A Modular Framework for Socially Compliant Robot Navigation in Complex Indoor Environments

Sara Pohland*[1,2], Alvin Tan*[1], Corban Rivera[2], I-Jeng Wang[2], Prabal Dutta[1], Claire Tomlin[1]

***Abstract*— An important challenge in developing assistive robots is the design of socially compliant robot navigation policies that enable safe and comfortable movement around people. Previous works demonstrate that deep reinforcement learning (DRL) methods are effective in developing such navigation strategies. However, existing DRL policies are designed for simple, open-space environments, and through our experiments, we find that such policies do not generalize well to more realistic environments containing walls and other stationary objects. In this paper, we present a modular approach to social navigation in complex environments, in which we combine a DRL policy with a global path planner and a deterministic safety controller. By designing each component of the modular architecture to handle different, and potentially conflicting, navigation objectives, we divide the indoor navigation problem into logically distinct and manageable steps. This allows us to extend the applicability of existing DRL policies to complex indoor spaces. When compared against a traditional navigation technique that does not employ learning, we find that our approach results in fewer collisions, comparable navigation times, and higher success rates when guiding a robot through complex environments. We also implement our approach on a physical robot, which successfully navigates indoor spaces containing various humans and stationary objects. Our results demonstrate the value of using learning-based methods as a single component in a larger framework to develop socially compliant robot navigation policies that are effective in real-world settings.**

I. INTRODUCTION

Socially assistive robots (SARs) provide assistance to human users through social interaction [1]. Such robots are particularly helpful in assisting elderly populations, individuals in convalescent care, individuals with physical impairments, and individuals with social and mental disorders in hospitals, assisted living homes, and schools [1]. To effectively employ SARs in any of these environments, these robots need to be equipped with socially compliant navigation algorithms that enable them to traverse complex indoor spaces while maintaining personal safety and comfort.

There are currently four main approaches to socially compliant navigation: human trajectory models, imitation learning, inverse reinforcement learning (IRL), and deep reinforcement learning (DRL). The first approach attempts to explicitly model the trajectories of pedestrians in the environment and plan a path around these trajectories. While this method works well in settings with very few pedestrians, it does not scale well to environments with heavy pedestrian traffic [2]. Approaches that use imitation learning or IRL produce navigation strategies from implicit or explicit human feedback. These approaches can generate robot behavior that mimics a human demonstrator, but a lack of negative examples, such as collisions, can lead to a model that does not consider comfort or safety [3].

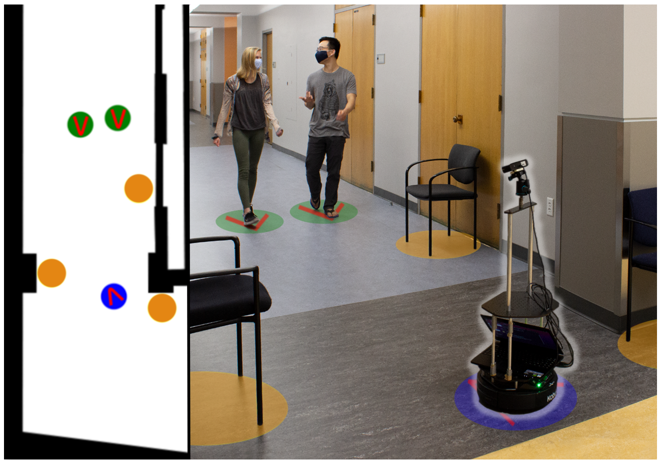
Fig. 1: TurtleBot navigating both simulated and real-world environments with humans and stationary objects, using a modular framework for socially compliant robot navigation.

With these considerations in mind, we believe DRL is a promising direction to explore for socially compliant robot navigation. Recently there has been substantial progress in designing navigation policies that use DRL to effectively navigate crowded environments while maintaining human social norms [4], [5], [6], [7], [8], [2], [9], [10]. However, such algorithms are designed for simple, open-space environments that do not contain walls or other stationary obstacles. We find that as a result, such algorithms are unable to generalize to more complex and realistic environments.

To extend the use of socially compliant DRL policies to complex indoor environments, we propose a modular framework that uses a DRL policy as a single component of a larger system. Our proposed framework consists of a socially-aware DRL policy, a global path planning algorithm, and a custom safety controller. We demonstrate the utility of our approach by navigating both a simulated robot and a physical robot through complex indoor spaces (Figure 1). Our contributions are:

- A modular framework that extends the applicability of socially-aware DRL policies to diverse environments without retraining;
- A socially-aware DRL policy that considers both dy-

* Both authors contributed equally to this work.
[1] Dept. of EECS, UC Berkeley, Berkeley, CA, USA.
[2] Johns Hopkins University Applied Physics Lab, Laurel, MD, USA.

namic humans and stationary objects;

- Results in simulated and real environments that demonstrate the efficacy of our approach; and
- Software extensions to the CrowdNav simulation [2] to support further complex indoor navigation research.

II. BACKGROUND & RELATED WORK

Our modular approach seeks to combine recent advances in DRL robot navigation algorithms with traditional navigation techniques to realize the benefits of both methods.

A. DRL-Based Socially Compliant Navigation

When using DRL, a robot learns a policy that maximizes its expected accumulation of a developer-designed reward through trial and error. In the case of socially compliant navigation, this approach generally encodes comfort criteria into the reward function to encourage the robot to learn behaviors that avoid annoyance and stress for humans [11]. In particular, a socially compliant robot learns to maintain enough distance between itself and nearby humans to prevent discomfort. Chen et al. developed a popular approach to DRL-based socially compliant navigation that used an attentive pooling mechanism to learn the collective importance of neighboring humans with respect to their future states [2]. Through various experiments, this approach proved to be more effective than competing methods. Despite its success, when we performed additional experiments, we found that this algorithm does not generalize to environments with walls and other stationary objects. To the best of our knowledge, there are currently no straightforward methods to incorporate useful information about walls into training or balance competing near-term and long-term navigation objectives that are present in more complex spaces.

B. Standard Navigation Techniques

The traditional approach to robot navigation uses path planning algorithms to find safe and efficient paths between the robot's initial and goal positions while avoiding collisions with obstacles [12]. Path planning algorithms can be divided into two categories, which differ in their time horizon and assumptions about the environmental information acquired during navigation [13]. The first category is global planners, which use a static global map to generate a fixed, collision-free path from the robot's initial position to its goal. There are many global planners, but one popular algorithm is the probabilistic roadmap (PRM) planner, which constructs a map of feasible paths between collision-free points, then searches this map for a path joining the initial and goal positions [14]. The second category is local planners, which use knowledge of static and dynamic obstacles to generate shorter paths that change during navigation. Arguably, the most effective local planner is the optimal reciprocal collision avoidance (ORCA) planner, which considers the reactive behavior of other agents and the acceleration constraints of the robot to generate maneuvers that avoid collisions [15]. Typically, a local and global planner are used together to navigate robots through complex spaces with static and dynamic obstacles. While this navigation approach is generally effective, it does not necessarily generate socially compliant behavior.

III. PROBLEM FORMULATION

Our objective is to enable a robot to navigate complex indoor spaces while avoiding collisions and limiting close encounters with humans. Because SARs generally operate in familiar spaces, we assume the robot has access to a map of the walls in its environment during navigation and that it can locate and orient itself within its environment. We also assume that the robot can identify nearby obstacles and distinguish between static objects and dynamic humans. We only consider cases where the robot has a clear path from its initial position to its goal and assume that this path is never completely blocked for extended periods of time.

A. Observation Space

We assume that the robot has access to its full state but only knows part of the state of each human and object in its environment. The observable state of the robot, the ith human, and the jth object are represented by $o^{(r)}$, $o^{(h_i)}$, and $o^{(o_j)}$ respectively and are defined as follows:

$$o^{(r)} = \begin{bmatrix} p_x^{(r)} & p_y^{(r)} & v_x^{(r)} & v_y^{(r)} & r^{(r)} & g_x^{(r)} & g_y^{(r)} & v_{pref}^{(r)} & \theta^{(r)} \end{bmatrix}$$

$$o^{(h_i)} = \begin{bmatrix} p_x^{(h_i)} & p_y^{(h_i)} & v_x^{(h_i)} & v_y^{(h_i)} & r^{(h_i)} & 1 \end{bmatrix}$$

$$o^{(o_j)} = \begin{bmatrix} p_x^{(o_j)} & p_y^{(o_j)} & v_x^{(o_j)} & v_y^{(o_j)} & r^{(o_j)} & 0 \end{bmatrix}$$

where (p_x, p_y) is the current position, (v_x, v_y) is the velocity, r is the radius, (g_x, g_y) is the goal position, v_{pref} is the preferred speed, and θ is the turning angle of the agent. The radius is measured assuming each agent can be represented by a circle. The preferred velocity is the maximum allowable speed of each agent, which is the speed at which it would travel if nothing is obstructing its path. The last value of the observable state for humans and objects is a flag that allows the robot policy to distinguish between these agents.

To make our policy viable for real-world environments, the robot's observations are restricted to those that could be realistically obtained. Assuming a 360° field of view with a four-meter detection range, only humans and stationary objects that are within this range are visible to the robot. The robot also cannot view humans and objects that are obstructed by another human, object, or wall. If N humans and M objects are visible to the robot, its observation is:

$$o = \begin{bmatrix} o^{(r)} & o^{(h_1)} & \dots & o^{(h_N)} & o^{(o_1)} & \dots & o^{(o_M)} \end{bmatrix}.$$

B. Action Space

Our policy outputs a velocity command that controls the robot. The action space is discretized into 5 speeds exponentially spaced in the range $(0, v_{pref}^{(r)}]$ and 16 directions evenly spaced in the range $[0, 2\pi)$. The robot is also able to receive a stop command, resulting in 81 possible actions.

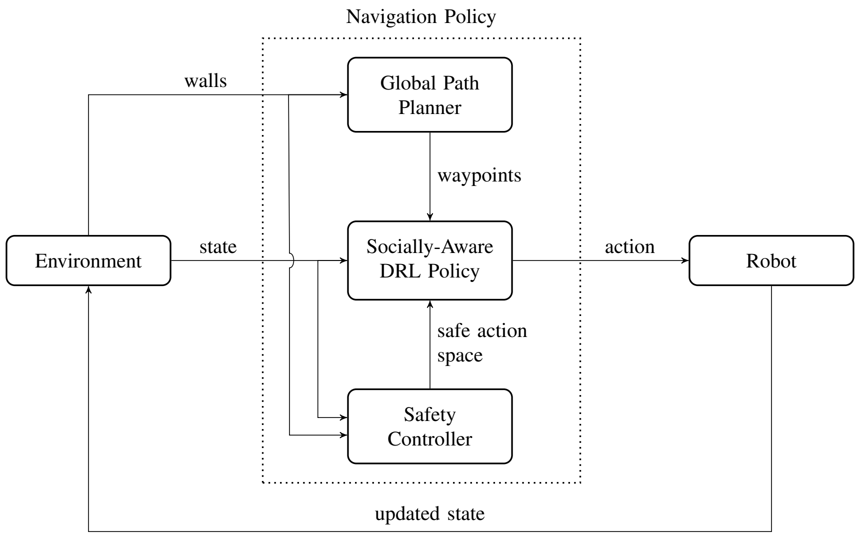

Fig. 2: Our policy is composed of three components: a socially-aware DRL policy, a global path planning algorithm that constructs waypoints between the robot and its goal, and a custom safety controller that avoids wall collisions by temporarily limiting the action space of the robot. An optimal action for the robot is chosen using these three components.

IV. PROPOSED POLICY ARCHITECTURE

While DRL algorithms are effective in designing socially compliant robot navigation strategies, previously developed algorithms are not designed to operate in complex indoor environments with walls and other stationary objects. Conversely, standard navigation techniques enable robots to navigate in complex environments but are not designed to generate socially compliant behavior. We combine the benefits of these approaches by using the modular architecture shown in Figure 2, which consists of three main components: a socially-aware DRL policy, a global path planning algorithm, and a custom safety controller. We train a DRL policy that uses knowledge of humans and small stationary objects, such as chairs, to develop a socially compliant local planner. We use a global path planning algorithm to generate waypoints used as local goals by the DRL policy. We design a separate safety controller that uses knowledge of walls to determine the set of safe actions from which the DRL policy selects the optimal action.

We chose this modular architecture in order to leverage the benefits of socially-aware DRL policies without overloading the capabilities of these algorithms. The most effective existing DRL policies focus on near-term interactions with dynamic humans within close proximity of the robot. A simple extension of these algorithms enables robots to consider interactions with small stationary objects as well. Conversely, global path planners address longer-horizon considerations driven by the environment layout and the robot's ultimate goal. Relying on a single DRL policy to address both these goals simultaneously would likely lead to an undesirable trade-off between these objectives. Thus, combining a DRL policy with a global path planner does not seem unreasonable. Additionally, it is difficult to incorporate useful information about the environment layout into the DRL policy effectively. This creates the need for a separate safety controller designed specifically to avoid collisions with walls. For these reasons, we propose a modular framework that can leverage the strengths of existing socially-aware DRL policies without placing too many additional demands on these algorithms.

A. Socially-Aware DRL Policy

The DRL policy was trained in a simulation environment built on OpenAI Gym. Each time a training simulation is run, a square room is generated with a single robot at one edge of the room whose goal is to move to the opposite side. The number of humans and objects and their physical attributes are randomly sampled from uniform distributions. Each object is stationary and is given a random fixed position. Each human is given a random starting position and goal within its field of view, and its velocities are controlled using the ORCA policy [15]. Once the human has reached its goal, it receives a new goal, so that humans are perpetually moving. The robot is always visible to each human, assuming their 360° view is not obstructed, and the simulated humans treat the robot as another human operating under the same policy. After initialization, the robot chooses an action according to its policy at each time step and each human chooses an action according to its policy until the robot reaches its goal, runs out of time, or collides with an obstacle.

To design a policy that enables a robot to reach its goal, while avoiding collisions and maintaining human social norms, we use the following reward function:

$$\begin{aligned} r = H(0.3 - d_{goal}) - 0.15H(-d_{obj}) \\ - 0.25H(-d_{hum}) + 0.5d_{hum}H(0.1 - d_{hum}). \end{aligned}$$

In this function, d_{goal} is the distance from the robot to the goal, d_{obj} is the distance from the robot to the closest object, d_{hum} is the distance from the robot to the closest human, and

MLP	Layers
1	Linear(14, 150) → ReLU → Linear(150, 100) → ReLU
2	Linear(100, 100) → ReLU → Linear(100, 50)
3	Linear(200, 100) → ReLU → Linear(100, 100) → ReLU → Linear(100, 1)
4	Linear(56, 150) → ReLU → Linear(150, 100) → ReLU → Linear(100, 100) → ReLU → Linear(100, 1)

TABLE I: The four MLPs in the value network used for the DRL policy are composed of Rectified Linear Unit (ReLU) activation functions and linear layers, whose input and output sizes are as shown.

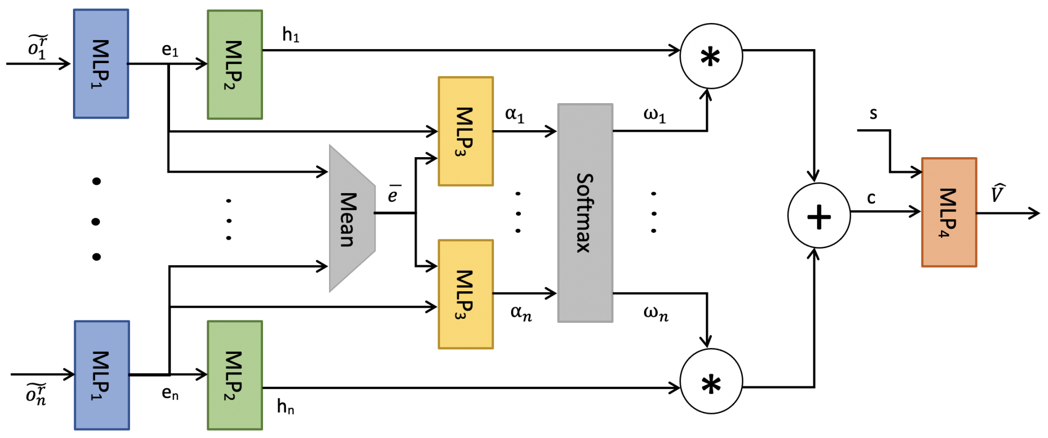

Fig. 3: Given observation $\tilde{o}^r$, MLP_1 is used to obtain an embedding vector e_i, which models the pairwise interaction between the robot and each of the n humans and objects. This embedding vector is passed through MLP_2 to obtain the pairwise interaction feature h_i for each human or object. A self-attention mechanism passes each of the embedding vectors e_i, along with their mean $\bar{e}$, into MLP_3 to determine the attention score α_i, which reflects the relative importance of each human/object. The softmax function is then applied to each attention score to obtain a normalized set of weights ω_i. A compact representation c of the entire set of humans and objects is obtained by computing a linear combination of the pairwise interaction features. The representation c has a fixed size, regardless of the number of humans and objects, and can be passed into MLP_4, along with the robot's state s, to obtain an estimate of the value function $\hat{V}$.

$H(\cdot)$ is the Heavyside step function. The first term rewards the robot for reaching the goal within a nominal 0.3-meter radius, the second penalizes the robot for colliding with a stationary object, the third penalizes the robot for colliding with a human, and the fourth penalizes the robot for getting within an uncomfortable distance (0.1 meters) of a human.

We use a value network algorithm to generate our DRL policy, in which observations are fed into a neural network to generate a model of the value function. The value network architecture is shown in Figure 3. This network is composed of four sets of multilayer perceptrons (MLPs) whose layers are given in Table I. Before the observation o is passed into the neural network, the observation is first transformed and rotated to a robot-centric coordinate frame to obtain a new observation $\tilde{o}^r$. Each row of $\tilde{o}^r$ reflects the interaction of the robot and a single human or object.

After generating a model of the value function, this model is used to design a policy expected to result in high rewards. Given the current state of the environment, the expected next state is determined for each action in the discrete action space, using a simple model to approximate the motion of the robot and humans. The next state is then used as input to the value network to determine the value associated with each discrete action and to choose the optimal action.

Comparing against existing socially-aware DRL policies, we find that our policy is more effective at dealing with static humans and objects, as well as humans that suddenly change direction. This is because previous DRL policies are trained with the assumption that humans are constantly moving towards a fixed goal. By removing this assumption, our policy is better suited to operate in real-world environments.

B. Global Path Planner

The PRM planner is used to generate a path from the robot's starting position to its goal position, while avoiding walls specified by the map of the environment. This planning algorithm relies on a network graph whose nodes are unoccupied points in the environment and whose edges are collision-free paths between these points [14].

After finding a collision-free path from the robot's initial position to its goal, we use this path to generate waypoints for the robot. Waypoints are chosen to be more dense around corners and doorways and less dense in open spaces to give the robot more guidance in challenging maneuvers and more freedom when the navigation task is simple. Based on the robot's current position, its local goal is chosen from the list of waypoints. Initially, its local goal is the first waypoint after its starting position. Once the robot has entered within a one-meter radius of this waypoint, its local goal is set to the next waypoint in the sequence. This process continues until the robot's local goal is its final goal. The PRM planner uses knowledge of walls in the robot's environment but does not consider other objects, which may not be fixed for the robot's lifetime. For this reason, it is possible for a local goal to be placed in the same location as a stationary object. To deal with this issue, if the next waypoint in the robot's path is in the same location as an object, the robot's local goal is adjusted to be on either side of the obstacle.

C. Safety Controller

Rather than generating a representation of walls that can be passed into the DRL policy, wall avoidance is handled by limiting the action space of the policy. At each time step, before deciding the optimal action with the DRL policy, the safety controller first determines which actions in the full action space are expected to result in a collision with a wall. The DRL policy then computes the value of each action among the set of actions that are not expected to result in a wall collision, and the action with the largest value among those safe actions is provided to the robot.

V. EXPERIMENTAL EVALUATION

We designed 17 environment configurations of varying sizes and complexities, chosen to reflect the majority of possible scenarios a SAR may encounter when navigating a hospital, assisted living home, or school, where these robots typically operate. These configurations are shown in Figure 4. 100 trials were performed within each of these environment configurations. In our analysis, we grouped the 17 configurations into five qualitatively distinct sections to better evaluate our approach. These five sections in roughly increasing difficulty are:

- *Open Space*: the goal is 10m away from the starting position with walls only on the boundary.
- *Hallways*: the goal is further from the starting position but can still be reached by a straight path.
- *Intersections*: the robot must navigate around a single corner at a four-way intersection to reach the goal.
- *Doorways*: the robot must navigate through doorways.
- *Corners*: the robot must move around multiple corners.

A. Evaluation Metrics

We evaluated various policies based on robot navigation performance and social compliance. For robot navigation performance, the metrics represent the quality of the robot's ability to navigate to the goal quickly without collision. To assess social compliance, we quantified how the robot maintained distance among humans. Specifically, we measured:

- *Success Rate*: the percentage of trials in which the robot successfully reaches its goal within 100 seconds.
- *Collision Rate*: the percentage of trials in which the robot collides with a human, an object, or a wall.
- *Navigation Time Increase*: the percentage of additional time required to navigate to the goal, relative to the baseline, compared across mutually successful trials.
- *Average Speed*: the average speed of the robot in successful trials.
- *Average Distance*: the average distance between the robot and the closest human in successful trials.
- *Minimum Distance*: the closest the robot gets to any human, collected across successful trials.
- *Discomfort Time*: percentage of time spent in a discomfort zone (0.1m-radius circle around each human).

B. Controller Components & Performance

As described previously, the robot controller we designed is composed of a socially-aware DRL policy, a global path planner used to generate waypoints, and a safety controller designed to limit collisions with walls. Table II helps demonstrate how each component impacts the overall performance.

The DRL policy by itself performs well in the open space environment, which is similar to the environment used while training. However, as the environment gets more complicated, the success rate quickly drops, indicating that the policy is unable to generalize to more complex layouts, in which the robot must adjust for walls and perform more complicated maneuvers around corners and doorways.

Adding a global path planner to guide the robot dramatically increases the success rate in moderately complex environments. However, since the DRL policy has no knowledge of walls in its environment, there is still a nontrivial number of wall collisions in the most complex environments.

Using our safety controller to limit the DRL action space reduces the wall collision rate to zero in all environment configurations and improves the success rate in the most complex environments. By recording the number of times the safety controller adjusted the action that the DRL policy would have chosen, we find that the safety controller is almost never active in open spaces and hallways. It is active around 1.5% of the time in intersection environments, 4.0% of the time in environments with doorways, and 2.1% of the time in environments with multiple corners. These values are very low, which indicates that the DRL policy is able to take the socially optimal action the vast majority of the time.

C. Learning & Performance

A key component of our robot controller is the DRL policy, which is designed to enable socially compliant behavior. It is interesting to consider how learning can enable more effective robotic behavior around crowds of humans and stationary objects. To demonstrate the impact of learning, we compare our full modular policy to a baseline policy that uses ORCA as a local planner and the PRM algorithm as the global planner. Because ORCA is a popular and effective local planner that does not rely on deep learning, this controller serves as a good baseline to analyze the impact of learning on social compliance and navigation performance.

As seen in Table II, our approach generally provides higher success rates and lower object collision rates than the baseline. This difference becomes progressively more apparent as the environment gets more complex and more reflective of real-world spaces. By individually inspecting successful

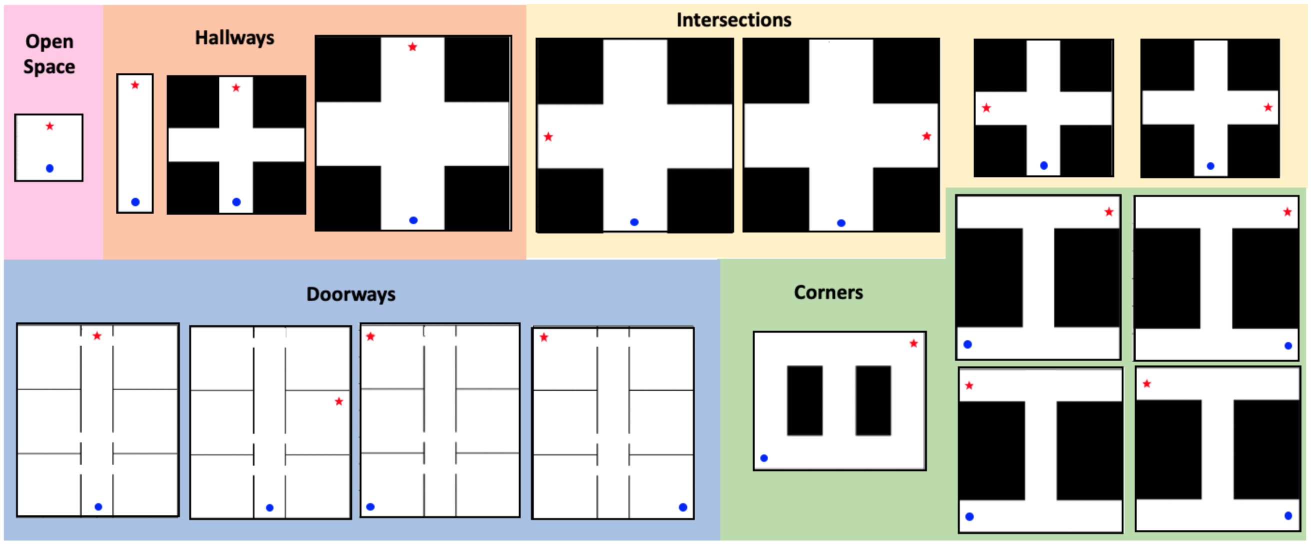

Fig. 4: Seventeen environment configurations of varying sizes and complexities used for evaluation. For each configuration, walls are shown in black, the blue circle is the robot's initial position, and the red star is the robot's goal position. Randomized obstacles are generated and added to the environments for each trial.

TABLE II: Aggregate results from 1700 trials split into five groups describing progressively more complex environments. We evaluate the utility of each component in our approach through an ablation study and collect metrics on robot navigation performance and social compliance. We compare our approach to a non-learning-based baseline policy comprised of the ORCA local planner and the PRM global planner. The bold numbers reflect the best performance for each metric.

	Navigation Policy	Success Rate (%)	Collision Rate (%)			Nav. Time Increase (%)	Average Speed (m/s)	Average Dist. (m)	Minimum Dist. (m)	Discomfort Time (%)
			Human	Object	Wall					
Open	DRL	**100.0**	0.0	**0.0**	0.0	4.3 ± 7.5	**1.00** ± 0.00	**1.58** ± 0.74	0.08	**0.0** ± 0.3
	DRL, Planner	**100.0**	0.0	**0.0**	0.0	0.5 ± 3.1	**1.00** ± 0.00	**1.58** ± 0.74	**0.10**	**0.0** ± 0.0
	DRL, Planner, Safety	**100.0**	0.0	**0.0**	0.0	0.5 ± 3.1	**1.00** ± 0.00	**1.58** ± 0.74	**0.10**	**0.0** ± 0.0
	Baseline	99.0	0.0	1.0	0.0	**0.0**	0.98 ± 0.06	1.57 ± 0.73	0.04	0.2 ± 2.1
Hallways	DRL	33.3	0.0	**0.0**	66.3	18.2 ± 15.4	0.99 ± 0.03	1.89 ± 1.26	**0.04**	**0.0** ± 0.1
	DRL, Planner	**99.7**	0.0	**0.0**	**0.0**	1.5 ± 3.2	**1.00** ± 0.00	**1.97** ± 1.40	0.00	0.1 ± 0.5
	DRL, Planner, Safety	**99.7**	0.0	**0.0**	**0.0**	1.5 ± 3.1	**1.00** ± 0.00	**1.97** ± 1.40	0.00	0.1 ± 0.6
	Baseline	97.3	0.0	1.3	**0.0**	**0.0**	0.99 ± 0.05	1.96 ± 1.41	0.02	0.2 ± 0.9
Intersects	DRL	1.8	0.0	**0.0**	98.0	7.7 ± 8.3	0.99 ± 0.02	2.10 ± 1.38	**0.20**	**0.0** ± 0.0
	DRL, Planner	**99.8**	0.0	**0.0**	0.2	1.3 ± 2.3	**1.00** ± 0.00	**2.21** ± 1.45	0.04	**0.0** ± 0.3
	DRL, Planner, Safety	**99.8**	0.0	**0.0**	**0.0**	2.1 ± 6.3	**1.00** ± 0.01	**2.21** ± 1.46	0.04	**0.0** ± 0.2
	Baseline	97.2	0.0	1.8	**0.0**	**0.0**	0.99 ± 0.05	2.18 ± 1.44	0.03	0.2 ± 1.4
Doorways	DRL	0.5	**0.0**	**0.0**	99.5	3.5 ± 2.8	**1.00** ± 0.00	1.69 ± 1.13	**0.24**	**0.0** ± 0.0
	DRL, Planner	96.8	0.5	**0.0**	2.0	1.2 ± 4.2	**1.00** ± 0.00	2.04 ± 1.41	0.02	0.1 ± 0.3
	DRL, Planner, Safety	**99.0**	**0.0**	**0.0**	**0.0**	2.5 ± 9.4	0.99 ± 0.05	2.05 ± 1.41	0.02	0.1 ± 0.3
	Baseline	93.0	0.5	3.5	0.8	**0.0**	0.99 ± 0.06	**2.07** ± 1.44	0.01	0.1 ± 0.8
Corners	DRL	0.0	**0.0**	**0.0**	99.8	–	–	–	–	–
	DRL, Planner	95.4	0.2	0.2	3.2	2.4 ± 7.0	**1.00** ± 0.00	2.21 ± 1.73	0.00	**0.1** ± 0.5
	DRL, Planner, Safety	**98.6**	0.2	**0.0**	**0.0**	3.8 ± 9.5	0.99 ± 0.03	**2.22** ± 1.75	0.00	**0.1** ± 0.6
	Baseline	92.2	**0.0**	4.0	**0.0**	**0.0**	0.99 ± 0.05	2.18 ± 1.72	**0.03**	0.2 ± 0.7

and unsuccessful trials, we find that the improvement in success rate is a result of our DRL policy's ability to learn more difficult maneuvers. In the majority of cases where our learning-based controller is successful and the baseline controller is unsuccessful, the ORCA policy is unable to maneuver around clumps of stationary objects. The robot controlled by the baseline controller either attempts to move around the group of objects and hits one, or it gets stuck at the group of objects and runs out of time.

Our approach also maintains a larger distance from humans in the environment on average and spends less time in people's discomfort zones, compared to the baseline. However, this difference is not very large. Both policies maintain a reasonable distance from humans, typically intruding on discomfort zones for less than 0.2% of the time. Thus, our modular policy is able to successfully navigate more scenarios than the baseline while remaining social compliant.

Although our approach results in longer navigation times, this increase is less than 4% on average for all environments. Using our policy, the robot maintains a velocity close to its preferred velocity of one meter per second, indicating that the robot chooses slightly longer paths to avoid collisions,

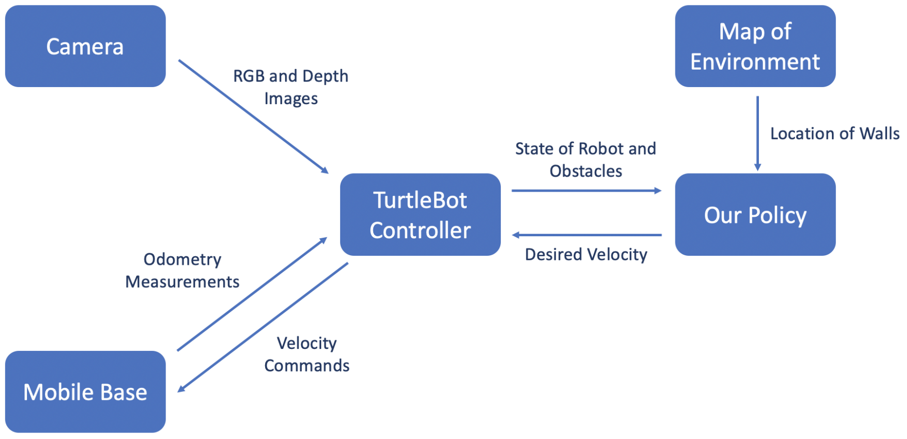

Fig. 5: The TurtleBot controller communicates with the camera, mobile base, and our policy to control the robot. This figure depicts the messages that are sent between these components during the hardware demonstrations.

compared to the baseline. This small increase in path length significantly increases success rates, suggesting that this is a fair trade-off and that our policy outperforms the baseline in terms of navigation performance.

VI. HARDWARE DEMONSTRATIONS

To demonstrate the applicability of our modular approach to real-world systems, we implement our algorithm on a physical robot. For these demonstrations, we use a TurtleBot kit, which includes a Kobuki mobile base, an Orbbec Astra camera, and a Gigabyte laptop computer. To control the TurtleBot, we design a controller that receives odometry measurements from the mobile base and receives color and depth images from the camera to determine the state of the robot and obstacles. This information is fed into our modular policy architecture, along with predefined information about the walls in the environment, to determine the desired velocity of the robot. The TurtleBot controller receives this velocity and sends the appropriate command to the mobile base to move the robot. This entire process is summarized in Figure 5.

A. *Obstacle Detection & Tracking*

To determine the position and radius of each obstacle in the robot's field of view, we use the "You Only Look Once" (YOLO) object detection system [16]. This system passes a color image through a neural network to generate bounding boxes for each human and object in the image. We use the bounding boxes obtained from YOLO in combination with the depth image to determine the distance of each object from the robot. We then find the angle between the obstacle and robot, which we use with the distance measurement to determine the obstacle's position. We estimate the radius of the obstacle based on its width in the image and its distance from the robot.

The TurtleBot only has a front-facing camera, so its visibility is considerably limited, presenting a need to track detected obstacles. Every time the robot observes a new obstacle, it stores the obstacle's observable state in memory. If a detected obstacle intersects with the expected position of an existing obstacle, these obstacles are assumed to be the same. When the robot detects an obstacle it has already seen, the obstacle's state is updated based on the state stored in memory and the state determined from the current image. Based on the number of times an obstacle was detected, it remains in memory for some time after it was last seen.

B. *Real-World Performance*

We conduct an ablation study to demonstrate how the robot behaves with and without the global path planner, as well as with and without the safety controller in a real-world environment. We set up stationary objects in a hallway such that all three components of our modular policy would be actively engaged at various points along the robot's path through the hallway. We find that the DRL policy by itself fails to navigate this crowded indoor space and frequently collides with walls. Adding the global path planner and safety controller adjusts the robot's behavior such that the full modular policy successfully traverses the hallway. Qualitatively, the global planner improves the maneuverability of the robot around stationary objects, while the safety controller improves behavior around walls. To show the performance of our approach in other complex environments, we set up situations where the robot uses our policy to navigate around corners and through hallways and doorways, in the presence of both stationary objects and moving people. Videos of these trials can be found in the supplementary video attachment.

VII. CONCLUSIONS

We demonstrate the value of using standard robot navigation methods to enable the effective usage of DRL navigation policies for socially compliant robot navigation in complex indoor spaces. In particular, we combine a socially-aware DRL policy with a global path planning algorithm and a custom safety controller in a modular architecture. Using all three components, a simulated robot could reach its goal from an arbitrary starting position, while limiting close encounters with humans and avoiding collisions with humans, objects, and walls. Compared to a baseline navigation policy that does not use a learning-based component, our policy resulted in fewer collisions, comparable navigation times, and higher success rates when guiding the robot through complex environments. Overall, our modular policy outperforms the baseline in terms of navigation performance and successfully navigates more scenarios than the baseline while remaining socially compliant. We also demonstrated the real-world applicability of our approach on a physical robot operating in complex indoor spaces.

Our work presents several directions for future research. To account for more real-world complexities, it could be useful to explore cases where there are multiple paths a robot could take to its goal, and to consider ways to select an optimal path and adapt if the path is completely blocked. To deploy SARs in safety-sensitive environments, it is necessary to consider worst-case robotic behavior and to incorporate probabilistic safety guarantees for socially compliant navigation. Another potential area for future work is in designing waypoints for task planning, where a robot has multiple tasks to complete in several locations. More generally for DRL policies, work could be done in designing reward functions to capture different methods of maintaining comfort around humans. For instance, robots could slow down around humans or avoid approaching humans from behind. Lastly, beyond the comfort aspect of socially compliant behavior, work could also be done on enhancing a robot's sociability by adhering to higher-order social norms, such as walking on a particular side of the hallway to avoid oncoming pedestrians.

ACKNOWLEDGMENTS

This work was supported in part by the Johns Hopkins University Institute for Assured Autonomy; and by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. Special thanks to Jared Markowitz, Chien-Ming Huang, Kapil Katyal, Yuxiang Gao, Matt Podolsky, Meghan Clark, Jean-Luc Watson, Shishir Patil, Tess Despres, Sean Zhang, Victor Zhang, and Varun Tolani for their advice and assistance throughout this project.

REFERENCES

[1] D. Feil-Seifer and M. Mataric, "Socially Assistive Robotics," in *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.* Chicago, IL, USA: IEEE, 2005, pp. 465–468. [Online]. Available: http://ieeexplore.ieee.org/document/1501143/

[2] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-Robot Interaction: Crowd-aware Robot Navigation with Attention-based Deep Reinforcement Learning," *arXiv:1809.08835 [cs]*, Feb. 2019, arXiv: 1809.08835. [Online]. Available: http://arxiv.org/abs/1809.08835

[3] C.-E. Tsai and J. Oh, "NaviGAN: A Generative Approach for Socially Compliant Navigation," *arXiv:2007.05616 [cs]*, Jul. 2020, arXiv: 2007.05616. [Online]. Available: http://arxiv.org/abs/2007.05616

[4] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized Non-communicating Multiagent Collision Avoidance with Deep Reinforcement Learning," *arXiv:1609.07845 [cs]*, Sep. 2016, arXiv: 1609.07845. [Online]. Available: http://arxiv.org/abs/1609.07845

[5] M. Everett, Y. F. Chen, and J. P. How, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning," *arXiv:1805.01956 [cs]*, May 2018, arXiv: 1805.01956. [Online]. Available: http://arxiv.org/abs/1805.01956

[6] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," *arXiv:1709.10082 [cs]*, May 2018, arXiv: 1709.10082. [Online]. Available: http://arxiv.org/abs/1709.10082

[7] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially Aware Motion Planning with Deep Reinforcement Learning," *arXiv:1703.08862 [cs]*, May 2018, arXiv: 1703.08862. [Online]. Available: http://arxiv.org/abs/1703.08862

[8] Y. Chen, C. Liu, M. Liu, and B. E. Shi, "Robot Navigation in Crowds by Graph Convolutional Networks with Attention Learned from Human Gaze," *arXiv:1909.10400 [cs]*, Sep. 2019, arXiv: 1909.10400. [Online]. Available: http://arxiv.org/abs/1909.10400

[9] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dube, "Robot Navigation in Crowded Environments Using Deep Reinforcement Learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 5671–5677. [Online]. Available: https://ieeexplore.ieee.org/document/9341540/

[10] K. Katyal, Y. Gao, J. Markowitz, I.-J. Wang, and C.-M. Huang, "Group-Aware Robot Navigation in Crowded Environments," *arXiv:2012.12291 [cs]*, Dec. 2020, arXiv: 2012.12291. [Online]. Available: http://arxiv.org/abs/2012.12291

[11] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, Dec. 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889013001048

[12] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A Survey of Path Planning Algorithms for Mobile Robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, Aug. 2021. [Online]. Available: https://www.mdpi.com/2624-8921/3/3/27

[13] K. Cai, C. Wang, J. Cheng, C. W. De Silva, and M. Q.-H. Meng, "Mobile Robot Path Planning in Dynamic Environments: A Survey," *arXiv:2006.14195 [cs]*, Mar. 2021, arXiv: 2006.14195. [Online]. Available: http://arxiv.org/abs/2006.14195

[14] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996. [Online]. Available: http://ieeexplore.ieee.org/document/508439/

[15] J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 3475–3482. [Online]. Available: http://ieeexplore.ieee.org/document/5980408/

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv:1506.02640 [cs]*, May 2016, arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640