

# Towards Automated Scenario Testing of Social Navigation Algorithms

Shashank Rao Marpally<sup>1</sup>, Pranav Goyal<sup>1</sup>, and Harold Soh<sup>1,2</sup>

<sup>1</sup>Dept. of Computer Science, National University of Singapore.

<sup>2</sup>Smart Systems Institute, NUS.

Email: {smarpally, pgoyal, harold}@comp.nus.edu.sg

**Abstract**—In this work, we take a step towards automating scenario-centric testing of social navigation algorithms. We propose a pipeline that generates context, task, and location-appropriate social navigation scenarios that can be readily realized in a simulator. Our pipeline takes simple scenario metadata and first generates a grounded textual scenario, then infers the pedestrian and robot paths as well as the behaviors of the pedestrians, which enables simulating the scenario through HuNavSim [9]. We use the social reasoning and code-writing abilities of Large Language Models (LLMs) to enable scenario generation and translation. Our experiments demonstrate that the design choices in our pipeline generate realistic simulation scenarios and significantly enhance scenario translation performance compared to naive LLM prompting.

## I. INTRODUCTION

Deploying robots in human-inhabited areas requires them to not only be performant and safe, but also socially adept. However, social adeptness is an abstract, context-dependent, and hard-to-quantify skill, making it challenging to evaluate [1]. For example, it is difficult to quantify and evaluate compliance with subjective objectives/constraints such as proactivity and contextual appropriateness.

One approach towards evaluating social robots is via scenario testing, which is a popular framework that tests a system in realistic and task-relevant scenarios. A scenario testing framework involves the test designers consulting prior data and assuming the role of an end user to design relevant test scenarios. However, hand-crafting individual scenarios for testing social navigation (SocNav) is challenging, tedious, and not scalable. In addition to the quantification issues outlined above, SocNav robots typically operate in unstructured and unpredictable human environments, which makes it difficult to identify important scenarios.

In this work, we propose a pipeline for automating scenario generation for social navigation (Fig 1). Along with our simple map annotation tool, our pipeline is capable of generating a wide variety of social context and robot-task-appropriate scenarios given a simulated location. We harness the power of Large Language Models (LLMs) to propose textual scenarios as well as transforming them into components that enable simulation through the HuNavSim [9] framework. We showcase the utility of our system by generating four realistic scenarios. The ease of use and scalability of our tool brings us a step closer to enabling accessible scenario-centric testing for social navigation algorithms.

## II. BACKGROUND

We build on recent work in SocNav evaluation frameworks. Francis et al. [1] provides an overview of the most prevalent simulation and evaluation frameworks.

Unlike current SocNav benchmarks which mainly focus on proxemics-based metrics in random dense crowds, we focus on evaluating navigation algorithms on specific social scenarios. Scenario cards, as described by [1], provide a structured method for defining and generating scenarios. A scenario card includes Metadata, Definition (location, intended robot task, intended human behavior), and a usage guide. For our framework, the user provides parts of the Metadata and Definition, which are used to infer the *path* and the *behavior* of the pedestrians in the simulation. SEAN 2.0 [11] generates scenarios represented by propositional logic, using behavior graphs initialized on the scene, but the occurrence of the provided limited scenarios is not guaranteed and the environments for scenario generation are limited. Additionally, current SocNav simulators do not support verbal/gesture-based interaction between the pedestrians and the robot, which is common in real-life scenarios.

Pedestrians are typically modeled as moving obstacles controlled by navigation algorithms like ORCA [12] and SFM [4] or by prerecorded trajectories from real data [3]. The trajectories of the pedestrians are typically random or need to be handcrafted to orchestrate specific scenarios [10]. Neither approaches allow scalable generation of context-appropriate diverse scenarios. Recent works like HuNavSim [9], SEAN 2.0 [11] and Arena 3.0 [6] enable pedestrians in the simulation to display more intelligent behaviors like grouping. However, to orchestrate an intricate scenario, the behavior of each pedestrian has to be handcrafted individually. Additionally, using our simple map annotation tool, we enable scenario generation in any arbitrary map, while previous works depend on hand-crafted map features. LLMs have been recently shown to be performant in human modeling [14], commonsense and social reasoning [8, 5] and code writing [7]. We harness these capabilities for scenario proposals as well as human path and behavior generation.

## III. METHODOLOGY

Fig. 1 shows an overview of our framework. We utilize LLMs for proposing textual scenarios given the Social Context, Location Description, Intended Robot Task and optionally

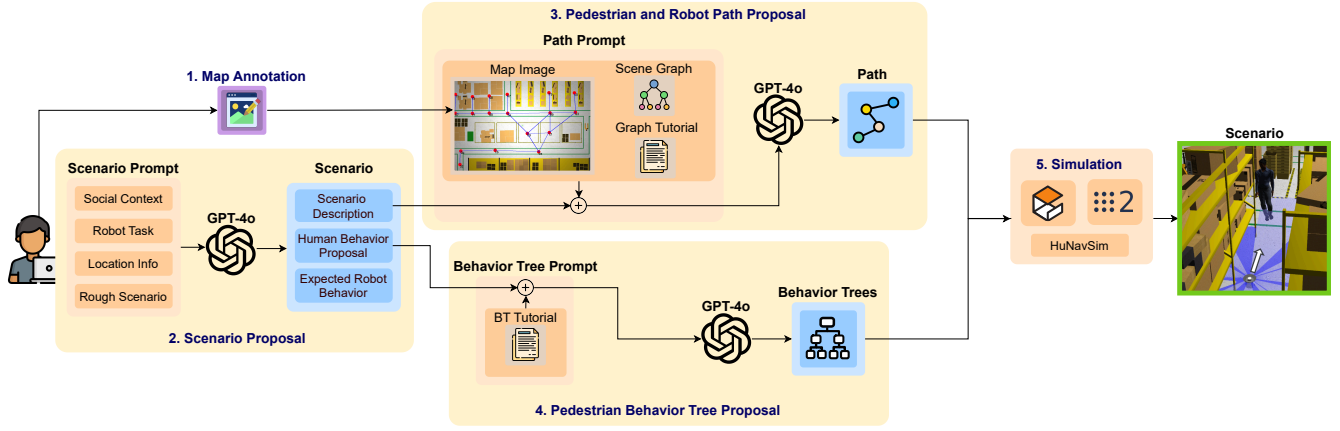


Fig. 1: Overview of our pipeline from scenario proposal to simulation execution for SocNav. Please see text for details.

a Rough Scenario (we call these inputs: *scenario metadata*), and then translate the description to a simulation instance by querying LLMs with the Scene Graph and an extended library of available behaviors from HuNavSim. We build our framework using HuNavSim because it specifies pedestrian behaviors through behavior trees, which an LLM can express textually in XML, and because it is based on ROS2, thus making integration with other robotics frameworks straightforward. This pipeline enables an easy transition from textual scenario metadata to a Gazebo<sup>1</sup> simulation instance through ROS2. Our framework mainly consists of 5 parts:

- 1) Map Annotation(Sec.III-A)
- 2) Scenario proposal (Sec. III-B)
- 3) Pedestrian and Robot path proposal (Sec. III-C)
- 4) Pedestrian Behavior Tree proposal (Sec. III-D)
- 5) Simulation (Sec. III-E)

#### A. Map Annotation

We enable our framework to generate scenarios in any simulated location by providing a scene graph and an annotated overhead image of the location as inputs to the LLM for path generation. This helps provide location-specific context to the LLM, allowing it to generate pedestrian paths via a sequence of scene graph nodes grounded in the given location. The scene graph nodes (with associated pixel coordinates) can then be converted to world coordinates with a simple transformation matrix. Thus, we expect the user to provide an overhead orthographic image that covers the full scene, and the transformation matrix to convert pixel coordinates to world coordinates. Our annotation tool helps the user in generating a scene-graph overlaid overhead image and a corresponding JSON scene graph for their specific locations. The node and edge schema for the scene graph in the annotation tool is customizable, and our pipeline is independent of the keywords used in the schema. An example scene graph overlaid on the map image for the Small Amazon Warehouse<sup>2</sup> is shown in Fig. 1. Note that, to reduce sequence-related biases by the LLM, we

use random 2-bit alphanumeric tokens for scene graph node names.

#### B. Scenario proposal

To generate a scenario relevant to the scenario metadata, in the given location, we construct a prompt for the LLM with the following structure:

- 1) Definition of Social Navigation and Scenarios
- 2) Capabilities of human agents in the simulation (derived from the behaviors available for use within the extended HuNavSim library described in Section III-D)
- 3) Rules to be followed when proposing Human Behaviors and Scenario Description
- 4) Social Context and Intended Robot Task: We ask the user to describe the social context for the robot (e.g.: “*The robot is an evacuation robot in geriatric hospital*”) and the task of the robot (e.g.: “*The robot is trying to guide patients to the nearest exit*”). These help in guiding the LLM to generate relevant scenarios.
- 5) Location Description: A short description of the location. Contains all the node and edge types used in the scene graph schema.
- 6) Rough Scenario (optional): The user can provide a scenario description that “roughly” specifies the scenario they would like the framework to generate.

We instruct the LLM to output the following:

- 1) Scenario Description: A description of the scenario that gives a basic idea about what the humans and robot are doing in the context of the given location.
- 2) Human Behaviors: A description of how humans behave in the presence and absence of the robot.
- 3) Expected Robot Behavior: A description of how the robot is expected to behave in the given scenario. We expect our tool to be used to generate scenarios that can be used to evaluate subjective socialness metrics via human evaluation.

We find that providing example inputs and corresponding scenarios greatly improves the quality of the LLM’s responses,

<sup>1</sup><https://gazebo.org/home>

<sup>2</sup><https://github.com/aws-robotics/aws-robomaker-small-warehouse-world>

and we provide a set of handcrafted responses in the prompt as examples.

### C. Pedestrian and Robot path proposal

Pedestrian paths and the robot’s waypoints for the scenario are specified by an LLM, with scene graph node names. We instruct the LLM to generate paths for each pedestrian in the scene by structuring a prompt with:

- 1) Scene Graph in JSON and the Annotated Map image as described in Sec III-A
- 2) Task Description: The LLM is instructed to choose a sequence of nodes from the scene graph to specify the robot and human paths.
- 3) Scene graph “tutorial”: A set of example Graph Q&A examples that delineate how the scene graph is to be used. Currently, this is adapted for the Small Amazon Warehouse Map but can be scripted to be generated for any map. We find that including such pedagogical examples improves the likelihood of the LLM generating a correct output.
- 4) Scenario Description (from III-B)

We also instruct the LLM to assign a *group ID* to each human in the scenario. HuNavSim uses these group IDs to generate groups of pedestrians dynamically in the simulation. We find that often the LLM can make mistakes and generate a discontinuous path. We detect such errors and requery the LLM by identifying its mistake and appending the chat history as context. We requery the LLM with identified mistakes for a maximum of 3 times and thereafter restart the conversation if it fails to generate valid paths again. We also provide example scenarios and corresponding paths to improve the quality of the LLM’s responses.

### D. Pedestrian Behavior Tree proposal

We specify human behaviors with Behavior Trees (BT) through BehaviorTree.CPP, which is used by HuNavSim to control the actions of the humans in the simulation. The behavior nodes available in HuNavSim do not support any interaction between the human and the robot and other complex behaviors. Thus, we implement additional behavior nodes to enable the generation of scenarios encompassing the 25 scenarios shown in Francis et al. [1], which include interactive scenarios (e.g. Intersection (“wait”), Entering Room etc.). For generating behavior trees, for each pedestrian, we construct a prompt for the LLM with the *Behavior Tree node library* (A list of the available behavior tree action and condition nodes that the LLM can compose to create the required behavior) and the proposed Human Behaviors (from III-B). As in the path generation module, we also provide a Behavior Tree “tutorial” with instructions on syntax, rules, and Q&A and example behavior descriptions and BTs. Including the pre-existing action/condition nodes in HuNavSim, the following conditions and actions are available for the LLM to use in the *Behavior Tree node library* (their functionality is self explanatory):

TABLE I: Specifications for testing the Scenario proposal

	Social Context	Robot Task
Scene 1	Emergency response robot inside a warehouse in a disaster situation.	Guide humans to safety
Scene 2	Delivery bot in a small warehouse.	Transport boxes in the warehouse
Scene 3	Maintenance robot inside a warehouse.	Clean the warehouse

- 1) Condition Nodes: *RobotSays* (check if the robot is gesturing “wait”, “proceed”, “excuse me”, “acknowledged”), *HasRobotMoved*, *IsRobotBlocking*, *IsRobotVisible*, *IsGoalReached*, and *TimeExpiredCondition*
- 2) Action Nodes: *MakeGesture* (“wait”, “proceed”, “excuse me”), *LookAtRobot*, *FollowRobot*, *AvoidRobot*, *BlockRobot*, and *GiveWaytoRobot*.

### E. Simulation

The scene graph node sequence output by the LLM for each pedestrian in the scenario is parsed, converted to world coordinates, and written as navigation goals along with the group IDs to a YAML file. The behavior trees are written to XML files. Running a simulation with these files through HuNavSim then creates a gazebo instance where the pedestrians follow the specified paths and act according to the BTs. To simulate interaction (gesturing), the pedestrians and the robot publish the required gesture to specific ROS topics, which can be subscribed to by the other agents. We provide a complete example of the user inputs and the corresponding outputs from the different parts of the pipeline in the Appendix.

## IV. RESULTS

We showcase the utility of our framework by describing 4 of the scenarios generated using our pipeline. For each scenario, we generate a simulation from the pipeline in Gazebo and teleoperate the turtlebot3 robot <sup>3</sup> through the waypoints output by the LLM. We also separately evaluate the scenario proposal module for correctness and realism of the generated scenarios and the Path and BT proposal modules for correctness w.r.t a given rough scenario.

### A. Scenario Proposal

We evaluate our scenario proposal module by generating 5 scenarios for 3 different scenario metadata and subjectively evaluate if they are realistic, executable, and relevant to the specifications. We used scenarios characterized by the various social contexts and robot tasks mentioned in Table I.

We observed a 100% success rate for the scenario proposal module when a rough scenario was provided and a 60% success rate when we prompted the LLM to generate scenarios with no rough scenario. Thus, our framework can generate grounded, simulatable, and context-appropriate scenarios. Fig. 4 shows a scenario generated by our pipeline for Scene 1. Here, the scenario starts with the robot navigating towards

<sup>3</sup><https://emanual.robotis.com/docs/en/platform/turtlebot3/>

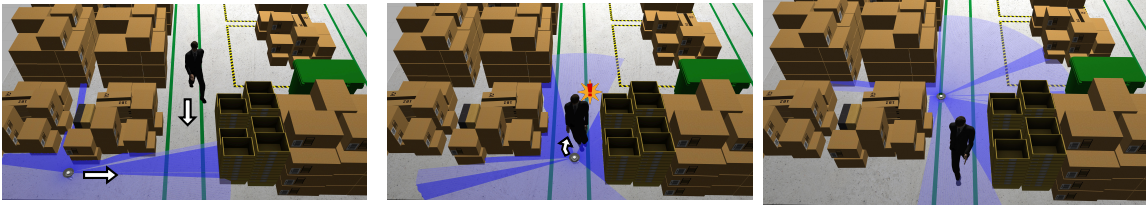


Fig. 2: A robot encounters a human in a blind corner and the human is startled.

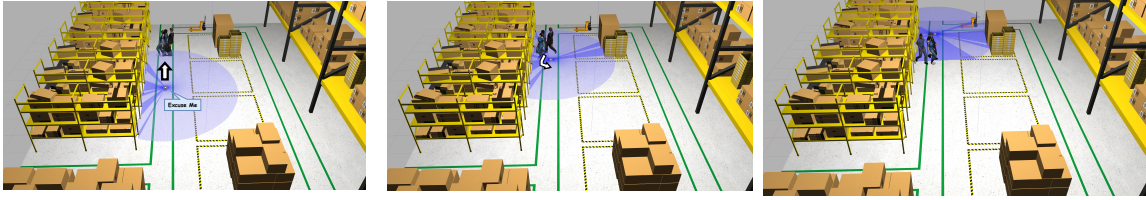


Fig. 3: A robot encounters a group and requests passage. The group gives way to the robot.

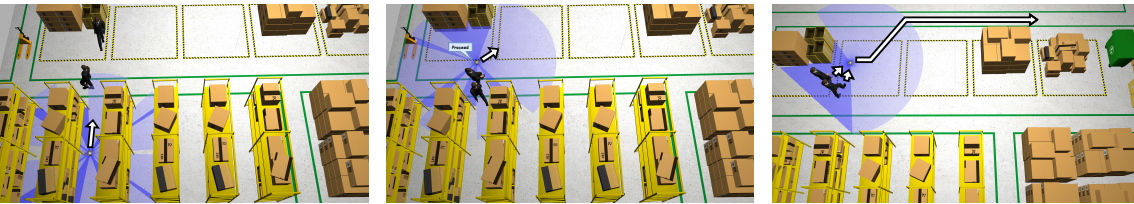


Fig. 4: An evacuation robot guides 2 humans to an exit. The humans follow the robot.

TABLE II: Scenarios for Path and BT proposal.

	<b>Rough Scenario</b>
<i>Scene A</i>	Robot starts inside a narrow aisle, A human is standing at the opposite end of the aisle. The human is still until the robot is blocking its path, if not the human starts moving towards its goal.
<i>Scene B</i>	Robot turns at a blind corner and encounters a human. The human sees the robot when it is very close and stops completely until the robot is not blocking it anymore.
<i>Scene C</i>	While transporting a box, the robot comes across a group of 3 humans. The humans checks if the robot says excuse me and if yes, then they give way to the robot, or else they continue standing in the group

the first human on the left and gesturing “PROCEED”, who, upon receiving the instruction, starts following the robot. Another human initially moves towards the robot, then follows it as well. The robot then “guides” the pedestrians to the exit waypoint. Note that this emulates a realistic evacuation scenario and was generated from scratch using our pipeline.

#### B. Path and BT Proposal

To test the path and behavior generation modules, we run the pipeline 5 times for 3 scenarios and subjectively evaluate them for correctness w.r.t their fidelity to the provided rough scenario as well as their simulatability. For all the scenarios we used the same Social Context of Scene 2 (Table I). We provided different rough scenarios for each scene (Table II). Scenes A and B are inspired by real-life pedestrian interaction studies done by Fujioka et al. [2] and Wolfinger [13], respectively. To evaluate our design decisions, we tested our method against a Naive method, where we directly queried the

TABLE III: Evaluation of Path and BT generation

	Path Validity	Behavior Tree Validity
Naive	0.26	0.13
<b>Ours</b>	<b>0.86</b>	<b>0.86</b>

LLM to output the scenario descriptions, paths, and the BTs, all at once, without our tutorials and extensive examples. The results in Table III show that our modular design and prompt engineering significantly improve the success rate of scenario execution.

#### V. CONCLUSION

In this work, we presented a customizable and automated pipeline for proposing and generating diverse social navigation scenarios in any simulated location. This is a first step towards a scenario-centric testbed for evaluating social navigation algorithms. Paired with human evaluation, we expect scenario-based testing to greatly complement current proxemic-centric benchmarks by enabling testing of subjective and hard-to-define social metrics through scenario-centric evaluation.

We plan to make a number of improvements to our framework in the next iteration. Currently, we expect the user to provide an orthographic map image enclosing the full scene along with the transformation matrix for pixel coordinates to world coordinates. We plan to automate this module so that only a gazebo world file is required. We also find that despite providing numerous examples in the prompt, the LLM makes logical/reasoning errors. We plan to design a verification module to match the generated simulator scene with the scenario description generated in III-B.



## REFERENCES

- [1] Anthony Francis, Claudia Pérez-D’Arpino, Chengshu Li, Fei Xia, Alexandre Alahi, Rachid Alami, Aniket Bera, Abhijat Biswas, Joydeep Biswas, Rohan Chandra, Hao-Tien Lewis Chiang, Michael Everett, Sehoon Ha, Justin W. Hart, Jonathan P. How, Hareesh Karnan, Tsang-Wei Edward Lee, Luis J. Manso, Reuth Mirksy, Soeren Pirk, Phani Teja Singamaneni, Peter Stone, Ada V Taylor, Pete Trautman, Nathan Tsoi, Marynel Vázquez, Xuesu Xiao, Peng Xu, Naoki Yokoyama, Alexander Toshev, Roberto Martin-Martin Logical Robotics, Nvidia, Stanford, Google, Epfl, Purdue, Cmu, Ut Austin, Mit, Northeastern, Georgia Tech, Aston, Bar Ilan, Adobe, Laas-Cnrs, Université de Toulouse, AI Sony, Honda, Yale, Gmu, and Apple. Principles and Guidelines for Evaluating Social Robot Navigation Algorithms. *ArXiv*, abs/2306.16740, 2023. URL <https://api.semanticscholar.org/CorpusID:259287246>.
- [2] Yusuke Fujioka, Yuyi Liu, and Takayuki Kanda. I Need to Pass Through! Understandable Robot Behavior for Passing Interaction in Narrow Environment. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI ’24, page 213–221, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703225. doi: 10.1145/3610977.3634951. URL <https://doi.org/10.1145/3610977.3634951>.
- [3] Fabien Grzeskowiak, David Gonon, Daniel Dugas, Diego Paez-Granados, Jen Jen Chung, Juan Nieto, Roland Siegwart, Aude Billard, Marie Babel, and Julien Pettré. Crowd against the machine: A simulation-based benchmark tool to evaluate and compare robot capabilities to navigate a human crowd. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, page 3879–3885. IEEE Press, 2021. doi: 10.1109/ICRA48506.2021.9561694. URL <https://doi.org/10.1109/ICRA48506.2021.9561694>.
- [4] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, May 1995. doi: 10.1103/PhysRevE.51.4282. URL <https://link.aps.org/doi/10.1103/PhysRevE.51.4282>.
- [5] Minae Kwon, Hengyuan Hu, Vivek Myers, Siddharth Karamcheti, Anca Dragan, and Dorsa Sadigh. Toward Grounded Commonsense Reasoning. 2024. URL <https://arxiv.org/abs/2306.08651>.
- [6] Linh Kästner, Volodymyr Shcherbyna, Huajian Zeng, Tuan Anh Le, Maximilian Ho-Kyoung Schreff, Halid Osmaev, Nam Truong Tran, Diego Diaz, Jan Golebiowski, Harold Soh, and Jens Lambrecht. Arena 3.0: Advancing Social Navigation in Collaborative and Highly Dynamic Environments, 2024. URL <https://www.arxiv.org/abs/2406.00837>.
- [7] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. URL <https://ieeexplore.ieee.org/document/10160591>.
- [8] Karthik Mahadevan, Jonathan Chien, Noah Brown, Zhuo Xu, Carolina Parada, Fei Xia, Andy Zeng, Leila Takayama, and Dorsa Sadigh. Generative Expressive Robot Behaviors using Large Language Models. 2024. URL <https://doi.org/10.1145/3610977.3634999>.
- [9] Noé Pérez-Higueras, Roberto Otero, Fernando Caballero, and Luis Merino. Hunavsim: A ros 2 human navigation simulator for benchmarking human-aware robot navigation. *IEEE Robotics and Automation Letters*, 2023. URL <https://ieeexplore.ieee.org/abstract/document/10252030/>.
- [10] Nathan Tsoi, Mohamed Hussein, Jeacy Espinoza, Xavier Ruiz, and Marynel Vázquez. Sean: Social environment for autonomous navigation. In *Proceedings of the 8th international conference on human-agent interaction*, pages 281–283, 2020. URL <https://dl.acm.org/doi/abs/10.1145/3406499.3418760>.
- [11] Nathan Tsoi, Alec Xiang, Peter Yu, Samuel S Sohn, Greg Schwartz, Subashri Ramesh, Mohamed Hussein, Anjali W Gupta, Mubbasir Kapadia, and Marynel Vázquez. Sean 2.0: Formalizing and generating social situations for robot navigation. *IEEE Robotics and Automation Letters*, 7(4):11047–11054, 2022. URL <https://ieeexplore.ieee.org/abstract/document/9851501>.
- [12] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium ISRR*, pages 3–19. Springer, 2011. URL [https://link.springer.com/chapter/10.1007/978-3-642-19457-3\\_1](https://link.springer.com/chapter/10.1007/978-3-642-19457-3_1).
- [13] Nicholas H. Wolfinger. PASSING MOMENTS: Some Social Dynamics of Pedestrian Interaction. *Journal of Contemporary Ethnography*, 24(3):323–340, 1995. doi: 10.1177/089124195024003004. URL <https://doi.org/10.1177/089124195024003004>.
- [14] Bowen Zhang and Harold Soh. Large language models as zero-shot human models for human-robot interaction. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7961–7968. IEEE, 2023. URL <https://ieeexplore.ieee.org/abstract/document/10341488>.

## VI. APPENDIX

### A. Example Scenario Generation

In this section, we give an example for the generation of a scenario and describe the LLM’s prompts and responses in detail. we annotated an overhead orthographic image of the Small Amazon warehouse 5 to generate the scene graph JSON and scene graph image. We used the following inputs to generate a scenario in a blind corner with a startled human:

- **Social Context:** Robot is a delivery bot in a small warehouse. There are employees performing daily duties walking around the warehouse.
- **Location Description:** The location consists of passageways, intersections, and open areas, organized to facilitate

efficient navigation and connectivity. Passageways form the primary paths that connect various parts of the facility, ensuring straightforward routes between specific nodes like corners and intersections. Intersections act as critical nodes where multiple passageways converge, enabling choices in directional movement. Open areas provide larger, unobstructed spaces that could serve as gathering places or functional zones. The entire layout is linked by direct connections between these nodes, forming a grid-like structure that aids in understanding the spatial organization and potential routes for movement within the environment.

- **Robot Task:** The robot has to transport a box from one point to another in the warehouse.
- **Rough Scenario:** The robot takes a turn at a blind corner, and encounters a human immediately. The human only sees the robot when its very close and just stops completely until the robot is not blocking it anymore

**Scenario and Behavior Proposal:** The output from the LLM for scenario description and behavior:

- **Scenario Description:** The robot is transporting boxes through a Small Amazon Warehouse. It moves through a Passageway and takes a turn at a blind corner near an intersection. Upon turning the corner, the robot encounters an employee who is walking and performing daily warehouse duties. The human, only seeing the robot when it is very close, comes to a complete stop and waits for the robot to pass before continuing with their task.
- **Human Behavior:** Stops completely upon seeing the robot very close and waits until the robot is not blocking its path anymore before proceeding.
- **Expected Robot behavior:** The robot should continue moving forward while transporting the boxes, ensuring that it safely navigates around the human who has stopped. If necessary, the robot should say “proceed” to the human to indicate that it will move quickly to clear the path.

**Path Proposal:** The output by the path generation LLM is given below. Note that the paths are node sequences in the scene graph as shown in Fig.1 and a group ID of -1 signifies a lone human (as opposed to a human in a group). The LLM has correctly selected a blind corner for the scenario proposed by the first module.

- paths:
  - ROBOT: [“11”, “07”, “f7”]
  - HUMAN1: [“f7”, “07”]
  - GroupIDs: “HUMAN 1”: -1

**Behavior Tree Proposal:** The output from the LLM for the behavior tree of the startled human is given below. The “startled” behavior is emulated by making the pedestrian stop navigation when the robot blocks their path.

```

1 <root main_tree_to_execute="StopAndWaitTree">
2   <include path="BTRegularNav.xml"/>
3   <BehaviorTree ID="StopAndWaitTree">
4     <Fallback name="StopAndWaitFallback">
5       <Sequence name="StopSequence">

```

```

        <IsRobotBlocking agent_id="{id}" distance="
0.5"/>
      </Sequence>
    </Sequence>
    <Sequence name="RegNav">
      <SetBlackboard output_key="agentid" value="
{id}"/>
      <SetBlackboard output_key="timestep" value=
"{dt}"/>
      <SubTree ID="RegularNavTree" id="agentid"
dt="timestep"/>
    </Sequence>
  </Fallback>
</BehaviorTree>
</root>

```

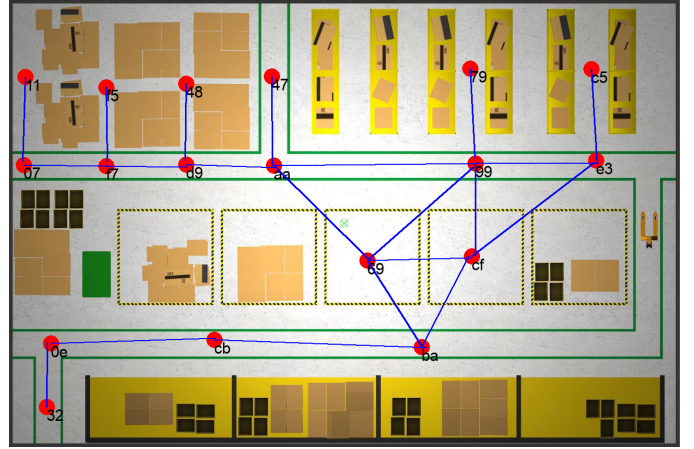


Fig. 5: Small Amazon Warehouse Map annotated with Scene graph. Each node in the graph has an associated type (here: intersection/open area/narrow aisle/area/corner) and each edge is also associated with types (here: doorway/narrow doorway/passageway).

#### B. LLM prompts

- 1 Social Navigation is a complex skill for a robot to accomplish and the appropriateness of the behavior of a robot is highly dependent on the task and the social context.
- 2 Thus a robots social navigation capabilities must be thoroughly tested, and this is done by evaluating the robots behavior in a number of scenarios in a variety of contexts.
- 3 You are a scenario designer. Your task is to generate scenarios to test the social navigation capabilities of a robot.
- 4 A Social Navigation [Scenario] is defined by:
  1. Scenario Description: very detailed description of the scenario. WHAT happens in the scenario and WHERE the scenario takes place. WHERE the robot and humans are located.
  2. Human Behavior: how human interacts with the robot when it is visible, for e.g. Human 1 is scared of the robot and asks it to stop, Human 2 doesn't notice the robot at all etc.
- 7 Your output description will be later used by an expert Behaviour tree designer to generate a Behavior Tree for each human in the scene.
- 8 The behavior tree designer is not allowed to modify the scenario and can only create behavior that can be generated using the following Actions and Conditions:
  - Conditions
    - 11 - Check the visibility of the robot
    - 12 - Check if the human has reached their goal
    - 13 - Check if robot is saying any particular phrase
    - 14 - Check if the robot is currently moving

```

15     - Check if the robot is blocking the human's path
16
17 - Actions:
18     - Make the human perform a gesture.
19     - Make the human perform normal navigation to reach
    its goal and treat the robot as a normal obstacle.
    This is regular behavior for humans.
20     - Make the human look in the direction of the robot
21     - Make the human follow the robot
22     - Make the human scared of the robot and avoid it.
23     - Make the human give way to the robot
24     - Make the human move quickly towards the front of
    the robot and block the robot.
25
26     NOTE: AT ANY GIVEN POINT OF TIME, THE HUMAN CAN
    ONLY PERFORM ANY ONE OF THE ABOVE ACTIONS.
27
28 The humans are only capable of performing the actions
    mentioned above.
29 User will provide a [Social context], a [Task] that the
    robot needs to do, a description of the location and
    optionally a [Rough Scenario].
30 Your generated scenario will be programmatically simulated
    through a pipeline into a scenario in the Gazebo
    physics simulator.
31 Rules:
32 - Describe human behavior by describing their [Human Task]
    and their [Behavior Towards Robot].
33 - The humans can say "WAIT", "PROCEED", "EXCUSE ME" to the
    robot to aid in navigation. The robot can say "WAIT",
    "PROCEED", "EXCUSE ME", "ACKNOWLEDGED" to the humans to
    aid in navigation.
34 - When the user provides a Rough Scenario, ensure your
    final scenario is strictly aligned to the rough
    scenario
35 - The humans in the simulator are SIMPLIFIED OBJECTS that
    only can move in 2D, send and receive simple phrases,
    detect and simulate looking at the robot, group
    together with other humans, navigate to a predefined
    goal and change their trajectory conditioned on the
    robot's position and velocity.
36 - When using groups in the scenario, add all group members
    to the humans in the scenario. Having only 1 human with
    'INTERACTING WITH GROUP' task is incorrect.
37
38 Design a scenario relevant to the following specifications:
39
40 [Social context]: Robot is a home assistant in a
    Singaporean old-age home and performs daily helpful
    duties for the residents
41 [Robot Task]: Deliver coffee
42 [Rough Scenario]: None
43 [Location]: The home has a Kitchen, a Bedroom and a
    Living Room. A doorway connects the Kitchen to the
    Living Room and a Passageway connects the Living Room
    and the Doorway.
44
45 YOU ADHERE TO THE FOLLOWING JSON FORMAT STRICTLY.
46 {
47     'Scenario Description': <very detailed description of the
        scenario >,
48     'Number of Humans': <Number of humans that are involved in
        the scenario>,
49     Human behavior': {
50         Human 1 : <Describe the behavior of Human 1>,
51         Human 2 : <Describe the behavior of Human 2>,
52     },
53     'Expected Robot Behavior': <Describe the behavior expected
        from the robot>
54 }
55
56 ASSISTANT:
57 {
58     'Scenario Description': "The robot is trying to deliver
        coffee from the Kitchen to the Living Room and
        encounters one of the elderly residents entering the
        Kitchen from the Living Room through the Doorway.",
        'Number of Humans': 1,
        'Human Behavior': {
        'Human 1': Human 1 is going from going to kitchen from
        the living room. If the robot is very close-by, Human
        asks the robot to stop and waits (for a maximum of 5s)
        for the robot to stop, then continues navigating.
        Ignores the robot if it asks the human to wait.
        },
        'Expected Robot Behavior': "The robot says "I AM HERE"
        to the resident. It waits for the resident to be well
        clear of the Doorway before going through the Doorway
        to the Living Room in a slow pace."
        }
59
60
61
62
63
64 }
65
66 USER:
67 Design a scenario relevant to the following specifications:
68
69 [Social Context]: <SOCIAL CONTEXT USER INPUT>
70 [Task]: <TASK USER INPUT>
71 [Location]: <LOCATION DESCRIPTION USER INPUT>
72 [Rough Scenario]: <ROUGH SCENARIO USER INPUT>
73
74 YOU ADHERE TO THE FOLLOWING JSON FORMAT STRICTLY. ""+""
75 {
76     'Scenario Description': <very detailed description of the
        scenario >,
77     'Number of Humans': <Number of humans that are involved in
        the scenario>,
78     Human behavior': {
79         Human 1 : <Describe the behavior of Human 1>,
80         Human 2 : <Describe the behavior of Human 2>,
81     },
82     'Expected Robot Behavior': <Describe the behavior expected
        from the robot>
83 }
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

23 'ANSWER': False
24 'REASONING': ""To determine if node "0e" is connected
    to node "07," we need to trace the paths in the
    provided edges.
25 Nodes:
26 "0e" at position [80, 660] (type: CORNER)
27 "07" at position [28, 318] (type: CORNER)
28
29 Edges involving "0e":
30 {"type": "PASSAGEWAY", "source": "0e", "target": "
    32"}
31 {"type": "PASSAGEWAY", "source": "0e", "target": "
    cb"}
32 Edges involving "07":
33 {"type": "NARROW PASSAGEWAY", "source": "11", "
    target": "07"}
34 {"type": "PASSAGEWAY", "source": "07", "target": "
    f7"}
35
36 There are no direct or indirect edges connecting "0e"
    to "07" through other nodes. Therefore, node "0e" is
    not connected to node "07".
37 }
38
39 ....
40
41 Let's initialize a simulator (like Gazebo) with the world
    given in the image and described by the json file and
    try to orchestrate the following human-robot scenario:
42
43 Robot is transporting a box from one point to another in
    the warehouse. While moving along a passageway, it
    encounters an employee walking in the opposite
    direction.
44
45 You must select the trajectories of the humans and the
    robot to orchestrate this scenario. You must also
    assign integer group ids to the HUMANS ONLY involved in
    the scenario according to the
46 following rule:
47 1. All the members of the same group must have the same
    group id.
48 2. If a human is not involved in a group, they will have a
    group id of -1.
49 3. If the scenario does not specifically mention a group,
    then assign -1 to all humans
50 4. All humans involved in the scenario must be assigned a
    group id.
51 Format your output in json as given below:
52 {
53     'REASONING': <Reasoning for Answer>,
54     'TRAJECTORIES': {
55         'ROBOT': <...>
56         'HUMAN 1': <...>,
57         'HUMAN 2': <...>,
58         'HUMAN 3': <...>,
59     },
60     'GROUP ID':{
61         'HUMAN 1': ..,
62         'HUMAN 2': ...,
63         ...
64     }
65 }
66 ASSISTANT:
67 'REASONING':""To create a scenario where robot is
    transporting a box from one point to another in the
    warehouse and encounters an employee walking in the
    opposite direction along a passageway, we need to
    identify a suitable passageway and ensure that robot
    and the employee move towards each other from opposite
    directions.
68
69 ### Scenario Setup:
70 1. robot starts at one node and moves along a passageway
    towards a destination node.
71 2. An employee starts at the destination node and moves
    along the same passageway towards robot.
72
73 ### Suggested Passageway:
74 We'll use the passageway between nodes "07", "f7", and "d9"
    again as it is a straightforward path that allows for
    clear movement in opposite directions.
75
76 ### robot's Trajectory:
77 1. **Start at node "f7"** (robot picks up the box here)
78 2. **Move to node "07"** (robot moves along the passageway)
79
80 ### Employee's Trajectory:
81 1. **Start at node "07"**
82 2. **Move to node "f7"** (The employee moves along the same
    passageway towards robot)
83
84 ### Detailed Node Sequence:
85
86 ##### robot's Nodes:
87 1. f7
88 2. 07
89
90 ##### Employee's Nodes:
91 1. 07
92 2. f7
93
94 ### Path Visualization:
95 - **robot's Path**: f7      07
96 - **Employee's Path**: 07      f7
97
98 ### Simulator Initialization:
99 - **robot** starts at node "f7" and follows the path f7
    07.
100 - **Employee** starts at node "07" and follows the path 07
    f7.
101
102 Since there is only one human in the scenario and the human
    is not involved in any groups, the group id for this
    human is the default -1.
103
104 In a simulator like Gazebo, you can set up the initial
    positions of robot and the employee at their respective
    starting nodes and let them move along the described
    paths. They will encounter each other along the
    passageway between nodes "f7" and "07".
105
106 ### Steps to Simulate:
107 1. Initialize robot at node "f7" with a box.
108 2. Initialize the employee at node "07".
109 3. Set both to move towards each other along the passageway
    .
110
111 This will create the scenario where robot, while
    transporting a box, encounters an employee walking in
    the opposite direction along the same passageway.""
112
113 'TRAJECTORIES': {
114     'ROBOT': ['f7','07']
115     'HUMAN 1': ['07','f7'],
116 },
117
118 'GROUP ID':{
119     'HUMAN 1': -1
120 }
121 ....
122
123 Now generate trajectories for the following scenario:
124 <SCENARIO DESCRIPTION FROM USER INPUT>
125 You must select the trajectories of the humans and the
    robot to orchestrate this scenario. You must also
    assign integer group ids to the HUMANS ONLY involved in
    the scenario according to the following rule:

```



```

126 1. All the members of the same group must have the same
    group id.
127 2. If a human is not involved in a group, they will have a
    group id of -1.
128 3. If the scenario does not specifically mention a group,
    then assign -1 to all humans
129 4. All humans involved in the scenario must be assigned a
    group id.
130 Format your output in JSON as given below:
131 {
132     'REASONING': <Reasoning for Answer>,
133     'TRAJECTORIES': {
134         'ROBOT': <...>
135         'HUMAN 1': <...>,
136         'HUMAN 2': <...>,
137         'HUMAN 3': <...>,
138     },
139     'GROUP IDS':{
140         'HUMAN 1': ..,
141         'HUMAN 2': ...,
142         ...
143     }
144 }

```

```

1 SYSTEM:
2 Act as an expert Behavior Tree Designer for a social
  navigation robotics simulator and an expert software
  engineer. You ALWAYS provide output that in JSON which
  is fully parseable with json.loads python.
3 You are diligent and tireless!
4 You NEVER leave comments describing code without
  implementing it!
5 You always COMPLETELY IMPLEMENT the needed code and do not
  leave placeholders or assume anything!
6
7 USER:
8 Your job is to design a Behavior Tree using the
  BehaviorTree.CPP library in XML according to the [
  BEHAVIOR] described by the user, using only the
  existing actions and conditions.
9 The tutorial below will explain how to design behavior
  trees in XML:
10
11 - The first tag of the tree is <root> with the attribute '
  main_tree_to_execute'. It should contain 1 or more tags
  <BehaviorTree> and the The tag <BehaviorTree> should
  have the attribute [ID]. E.g:
12 <root main_tree_to_execute = 'SaySomething'>
13 ....
14 <BehaviorTree ID="SaySomething">
15 ...
16 </BehaviorTree>
17 </root>
18 - The <BehaviorTree> tag can contain only 1 child node (
  including Fallback, Sequence, Action and Condition
  Nodes)
19 - Sequence and Fallback nodes contain 1 to N children and
  dictate control flow in the behavior tree.
20 - Sequence nodes execute their children in order and
  return Success only if ALL CHILD NODES returned SUCCESS
  . (like an AND gate).
21 - Each sequence node typically has multiple child
  nodes
22 - Fallback nodes execute their children in order and
  return Success if ANY CHILD NODE SUCCEEDS (like an OR
  gate).
23
24 - Each Sequence and Fallback TreeNodes are represented by a
  single tag with an associated name. E.g.:
25 <Fallback name="SaySomething 1">
26     <Sequence name = "SaySomething 2">
27         ....
28     </Sequence>

```

```

29 </Fallback>
30 JSOne">
31     <Sequence name="RegNav">
32         <SetBlackboard output_key="agentid" value="{id
  }" /> <!--Main tree creates a blackboard entry 'agentid
  ' with the value retrieved from the blackboard entry "
  id" -->
33         <SetBlackboard output_key="timestep" value="{dt
  }" /> <!--Main tree creates a blackboard entry '
  timestep' with the value retrieved from the blackboard
  entry "id" -->
34         <SubTree ID="RegularNavTree" id="agentid" dt="
  timestep" /> <!-- The "id" and "dt" ports of the
  subtree are mapped to the agentid and timestep ports of
  the main tree blackboard -->
35     </Sequence>
36 </BehaviorTree>
37
38 The following Action Nodes, Condition nodes and Decorators
  are available to use and can be composed into behavior
  trees to achieve the user's request.
39
40 - BT Conditions
41
42     - IsRobotVisible(agent_id ,distance) : returns
  Success when the robot is within the input distance of
  the agent and visible(in line of sight) to the agent.
  Distance is in meters.
43     - IsRobotNearby(agent_id, distance) : returns
  Success when the robot is within the input distance of
  the agent. Distance is in meters.
44     - IsGoalReached(agent_id) : returns Success if the
  agent has reached their current goal
45     - TimeExpiredCondition(seconds, ts, only_once):
  Creates a timer and returns SUCCESS if the input
  duration in seconds has expired since ticking this node
  the first time (and FAILURE otherwise). If only_once =
  False, then the timer repeats (use this for
  periodically occurring behaviors).
46     - RobotSays(agent_id, message) : Returns Success if
  the robot is currently performing a gesture
  corresponding to the message(int) passed to the
  functions. Messages correspond to gestures as: [0 (No
  gesture), 1("WAIT"), 2("PROCEED"),3("ACKNOWLEDGED"),4("
  EXCUSE ME")].
47     - RobotMoved(agent_id): Returns success if the
  robot has non zero velocity.
48     - IsRobotBlocking(agent_id, distance): Returns
  Success if the robot is in direct line of sight of the
  agent and within the input distance. Distance is in
  meters.
49
50 - Note: In the simulator, for distance, 0.5 is
  considered very close, 1.0 is considered very closed,
  2.0 is considered moderate distance and 5.0 is
  considered far
51
52 - BT Actions:
53     - UpdateGoal(agent_id) : Updates the goal of the
  agent to the next goal in the agents goal queue
54     - MakeGesture(agent_id,message): Makes the agent
  perform a gesture. Choices are: [0 (No gesture), 1("
  WAIT"), 2("PROCEED"), 3("EXCUSE ME")]. Initial value is
  0 and once this node is ticked, the agent will keep
  making the gesture until it is set back to 0.
55     - RegularNav(agent_id,time_step) : Makes the agent
  perform standard social-force-model based motion
  planning, where the robot is treated as a normal
  obstacle.
56     - LookAtRobot(agent_id) : Makes the agent look in
  the direction of the robot
57     - FollowRobot(agent_id,time_step): Makes the agent
  follow the robot
58     - AvoidRobot(agent_id,time_step): Makes the agent
  overly avoid the robot.
59     - GiveWaytoRobot(agent_id,time_step): Makes the
  agent give way to the robot.

```

```

57     - BlockRobot(agent_id,time_step): Makes the agent 115 }
    move in front of the robot and block it
58
59 - BT Decorators:
60     - Inverter: An inverter block inverts the output of
    its children (If children output failure, inverter
    outputs success and vice-versa).
61
62 The following behavior tree is available for including as a
    subtree, which implements a simple obstacle-avoiding
    human:
63     - BTRegularNav.xml:
64     <root main_tree_to_execute = "RegularNavTree">
65         <BehaviorTree ID="RegularNavTree">
66             <Fallback name="RegularNavFallback">
67                 <Sequence name="RegularNavigation">
68                     <Inverter>
69                         <IsGoalReached agent_id="{id}" />
70                     </Inverter>
71                     <RegularNav agent_id="{id}" time_step
   ="{dt}" />
72                 </Sequence>
73                 <UpdateGoal agent_id="{id}" />
74             </Fallback>
75         </BehaviorTree>
76     </root>
77 Now answer the following question:
78
79 If i want a human to gesture "WAIT" what should be the
    corresponding xml code
80
81 Return output in the following json format:
82 {
83     'REASONING': <Reasoning behind answer>
84     'ANSWER': <XML CODE>
85 }
86
87
88 ASSISTANT:
89 {
90     'REASONING': "To create a behavior tree where a human
    gestures "WAIT," you need to use the 'MakeGesture'
    action node with the message corresponding to the "WAIT
    " gesture. The message code for "WAIT" is 1.
91 In this behavior tree:
92 - '<root main_tree_to_execute="WaitGestureTree">' specifies
    the main tree to execute.
93 - '<BehaviorTree ID="WaitGestureTree">' defines a behavior
    tree with the ID "WaitGestureTree."
94 - '<Sequence name="WaitGestureSequence">' creates a
    sequence node named "WaitGestureSequence."
95 - '<MakeGesture agent_id="{id}" message="1"/>' makes the
    agent perform the "WAIT" gesture (message code 1).
96 "
97     'ANSWER': "<root main_tree_to_execute="WaitGestureTree
    ">
98     <BehaviorTree ID="WaitGestureTree">
99         <Sequence name="WaitGestureSequence">
100             <MakeGesture agent_id="{id}" message="1"/>
101         </Sequence>
102     </BehaviorTree>
103 </root>
104 "
105 }
106 ...
107 Now, create a behavior tree for the following behavior:
108 <BEHAVIOR DESCRIPTION FROM SCENARIO PROPOSAL>
109
110 YOU MUST NOT USE ANY CUSTOM ACTION/CONDITION NODES AND ONLY
    USE THE NODES AVAILABLE.
111 Return an answer in JSON format shown below:"""
112 {
113     'REASONING': <reasoning behind tree design>,
114     'TREE': <XML Behavior Tree ONLY>,

```