reciprocal collision avoidance (ORCA) [2] to test and evaluate their planners. These systems have the benefit of often being scalable and thus can provide groups or even crowds composed of a large number of agents. However, despite their number, the generated agents usually fail in intricate or narrow scenarios. Some recent works like VirtualHome [3] and SEAN [4] discuss simulating human agents to challenge robot systems, but the navigation of the agents in these systems is still based on reactive-only models. Another recent work presented in [5], proposes to generate more realistic pedestrian navigation using a learning-based method. This ongoing work shows an interesting navigation behavior like waiting and letting the other agent pass embedded in iGibson [6] simulator. However, this work is more focused on motion generation than decision-making to solve conflicts. Also, as of today, it is not publicly available.

We propose the InHuS System to contribute to the lack of intelligent human agents and help 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. Our system can create repeatable and challenging situations to evaluate the human awareness of a given robot controller. Thus, we believe that our contribution could be used as an efficient tool to debug, test, or tune robot social navigation systems before being deployed on real robots.

In this paper, we use the term 'rational' in a meaning close to Goal Reasoning [7], [8], 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.

The rest of the paper is structured as follows. Section II outlines the motivations behind this project by exposing the constraints of real-life tests and the limitations of simulated humans in social navigation. Section III presents the InHuS System and presents explanations on its several features. Next, section IV presents a variety of results through a set of experiments that show how our system can help tune, test, and debug social navigation systems. Then, section V provides discussions about some specific features of the system and its current limitations, followed by section VI which presents conclusions and possible future work.

[1] With LAAS-CNRS, Universite de Toulouse, CNRS, Toulouse, France
[2] With Artificial and Natural Intelligence Toulouse Institute (ANITI)
* Contacts : {ptsingaman, anthony.favier, rachid.alami}@laas.fr
** This work has been partially funded by the Agence Nationale de la Recherche through the ANITI ANR-19-PI3A-0004 grant

II. Why Human Simulation ?

Testing and evaluating the performance of robot software for HRI is challenging [9]. Due to the nature of the HRI field itself, a human is essential for experiments. This is true even in the case of social navigation. Experiments can be classified into three categories according to the type of human used: Real-life human, Operator-controlled simulated human, Autonomous avatar. Each category has its limitations which we discuss below with related works.

A. Real-life human

Tests in the real world are burdensome to conduct. They are slow to set up, and neither can be run faster than real-time nor can be parallelized. They are also hardly controllable and reproducible and are limited by human fatigue. Thus, we can hardly run tests in high numbers or for a long period. Lastly, more than requiring real humans, such a method requires exclusive physical access to the robot and a place to run the tests. It can be constraining, expensive and also prevent others from using the resources meanwhile. Because of all these reasons, debugging with real humans is tiresome and extends the development duration. However, real-life tests are mandatory for the final validation of a system.

B. Operator-controlled simulated human

One of the main benefits of simulations [10] is the fact that there is no need to access the real robot. Hence, multiple tests can be run simultaneously by running several instances of the simulation environment and even faster than in real life. Moreover, the environment for the tests can be changed very easily in contrast to real-life tests. Thus, simulations are quicker and easier to perform which makes them ideal for testing and debugging during development.

However, despite the benefits of simulating the robot and the environment, it is very challenging to simulate humans and their behavior. This is the main drawback of simulation in the HRI context. A solution to this issue is to manually control the human avatar [11]. This can be done by using a game controller, motion capture, or online interactive tools [12]. Augmented or Virtual Reality can also be used to improve immersion and thus have even more realistic reactions from the human avatar [13]. However, both Virtual Reality and motion capture benefits mostly the manipulation scenarios because the human will be able to control the avatar with high fidelity. However, navigation scenarios require a game controller or a keyboard to move the avatar. Creating realistic and accurate movements with such devices is not very easy. Moreover, having a controlled avatar requires a real human only focused on controlling it. It brings us back to some limitations discussed in the previous section like the running speed or the limitations due to human fatigue.

C. Autonomous avatar

Using an autonomous simulated human is the ideal solution to test HRI systems, but intelligent autonomous avatars do not exist yet. Autonomous avatars can also be classified into three categories of intelligence.

Firstly, the avatar can be scripted. It only executes a series of predefined actions without being reactive to its environment. It is easy to set up, but interactions will be very limited. Nevertheless, scripting is a decent solution to quickly debug or tune a system in its very early stages.

Secondly, the avatar can be purely reactive. Many works [14], [15] focus on crowd navigation, and such contexts can now be simulated efficiently with reactive models. Works like MengeROS [16] or PedSim_ROS[1] use the social force model and offer a scalable and efficient way to simulate crowds of human agents. Other crowd simulators using the ORCA method can be found like the work in [17]. Reactive-only methods are clearly useful for crowds but provide very limited possibilities when trying to simulate individual agents, particularly in intricate scenarios and narrow environments.

D. Our contribution

Given the limitations mentioned previously, this work aims to help people working in the field of Human-Aware or Social Robot Navigation to test and debug their systems. To do so, we provide a system designed to run, analyze and evaluate repeatable and long navigation scenarios involving a robot and an autonomous reactive and rational avatar. Our work is focused on intricate and narrow scenarios where, in addition to being reactive, rational decisions should be taken in order to solve the conflicts occurring. Additionally, please note that our system is using an existing motion planner. Thus our contribution is focused on navigation decision-making and not the motion generation itself.

III. InHuS System

The InHuS System[2] works along with a human operator, a chosen simulator, and the challenged robot controller as depicted in Fig. 1. The system is majorly implemented using ROS. Besides, note that the InHuS System is three-sided. First, the system comes with a high-level interface called Boss that helps 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.

A. 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 navigation action to be achieved. However, the human agent can handle “Compound goals” that need a specified sequence of navigation and waiting for actions to be achieved. This type of compound goal

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

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

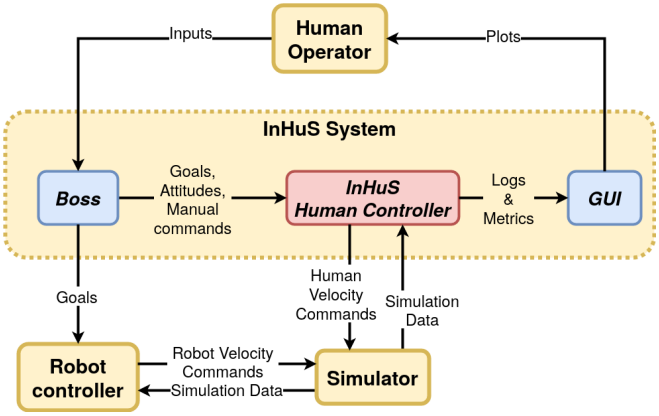

Fig. 1. The InHuS System interacts with three external systems: the simulator, the robot controller, and a human operator. Our system is separated into three parts: the Boss high-level interface gathering inputs from the human operator, InHuS which is the actual human controller, and a GUI to plot the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

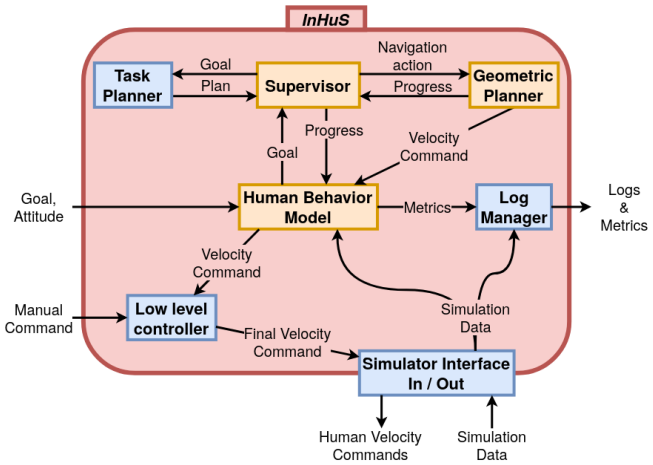

Fig. 2. The human controller InHuS is itself composed of several component and subsystems. All these sub-components interact together to produce some reactive and rational behavior for the avatar and also to produce logs and metrics for further analysis.

is useful to emulate more complex activities. For example, "Make coffee" could be described as a sequence of three actions: nav_action(coffee_machine), wait_action(15s), nav_action(my_office).

The Boss allows defining scenarios with start positions and goals for each agent to repeatedly generate the same situation. The starting positions of these scenarios are defined with "Pose goals" for all the agents. Since only the human can handle "Compound goals", the goal for the robot must be a "Pose goal", whereas it can be of any of the available types for the human agent. After initializing the scenario, i.e., after sending the agents to their respective starting positions, the scenario can be started, which sends the corresponding goals 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. In this mode, the agent continuously gets a new goal from a given list after completing a goal.

Each navigation action can specify a radius for the "Pose goal", within which a new "Pose goal" can be 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 from one environment to another, and the corresponding goal file will be loaded automatically.

B. 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.

1) 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 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. other 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, here set to $1.5m$, of the blocking location, 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 the modes affecting both goal decisions and reactions towards 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 [18], and 3) stopping close to the robot and looking at it for a few seconds before resuming its goal which emulates a curious behavior. Combined with these

Attitudes the human agent can challenge the robot planner with a variety of situations.

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 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.

2) 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. If the received goal is a "Pose goal", a plan filled with a single navigation action is generated. Otherwise, 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 can handle 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 GeometricalPlanner 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*.

3) 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.

InHuS comes with two different planners that can be selected while starting the system. The first one uses the default global planning[3] from the navigation stack and an openly available local planner called *teb_local_planner*[4] which is based on the timed elastic band [19] approach.

The second one is a publicly available human-aware navigation planner from our lab called CoHAN [20]. It is also 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. To be used in InHuS, slight modifications were made to CoHAN to remove some conflicting features. More details and results on this integration can be found in our previous work [21]. With this planner, 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, and this helps to generate different agent behaviors.

The first planner is computationally lighter and can be used to save performance in scenarios that don't require much anticipation. Yet, the CoHAN planner produces more relevant movements with some additional computation power, and we tend to make it the default planner in the upcoming versions. Besides, this illustrates the ability to use different motion planners, even custom ones, in InHuS.

C. 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 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

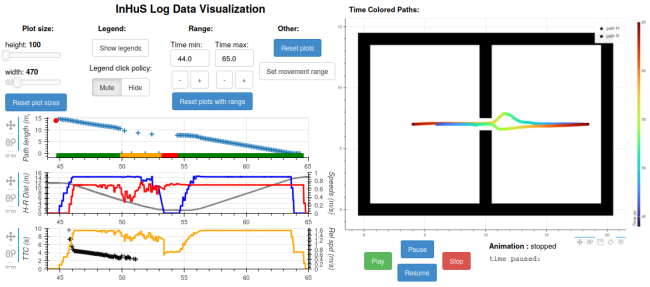

Fig. 3. Overview of the entire GUI interface. All log data and metrics produced by InHuS are plotted here. More legible figures of the plots are given and analyzed in the following sections. The interface is organized as follows. On the right side are shown the paths taken by the agents and colored over time. On the left side are shown several metrics and data plotted over time on graphs. Additional widgets help to configure the plots.

[3]http://wiki.ros.org/global_planner

[4]wiki.ros.org/teb_local_planner

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.

IV. 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 is able to challenge a robot navigation system and how we can interpret the produced plots of metrics. Next, we show how the InHuS System can compare human-aware performances using two different robot controllers. Finally, we present some additional experiments to show how InHuS can generate diverse behaviors using the *Attitudes* and how long runs can benefit the development of a robot planner.

A. 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 upper 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 type of behavior is confusing, and the agent's intentions might be unclear to the robot planner. The narrow corridor scenario, shown in the lower 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 one of them to back off. Here the path

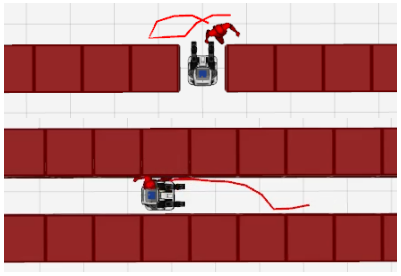

Fig. 4. In the doorway scenario at the top, the reactive-only (Pedsim) agent never stops moving and trying to go through the robot even though its path is blocked. Moreover, sometimes the agent squeezes itself between the wall and the robot colliding with both, like with the narrow corridor scenario at the bottom. Not having collision shapes is a big limitation for Pedsim since it can't realistically react to intricate conflicts.

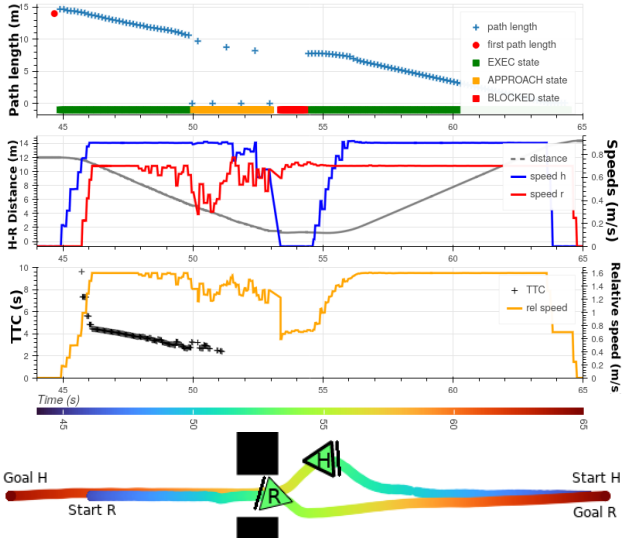

Fig. 5. Condensed view of the InHuS GUI after running the doorway scenario with the CoHAN planner on the robot. Several subplots can be seen as the time-colored paths of the agents. The robot blocks the human's path while crossing. The conflict is detected by InHuS which switches into the approach state and eventually to the blocked state (orange and red line in first subplot). The velocity of the robot decreases as it approaches the human, and it only increases again when there is no TTC value (second and third subplots). Accelerating only when there is no collision threat shows some intended human-aware behavior.

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 the repulsive social forces to prevent the 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 like the above.

Based on the above observations, we can infer that such approaches can work well in large spaces or crowds but could lead to confusing or even weird behaviors in narrow environments and intricate scenarios, where conflict resolution is required, apart from being reactive.

B. Interpretation of plots with human-aware planner

The InHuS System is able to generate challenging situations that can be analyzed further with the plots generated from the log data and metrics. 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 already mentioned doorway scenario.

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 $50s$ 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 to the blocking robot around $53s$, 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 $51s$, 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.

C. Quantitative comparison between two robot controllers

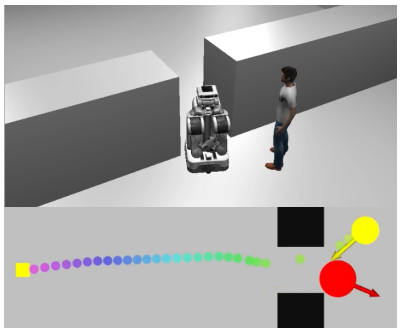

Fig. 6. MORSE simulator and RViz views of the doorway scenario being executed with the CoHAN planner. The planned path of the avatar to the goal can be seen.

Our system can be used to run similar scenarios repetitively to produce robust metric values. With these values, we can help measure 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, and 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 I. The metrics are respectively: the time taken by the avatar to reach its goal (Total time), the minimum distance between the robot and the human (HRDist), and the minimum time to collision (TTC). Intuitively, we want the total time 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.

At first glance, we see in Table I 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 total time 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 of the computed metrics compared to a non-human-aware robot controller. This infers that our system can help evaluate and compare human-aware robot controllers.

D. Attitudes and Long runs

In this section, some details on how InHuS can generate different agent behaviors, with *Attitudes*, are given. We also briefly explain how conducting long runs and scenarios can benefit the robot planner.

1) Generating different behaviors: InHuS is capable of generating different agent behaviors to diversify situations and conflicts to challenge the robot navigation system. One way to do so is by tuning InHuS parameters about the navigation conflicts or the GeometricPlanner parameters.

Experiment	CoHAN			SMB		
	Total Time (s)	*Min HRDist* (m)	*Min TTC* (s)	*Total Time* (s)	*Min HRDist* (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

TABLE I

MEAN VALUES OF SOME 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 PERFORMANCES THAN THE OTHER.

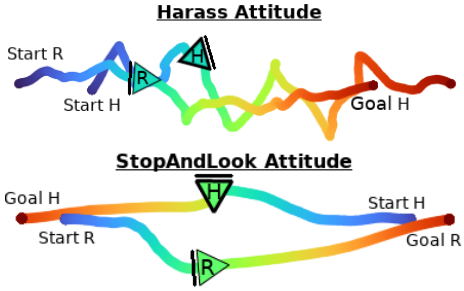

Fig. 7. Behaviors obtained by activating the *Harass* and *StopAndLook Attitudes*. With *Harass* (top), the human always goes in front of the robot. By observing the colors, one can that the human is always ahead of the robot. With *StopAndLook* (bottom), the human stops to look at the robot for a few seconds when close to it. This can be seen in the plot as a sudden change of color from green to dark blue in the human's path.

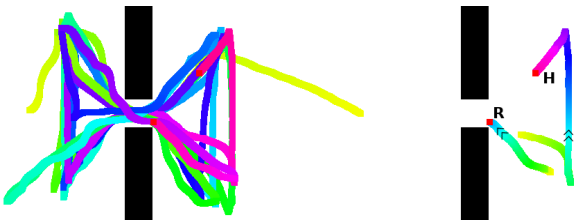

Fig. 8. Execution of the long run scenario using the TDP robot planner and InHuS. We see the complete set of time-colored paths on the left. On the right, the same path is cut, around the moment when the robot got stuck in the wall. Long runs help to debug such unexpected issues.

Changing the velocity and path planning of the agent has a lot of influence on the produced behavior. Besides parameter tuning, activating *Attitudes* produces more complex behaviors and reactions. We present the time-colored paths for the execution of two *Attitudes* : *Harass* and *StopAndLook* in Fig. 7 . 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. At the bottom of the same figure we see the execution of the *StopAndLook Attitude*. The color discontinuity behind the human marker shows how this *Attitude* makes the human suspend 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.

2) Long runs and scenarios: Finally, the proposed system can do long runs with the help of the Boss component that autonomously sends goals to the agents. Such a feature is interesting as it helps to test the robot planner's stability and robustness. Moreover, when randomness is added to the goals of the long run, unexpected situations and conflicts might be generated. Some of these generated situations could be of interest, and the navigation planner under the test might have to be modified to address these. For instance, Fig. 8 depicts a long run executed with InHuS and a human-aware robot planner from Kollmitz et. al. [22] referred to here as TDP. The agents were made to endlessly loop over four goal-positions (each with a $1m$ 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, which could be an issue of interest. 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.

V. 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[5], Stage[6] and Gazebo[7]. Furthermore, any robot controller using the ROS Navigation Stack can be directly used with InHuS.

Like any other system, InHuS has limitations too. Firstly, simulating intelligent human avatars is a novel field, and only a few very recent works address this topic. The currently existing ones neither provide any implementation details nor code or system to run. Thus we are looking forward to testing other similar systems and comparing them with InHuS. Secondly, we claim to generate only reactive and some rational behavior, which is still far from natural or realistic human behavior. We currently handle scenarios with

[5]https://morse-simulator.github.io
[6]https://github.com/ros-simulation/stage_ros
[7]http://gazebosim.org

two agents only, the human and the robot. We can run scenarios with other human agents, but they will be treated like robots.

VI. Conclusion and Future work

Human-aware robot navigation is rapidly growing, but the community lacks good human agent simulations to test and debug their systems. Reactive-only approaches exist, but we have shown that they are limited. Through the InHuS system, we propose 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 are able to compare the human awareness of 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. The 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, it would be possible to integrate situation detection and diagnosis in the long run to catch the problematic situations that need to be analyzed afterward to tune, refine or extend a given planner. We also plan to handle scenarios with more human agents, like groups and maybe even crowds, by using both some intelligent and reactive-only agents. Finally, We also intend to enrich the set of available metrics and generation of conflicting scenarios that could help evaluate social robot navigation.

References

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

[2] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[3] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "VirtualHome: Simulating Household Activities Via Programs," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 8494–8502.

[4] 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 (HAI)*, November 2020.

[5] L. Yige, L. Siyun, L. Chengshu, P.-D. Claudia, and S. Silvio, "Interactive pedestrian simulation in igibson," *RSS Workshop on Social Robot Navigation*, 2021.

[6] B. Shen, F. Xia, C. Li, R. Martın-Martın, L. Fan, G. Wang, S. Buch, C. D'Arpino, S. Srivastava, L. P. Tchapmi, K. Vainio, L. Fei-Fei, and S. Savarese, "igibson 1.0: a simulation environment for interactive tasks in large realistic scenes," *arXiv preprint*, 2020.

[7] S. Vattam, M. E. Klenk, M. Molineaux, and D. W. Aha, "Breadth of approaches to goal reasoning: A research survey," in *Annual Conference on Advances in Cognitive Systems: Workshop on Goal Reasoning*, 2013.

[8] B. Johnson, M. W. Floyd, A. Coman, M. A. Wilson, and D. W. Aha, "Goal Reasoning and Trusted Autonomy," in *Foundations of Trusted Autonomy*. Springer, 2018, vol. 117.

[9] A. Thomaz, G. Hoffman, and M. Çakmak, "Computational human-robot interaction," *Found. Trends Robotics*, vol. 4, no. 2-3, pp. 105–223, 2016.

[10] H. Choi, C. Crump, C. Duriez, and E. A. Elmquist, "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward," *Proceedings of the National Academy of Sciences*, vol. 118, no. 1, Jan. 2021.

[11] G. Echeverria, S. Lemaignan, and A. e. A. Degroote, "Simulating Complex Robotic Scenarios with MORSE," in *Simulation, Modeling, and Programming for Autonomous Robots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7628, pp. 197–208.

[12] N. Tsoi, M. Hussein, O. Fugikawa, J. D. Zhao, and M. Vázquez, "An approach to deploy interactive robotic simulators on the web for hri experiments: Results in social robot navigation," 2021.

[13] O. Liu, D. Rakita, B. Mutlu, and M. Gleicher, "Understanding human-robot interaction in virtual reality," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. Lisbon: IEEE, Aug. 2017, pp. 751–757.

[14] T. Fraichard and V. Levesy, "From Crowd Simulation to Robot Navigation in Crowds," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 729–735, Apr. 2020.

[15] B. Zhang, J. Amirian, H. Eberle, J. Pettré, C. Holloway, and T. Carlson, "From HRI to CRI: Crowd Robot Interaction-Understanding the Effect of Robots on Crowd Motion," *International Journal of Social Robotics*, pp. 1–13, June 2021.

[16] A. Aroor, S. L. Epstein, and R. Korpan, "MengeROS: a Crowd Simulation Tool for Autonomous Robot Navigation," *arXiv:1801.08823 [cs]*, Jan. 2018.

[17] W. van Toll, F. Grzeskowiak, A. L. Gandía, J. Amirian, F. Berton, J. Bruneau, B. C. Daniel, A. Jovane, and J. Pettré, "Generalized microscropic crowd simulation using costs in velocity space," in *Symposium on Interactive 3D Graphics and Games*. ACM, 2020, pp. 1–9.

[18] T. Nomura, T. Kanda, H. Kidokoro, Y. Suehiro, and S. Yamada, "Why do children abuse robots?" *Interaction Studies*, vol. 17, no. 3, pp. 347–369, 2016.

[19] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[20] P. T. Singamaneni, A. Favier, and R. Alami, "Human-aware navigation planner for diverse human-robot ineraction contexts," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[21] A. Favier, P.-T. Singamaneni, and R. Alami, "Simulating Intelligent Human Agents for Intricate Social Robot Navigation," in *RSS Workshop on Social Robot Navigation 2021*, Washington, United States, July 2021.

[22] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard, "Time dependent planning on a layered social cost map for human-aware robot navigation," in *2015 European Conference on Mobile Robots (ECMR)*. Lincoln, United Kingdom: IEEE, Sept. 2015, pp. 1–6.