SocNavBench: A Grounded Simulation Testing Framework for Evaluating Social Navigation

ABHIJAT BISWAS, ALLAN WANG, GUSTAVO SILVERA, AARON STEINFELD, and HENNY ADMONI, Carnegie Mellon University

The human-robot interaction community has developed many methods for robots to navigate safely and socially alongside humans. However, experimental procedures to evaluate these works are usually constructed on a per-method basis. Such disparate evaluations make it difficult to compare the performance of such methods across the literature. To bridge this gap, we introduce *SocNavBench*, a simulation framework for evaluating social navigation algorithms. *SocNavBench* comprises a simulator with photo-realistic capabilities and curated social navigation scenarios grounded in real-world pedestrian data. We also provide an implementation of a suite of metrics to quantify the performance of navigation algorithms on these scenarios. Altogether, *SocNavBench* provides a test framework for evaluating disparate social navigation methods in a consistent and interpretable manner. To illustrate its use, we demonstrate testing three existing social navigation methods and a baseline method on *SocNavBench*, showing how the suite of metrics helps infer their performance trade-offs. Our code is open-source, allowing the addition of new scenarios and metrics by the community to help evolve *SocNavBench* to reflect advancements in our understanding of social navigation.

CCS Concepts: • **Human-centered computing** → **Social navigation**; *HCI design and evaluation methods*; • **Computing methodologies** → *Simulation types and techniques;*

Additional Key Words and Phrases: Benchmark, social navigation, pedestrian, human robot interaction

ACM Reference format:
Abhijat Biswas, Allan Wang, Gustavo Silvera, Aaron Steinfeld, and Henny Admoni. 2022. SocNavBench: A Grounded Simulation Testing Framework for Evaluating Social Navigation. *Trans. Hum.-Robot Interact.* 11, 3, Article 26 (July 2022), 24 pages.
https://doi.org/10.1145/3476413

1 INTRODUCTION

Robots must navigate in a safe, predictable, and socially acceptable manner to succeed in spaces designed for and occupied by humans—often referred to as social navigation. Many methods for social navigation have emerged in the **human-robot interaction (HRI)** literature [10, 19, 24, 30, 33, 35, 52, 57, 60], focusing on large dense crowds [19, 52], local interactions with small crowds [35, 57], and specific methods for service scenarios [7, 39, 49]. However, it remains difficult to evaluate these methods against one another in a consistent manner. Each algorithm tends to be

This work was funded under grants from the National Science Foundation (Grant No. NSF IIS-1734361) and the National Institute on Disability, Independent Living, and Rehabilitation Research (Grant No. NIDILRR 90DPGE0003).
Authors' address: A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213; email: abhijat@cmu.edu.

2573-9522/2022/07-ART26 $15.00
https://doi.org/10.1145/3476413

evaluated using scenarios and metrics chosen by the researchers for their specific implementations. There is currently no single method that enables the characterization of the strengths and weaknesses of different social navigation approaches in a consistent and interpretable manner. In this work, we introduce a simulation-based evaluation framework to enable direct comparisons between different social navigation approaches. Our algorithm-agnostic framework uses ecologically valid test scenarios based on real-world pedestrian data and introduces a series of metrics that quantify different aspects of navigation.

Social navigation is a difficult task to benchmark for several reasons. First, what constitutes successful social navigation is subjective and heavily dependent on contextual factors, including location, available space, the mobility of surrounding humans, the robot's function, and local or cultural norms. For example, it is much more acceptable for a robot to cut across pedestrian paths if ferrying emergency medicine to a mall customer experiencing an anaphylactic reaction than if delivering food to a customer in the food court. However, not all scenarios are this clear. The same robot performing the same task in a hospital should now avoid obstructing nurses, doctors, and patients. In our navigation benchmark, we provide a number of automatically computed metrics that quantify system characteristics, such as robot energy expenditure, path smoothness, speed, and safety, among others. Using these metrics, evaluations of social navigation algorithms can prioritize characteristics appropriate for the robot's context.

Second, it is difficult to create consistent and repeatable realistic navigation scenarios with which to fairly compare different algorithms, while also being faithful to real-world pedestrian behavior. If we attempted to replicate scenarios across different robots and algorithms in the real world, then we would have to artificially constrain human participants to follow certain paths or move to fixed targets. However, such artificially contrived scenarios can cause people to behave in unnatural and constrained ways, which would render such an evaluation ineffective at representing real-world, natural social navigation. Instead, our navigation benchmark evaluates social navigation in simulation, allowing for perfectly repeatable experiments. To keep our simulator representative of real-world pedestrians, we crafted 33 representative navigation scenarios with real-world pedestrian trajectories. Additionally, the simulator provides 3D rendering and a depth map for algorithms that use those to provide a more realistic simulation.

Third, like most HRI problems, social navigation is subject to immense variation amongst humans. While many social navigation algorithms are currently evaluated in simulation [9–11, 19, 20, 22, 34, 54, 57, 60], such simulators are usually constructed ad hoc by each individual research group, with pedestrian trajectories generated via models of pedestrian behavior. Evaluating with synthetically constructed pedestrian trajectories means that the evaluation may not represent real-world performance around human pedestrians. In contrast, our simulator uses real-world pedestrian data replayed in realistic navigation scenarios. The data are sourced from several open-source data sets that capture a variety of crowd characteristics and navigation environments. While this does not account for mutual motion where pedestrians react to the robot motions, this does support the default social navigation goal of not affecting the humans' intended motion.

In summary, our work addresses existing gaps in the evaluation of social navigation algorithms with *SocNavBench*, a pre-recorded pedestrian simulation framework. We include a set of curated episodes containing a variety of social navigation scenarios in different environments and featuring different crowd characteristics. Further, we propose a suite of metrics that evaluate social navigation algorithms along complementary axes of performance to illustrate the trade-offs that social navigation algorithms must make. These trade-offs include balancing robot speeds and directness of trajectories with causing minimal pedestrian disruption.

SocNavBench is useful for researchers interested in comparing social navigation algorithms and robot designers interested in selecting the best navigation algorithm for their application. We

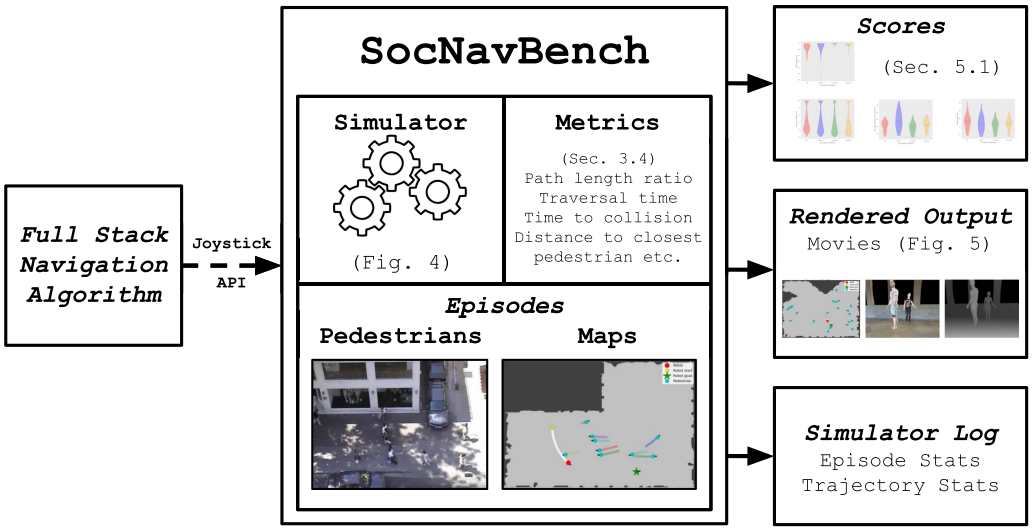


Fig. 1. SocNavBench is a social navigation test bench comprising a photo-realistic renderer, a curated set of navigation scenarios based on real-world pedestrian data, and a suite of metrics to characterize the performance of robot navigation algorithms. SocNavBench can take as input a full or partial stack navigation algorithm. It supports simulated photorealistic RGB-D sensing and model-based control.

provide an API that supports testing any social navigation algorithm on a simulated robot in one of 33 scenarios (Figure 1). SocNavBench automatically outputs metric scores, a detailed simulator log, and videos that show the robot in action. Robotics researchers can use SocNavBench to compare their own algorithms to prior work with a repeatable scenario using consistent metrics, enabling apples-to-apples comparisons. Given several social navigation algorithms and a particular scenario in which to deploy a robot, a system designer can also use SocNavBench as a diagnostic tool to choose which algorithm (or specific parameters for a given algorithm) is appropriate for their particular context. To illustrate how SocNavBench can be employed, we implement three popular social navigation algorithms [10, 19, 57] and a naive, pedestrian-unaware baseline method within the simulator, and we report results comparing them.

In this article, our contributions are:

(1) a photo-realistic simulator with a navigation algorithm agnostic API,
(2) a curated set of episodic social navigation scenarios comprising various environment layouts and pedestrian densities based on real-world pedestrian data for use with the simulator,
(3) a suite of evaluation metrics to measure the performance of social navigation algorithms across these episodes,
(4) a test bench comprising the above three components for evaluating social navigation algorithms, and
(5) a comparison of three existing social navigation strategies and a baseline using the aforementioned episodes and metrics.

2 RELATED WORK

2.1 Social Navigation Methods

The evolution of social navigation algorithms developed by the HRI research community is reflected by the increasing capabilities to model high-level context and human behavior variation.

Early works on robot navigation focused on decoupled models. These models treat pedestrians as independent moving entities or simply dynamic obstacles. Some examples include the dynamic window approach [15], randomized dynamic planning approaches using RRT [28] and velocity obstacles approaches [13]. Later, researchers realized that the biggest challenge in addressing pedestrians lies in the uncertainty of pedestrians' future trajectories. Works such as References [11, 21, 41] attempted to predict future moving patterns of the dynamic entities in the environment.

Researchers later determined that interaction modeling is the key to better human behavior modeling. Models without interaction modeling often run into *the freezing robot* problem [52]. One of the earliest works to address interaction in navigation was the social forces model [19]. This model uses forces to steer agents away from obstacles and other agents and to steer agents toward goals and other attractive entities. More recent attempts to model interactions have been diverse. Vandenberg et al. proposed reciprocal velocity obstacles to account for the reciprocity of planning under velocity obstacles for a pair of pedestrians [57]. More recently, topology concepts have been employed in modeling how pedestrians unanimously reach a common meta-level passing strategy in a game theoretic setting [34]. Trautman et al. attempted to model interaction via a joint density term inside the pedestrians' Gaussian process mixtures models [51]. Other works explicitly modeled certain aspects of interaction and incorporated them in navigation, such as grouping considerations [22, 38, 62], proxemics [6, 55], and personality traits [6].

Growing in popularity, researchers have also attempted to use learning-based models [30, 51] to capture context and interaction. Deep reinforcement learning has been used to develop a collision avoidance policy, augmented with social awareness rules to inject interaction components [9, 10]. The model's reward definition can be adjusted to retrain policies for new context. Inverse reinforcement learning techniques have also been used in an attempt to implicitly capture interaction into a cost function by observing how real pedestrians navigate [23, 26, 40]. In a more well-defined problem space, the pedestrian trajectory prediction problem shares a similar necessity of modeling pedestrian interactions. In this problem domain, the learning-based models contain interaction modeling modules to enhance pedestrian future states predictions, which can then be potentially adopted as future obstacle space in a navigation setting. For example, Social-LSTM [2] and Social-GAN [17] used social pooling layers to summarize pedestrian spatial distribution for both context and interaction modeling. SoPhie [44] additionally used top-down view image patches around pedestrians to better understand the pedestrians' surrounding context. Another interaction modeling technique used graph-based relationships [36, 61] to implicitly learn attention weights for all pairs of pedestrians, with higher weights corresponding to more interactions.

2.2 Evaluating Social Navigation

High variation in context and human behavior is not only a challenge for social navigation algorithms but also for evaluating these algorithms. Furthermore, what set of metrics define a successful social navigation is subjective and difficult to define. Given infinite resources, the ideal test strategy could be running thousands of trials of a robot navigating through crowds in many different real-world locations. Likewise, the ideal metric could be to ask the hundreds of thousands of pedestrians who have interacted with the robot, as well as the people who tasked the robot, to rate their experiences. However, the cost of running such a study is impractical, so researchers have used various testing strategies and metrics to approximate *the ideal test*.

One group of strategies to approximate *the ideal test* is to evaluate social navigation algorithms in a small-scale, real-world setting. One such approach is to run a qualitative demonstration in the wild without objective or subjective metrics [10, 26]. These demonstrations are not easily reproducible and cannot be effectively used to compare different algorithms fairly. Another approach is

to run user studies in a controlled setting. Typically, this is done by asking a few participants to start at specific locations and reach specific goals while the robot navigates nearby [24, 26, 27, 32, 40, 56]. Similarly, some teams ask human participants to teleoperate a robot and compare the algorithm's performance with the humans' [23, 53]. While these controlled, real-world test strategies offer the benefit of realism, human participant studies at scale are very expensive. This also creates limits on crowd size and context variety. It may seem possible to overcome this by conducting user studies in the wild [14, 23, 45, 53], but it is nearly impossible to repeat the same scenario for algorithm comparisons and controlled iterative development.

Another method of approximating *the ideal test* is to use real-world datasets. This is accomplished by replacing one of the pedestrians with a robot and comparing the performance differences by treating the replaced pedestrian's trajectory as the ground truth trajectory [30, 52]. A similar method is to put a virtual robot in the scene and ask it to navigate to designated waypoints [5, 6, 8]. The benefit of the virtual-robot-in-dataset testing strategy is that it is a good approximation to real-world scenarios as it uses real-world pedestrian trajectories and environments. Because datasets typically contain hundreds of trajectories, this testing method offers large quantities of test cases. By combining scenes from different datasets, we can further gain a decent variety of context. For these reasons, we developed our approach around real-world pedestrian trajectories in our simulator. An unmet need in prior implementations of this method is that it is impossible for a simulated robot to obtain realistic sensor inputs. Thus, the social navigation algorithms being tested need to rely on the assumption of perfect perception.

The final popular approach to approximate *the ideal test* is to use simulators. Simulators offer the benefit of generating large numbers of test cases, but a critical concern is what models should be used to simulate pedestrian behavior. One approach some researchers used is called "self-play" [10, 19, 22, 34, 54, 57]. Each agent in the simulated environment uses the same social navigation algorithm to "play" against each other. Another common approach is called "against-all" [9, 11, 20, 60]. In this case, the social navigation algorithm is implemented on one agent. Then, that agent is tested against all other agents governed by a different model. The fundamental problem with these two pedestrian modeling approaches is that instead of testing against actual human behavior, the social navigation models are treated as ground truth.

Instead of simulating pedestrian behavior, we directly source real-world pedestrian trajectories from natural behavior datasets. To overcome the inability to obtain sensor inputs for these datasets, we simulate environments by applying real-world textures on pedestrians and environments, which we then use to generate photorealistic sensor inputs. Additionally, since ours is a simulated setting, we can still generate large quantities of test cases. By combining the benefits of the dataset and simulator-based testing strategies, every element in our approach is grounded in reality. Therefore, we believe our testing strategy is a better approximation of *the ideal test*.

In terms of the human rating metric used in *the ideal test*, it is possible to ask human observers to rate how simulated robots behave in our simulators, as is the case in References [31, 45]. However, our aim is to make our simulator a diagnostic tool that automatically outputs scores, so human ratings are not logistically reasonable. Therefore, we decided to adopt approximations to the human rating metric, as seen in most dataset- and simulator-based testing methods. Common approximations to the pedestrian experience ratings (i.e., social ratings) include collisions or success rate [6, 9, 10, 40], closeness to pedestrians [6, 23, 26, 51, 52], path predictability [32, 40], among others. Common approximations to evaluation by robot task-givers (i.e., task efficiency) include time to the reach goal [9, 10, 23, 40], speed [26, 51], path length [30, 40, 52], and other navigation performance-based metrics. Our large selection of metrics offers users of *SocNavBench* the ability to prioritize metrics they deem most suitable for their specific needs. We demonstrate the use and

interpretation of these metrics by testing on three different existing algorithms and providing a comparison of these metrics.

3 SOCNAVBENCH DESIGN

3.1 Overview

Our proposed tool, *SocNavBench*, is a simulator-based benchmark with prerecorded real-world pedestrian data replayed (Figure 2). The simulator can run in two modes, *Schematic* and *Full-render*. The *Schematic* mode focuses on trajectory-based navigation with the problem of perception abstracted away. In the *Full-render* mode, we provide an RGB-D image with customizable camera position and intrinsic parameters to simulate real-world, on-board sensing.

Algorithms evaluated by the tool see several episodes comprising different prerecorded human trajectories and environments. For each such episode, the algorithm under evaluation receives one of several environment maps (Figure 3) and a start and goal configuration within that environment that it must traverse in an efficient and socially acceptable manner. We measure robot navigation efficiency by measuring the robot's smoothness and directness of path as well as its energy expenditure. We measure social acceptability from potential disruptions to recorded human paths, with the most socially acceptable paths not only avoiding collisions with pedestrians but also maintaining a comfortable distance from pedestrians whenever possible.

An assumption we use is that the pre-recorded pedestrian trajectories that were recorded in the absence of a robot are the preferred trajectories of these pedestrians. At minimum, a social robot present in the original scenario should not cause pedestrians to deviate from this path. Therefore, we should not see collisions with the pre-recorded pedestrians. Likewise, any robot action that leads to motions very close to replayed human trajectories would probably have induced pedestrian deviations from their preferred path. This is also sub-optimally social. For a discussion of the validity and limitations of this assumption see Section 5.3

Our metrics suite is designed with these considerations in mind. In particular, we measure pedestrian disruption in two ways: a time-to-collision measure and a closest-pedestrian distance measure. The distribution of these metrics helps quantify how close the robot comes to cutting off pedestrians during its navigation. Additionally, we provide metrics that quantify the directness (versus circuitousness), smoothness (versus jerkiness), and energy efficiency of the robot's navigation. Together, these represent a characterization of both the appropriateness and efficiency of the robot's navigation.

3.2 Pre-recorded pedestrian Data and Episode Curation

The pre-recorded pedestrian data used in SocNavBench comes from the UCY [29] and ETH [42] pedestrian trajectory datasets, which are widely accepted and used by the community [2, 6, 8, 17, 52] and were also included in recent pedestrian trajectory prediction benchmarks [4, 25]. These data include varied crowd densities and walking speeds, as well as interesting pedestrian behaviors, such as grouping, following, passing, pacing, and waiting. Pedestrian trajectory data is replayed at the simulator tick rate (default 25 fps) with a 1:1 ratio to recorded time, i.e., replayed pedestrian trajectories have the same velocities as the recorded ones. If the simulated robot collides with a pedestrian, then SocNavBench registers that collision and the pedestrian's replayed motion is unchanged. Future support for reactive pedestrians is planned, which has its own set of assumptions and tradeoffs (see Section 5.3 for a discussion)

We manually curated a set of episodes that include challenging scenarios from the aforementioned pre-recorded datasets. These episodes consist of a diversity of crowd sizes, speeds, and densities, as well as directions of motion with respect to the robot's traversal task, while excluding

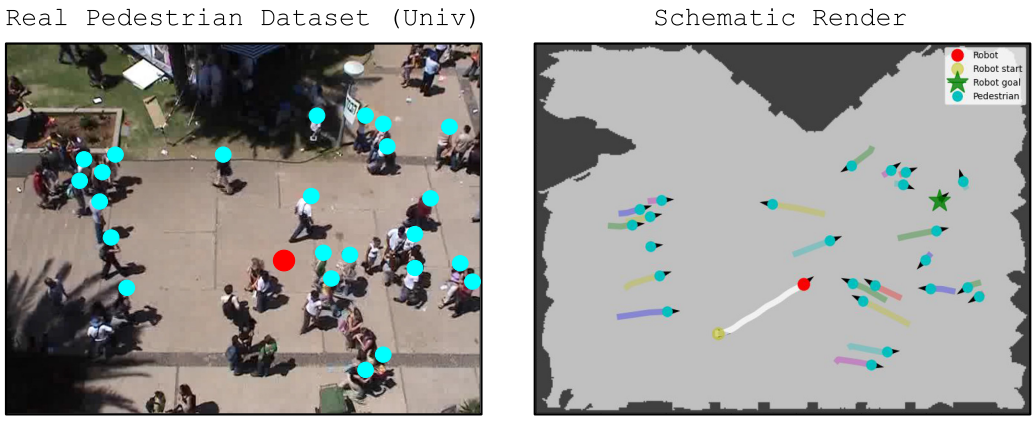


Fig. 2. An example of the transfer of real data to the simulator. Pedestrian trajectories are replayed at the same speed and environment structures are faithfully reflected for navigation.

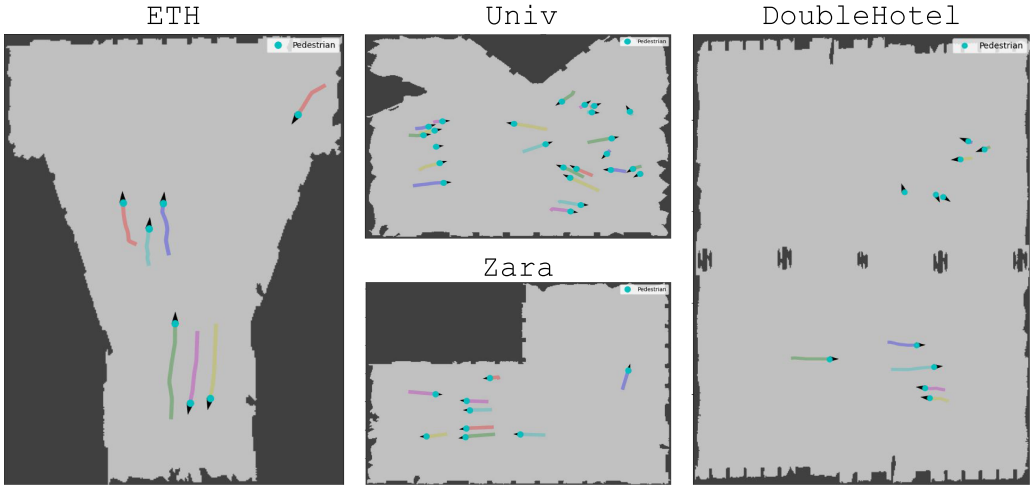


Fig. 3. Example maps available in *SocNavBench*. Maps have varied environment structures as well as varied crowd sizes and densities.

trivial scenarios with very low crowd densities.[1] We also include multiple environments (Figure 3) with different types of configurations, hence providing environmental obstacle diversity as well. In total, we have 33 episodes, each lasting for at most 60 s. Our curated episodes contain an average of 44 pedestrians (standard deviation = 13), with the minimum being 24 and the maximum being 72. Additionally, to allow users access to large numbers of episodes, we have added support for random episode sampling, which samples a queried number of random robot start and goal pairs in a randomly selected map along with one of the corresponding handpicked sections of recorded pedestrians from the aforementioned curated episodes. These random start and goal pairs are picked in a way that ensures they are reachable in 25 s when using the maximum permitted robot velocity. While forgoing the benefits of hand-picked starts and goals (such as scenarios that require the robot to cross a high density crowd flow), this will allow users to use the simulator for learning purposes where large sample sizes are especially important.

[1]Here, crowd density refers to the number of pre-recorded pedestrians per unit area.

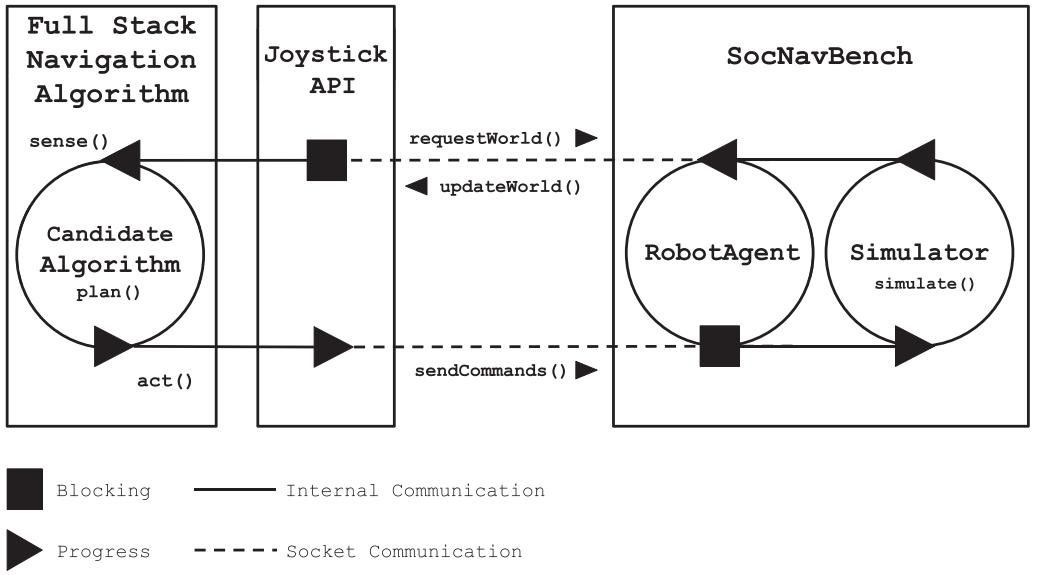


Fig. 4. An illustration of simulator flow control in *synchronous* mode. At each iteration, the candidate algorithms uses the `Joystick API` to send a `sense()` signal, plans with the newly received world data, and then sends back an `act()` signal.

We adapted meshes from the **Stanford Large-scale 3D Indoor Spaces Dataset (SD3DIS)** [3] to provide the physical environments in which these episodes to take place, modifying them to reflect the spaces in which the original data was collected. We chose a variety of segments from the pedestrian datasets to include a distribution of crowd sizes, crowd densities, pedestrian speeds, and pedestrian trajectories. Robot start and goal positions within these scenarios were sampled such that they were semantically meaningful whenever possible. For example, we chose the start position of the robot to be the exit from a building and the goal to be the end of a sidewalk in one scenario.

For perspective rendering, we based our system upon The HumANav Dataset [50], which uses Google's Swiftshader renderer for photorealistic RGB and depth visuals. Importantly, the rendering engine for this project is separate from the specific meshes used, so any custom meshes could be used. The building mesh scans in the environment originated from the SD3DIS [3], but they were modified to replicate the environments in which the pedestrian data was recorded (Figure 2). Human meshes were drawn from the SURREAL Dataset [59], which includes photorealistic meshes for various human body configurations, genders, and lighting conditions. In particular, 6,000 human models form SURREAL are used that are parameterized by body configuration, pose, and velocity during walking. For replaying human poses accurately, we can calculate the body orientation and walking velocity of the recorded pedestrians from their recorded trajectory. These are then plugged into the aforementioned parameterized human models to be rendered such that accurately represent the poses that would have occurred naturally when walking at that speed and orientation.

3.3 SocNavBench Simulator Mechanics

SocNavBench provides a socket-based interface for candidate algorithms (clients) to communicate with the simulator (server) and the simulated RobotAgent (Figure 4). At each evaluation start, the

client is sent metadata about the episodes on which it is going to be evaluated. Then, episodes are served to the client sequentially, including environment information, a time budget for each episode, and start and goal locations. Each episode ends either when the robot reaches the goal (*Completion*), collides with an environmental obstacle (*Environment Collision*), or runs out of time for that episode (*Timeout*). Each *Completion* may be a successful episode (*Success*) if there were no collisions. If the robot collided with a pedestrian during its completion of the episode, however, then we count the episode as a *Pedestrian Collision* and the episode cannot count as an overall success.

The flow control in *SocNavBench* can be understood based on a classical sense-plan-act cycle design. When a candidate algorithm performs a sense action, it requests an updated WorldState from the simulator. This state is a position for all pedestrians in the scene in *Schematic* mode and an RGB-D image in the *Full-render* mode. It can then plan on the received state and send either velocity or position commands to update the simulated RobotAgent. Importantly, this abstraction allows many different types of algorithms to be compatible with *SocNavBench*, including end-to-end learning-based methods. This also allows individual parts of a modular navigation algorithm to be tested in an ablation study. For example, a planning algorithm could be tested with an RGB-D-based perception method in *Full-render* mode and also with perfect perception using the *Schematic* mode, hence isolating the navigation performance differences attributable to the perception method.

3.3.1 Synchronicity. Our simulator can run in two modes with respect to robot client and simulator server synchronicity: *Asynchronous* and *Synchronous*. In asynchronous mode, the simulator constantly updates at a fixed rate (simulator time is moving forward) while listening to the robot client for inputs in the background. In this mode, more complex algorithms that run slower will loop through their sense-plan-act cycles in a manner close to the real world, where taking too long to plan may result in the sensed data being stale by the time the action is performed.

In contrast, in the synchronous mode, the simulator listens for a robot command at each simulator step. This is the default mode, which does not penalize algorithms for "thinking time." The synchronous option is preferred, since thinking time may be dictated in part by the compute hardware used, leading to irreproducible results. However, a user may want to run the simulator in asynchronous mode for a more realistic evaluation. This is especially relevant if the user (1) can control for the computational hardware across different algorithms and (2) is testing on the hardware to be used during deployment.

3.3.2 Rendering Modes. *SocNavBench* supports two rendering modes: *Schematic* and *Full-render* (Figure 5). For visualization purposes, our approach plots a schematic top-view of the entire scene, but it also includes the option of a full render from the perspective of a camera specified by an arbitrary 6-dof pose. The default pose corresponds to a camera attached atop a Pioneer robot [1]. The *Full-render* mode additionally contains building scans for the environment, and human meshes for the pedestrians. The default modes of operation are *Synchronous* and *Schematic*. Henceforth, when not specified, these are the modes used.

3.3.3 Simulated Robot Specifications. The simulated robot is able to run in velocity control mode and in direct position control mode. In velocity control mode, the robot is subject to kinematic constraints according to a three-dimensional unicycle model with dynamics:

$$\dot{x} = v\cos\phi, \quad \dot{y} = v\sin\phi, \quad \dot{\phi} = \omega, \tag{1}$$

where the position of the robot is (x, y) and the heading is ϕ.

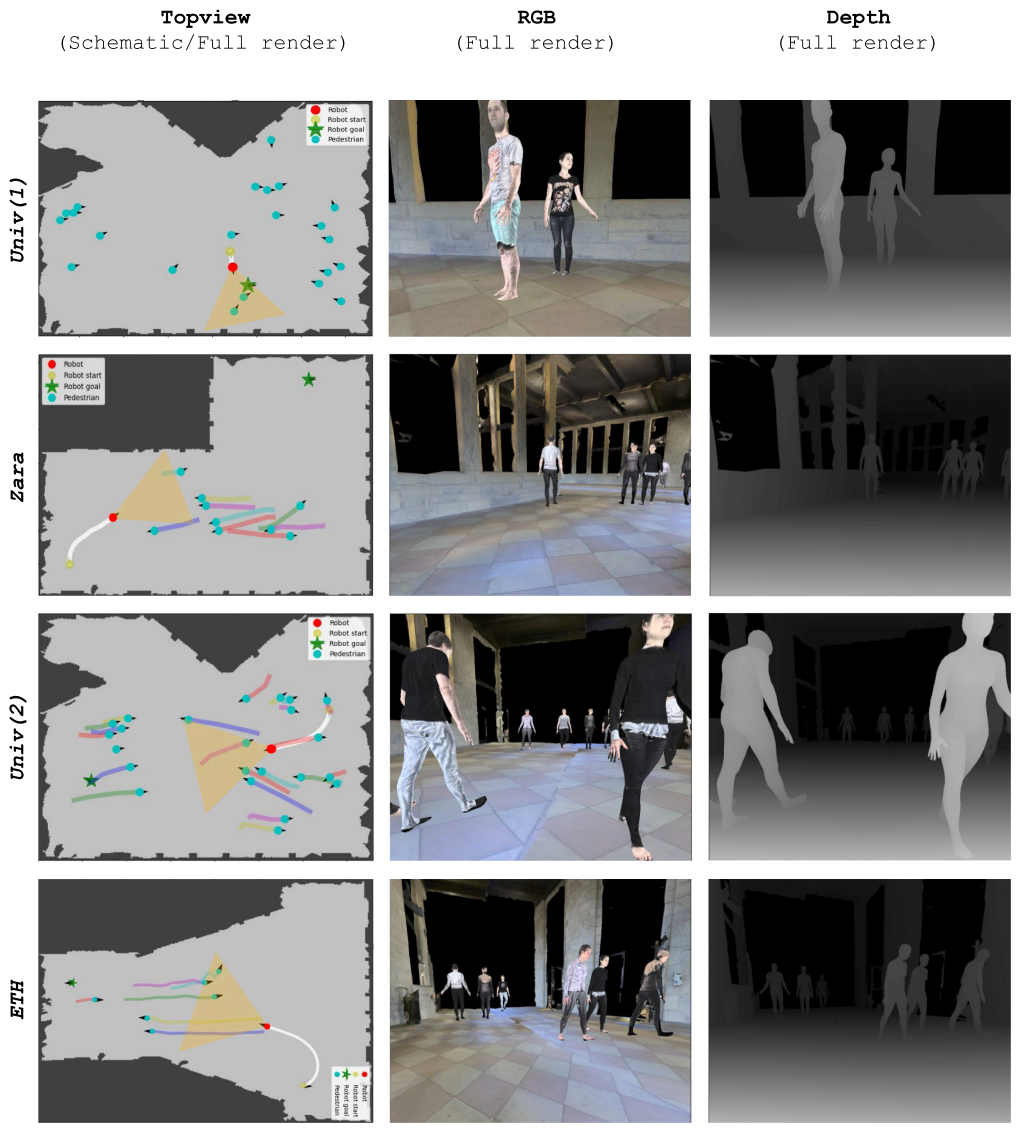


Fig. 5. The different types of sensing data available across both rendering modes, *Schematic* and *Full-render*. Camera direction is represented by the colored cone.

However, most social navigation algorithms [10, 19, 35], including ones that provide guarantees of collision-free navigation [57], do not include a dynamics model in the derivation of their guarantees. To ensure compatibility, we also provide a velocity-limited holonomic control mode, which is akin to position control with upper bounded instantaneous velocity.

We based the default simulated robot design on the Pioneer 3-DX robot [1] using the base system dimensions and maximum velocity according to off-the-shelf specifications. In the full 3D rendering mode, the default camera height is typical for an on-board RGB-D camera on this base. However, these options are customizable to represent any robot base and sensor position on that base, since any arbitrary 6-dof camera pose camera pose can be specified.

Table 1. The Metrics Suite Implemented in *SocNavBench*

Metric	Units	Definition
		Meta statistics
Overall success rate	—	the fraction of episodes successfully completed (robot reached goal position and did not collide with any pedestrians)
Failure cases	—	the distribution of failure cases over *Timeout*, *Pedestrian Collision*, and, *Environment Collision*
Total pedestrian collisions	—	the total number of pedestrian collisions across all episodes considered.
Average planning wait time	*seconds*	the average time spent by the simulator waiting for the planning algorithm per simulation step
		Path quality
Path length	*meters*	the total distance traversed by the robot in the episode
Path length ratio	—	the ratio of straight line distance between the end and goal to the robot's path length for any episode
Goal traversal ratio	—	the ratio of the robot's distance to goal at the episode end and distance to goal at the episode start
Path irregularity	*radians*	averaged over each point of the robot trajectory, this is the angle between the robot heading and the vector pointing to goal
Path traversal time	*seconds*	the total simulator time taken to traverse the robot's path
		Motion quality
Average speed	m/s	average speed of the robot over its entire trajectory
Average energy expenditure	*Joule*	assuming unit mass, the robot energy expenditure is calculated as the integral of the squared robot velocity over the entire robot trajectory
Average acceleration	m/s^2	average acceleration of the robot over its entire trajectory
Average jerk	m/s^3	average jerk (time derivative of acceleration) of the robot over its entire trajectory
		Pedestrian related
Closest-pedestrian distance	*meters*	at each robot trajectory segment, this is the distance to the closest-pedestrian
Time-to-collision	*seconds*	at each robot trajectory segment, this is the minimum time-to-collision for any pedestrian in the environment

See Section 3.4 for more details and considerations.

3.4 Evaluation Criteria and Metrics

We define several metrics (Table 1) that may be important considerations for a socially navigating robot and can help characterize the trade-off between robot appropriateness and efficiency. The metrics fall into four general evaluation categories: path quality, motion quality, pedestrian disruption, and meta statistics.

Path quality. Path quality metrics quantify the quality and efficiency of the path generated by the social navigation algorithm. These metrics are focused on the robot's path in a global sense.

(1) Path length (*meters*): the total distance traversed by the robot in the episode. Usually, lower is better and indicates that a robot takes more direct paths.
(2) Path length ratio: the ratio of straight line distance between the start and goal to the robot's path length for any episode. Usually, higher is better and indicates that a robot takes more direct paths. Trade-offs between navigation optimality and pedestrian safety may affect this metric.
(3) Path irregularity (*radians*): averaged over each point of the robot trajectory, this is the absolute angle between the robot heading and the vector pointing to goal [18]. For a straight path from start to goal, path irregularity is zero. Usually, lower is better and indicates that a robot takes more direct paths, though as with path length ratio, this may vary based on other factors like pedestrian safety.
(4) Goal traversal ratio: calculated for *Incomplete* episodes, this is the ratio of the robot's distance to goal at the episode end to start. This is useful to quantify the partial success of algorithms when not completing the episode due to timeout or collision with the environment. Lower is better and indicates the robot gets closer to the goal during *Incomplete* episodes.
(5) Path traversal time (*seconds*): total simulator time taken to traverse the robot's path. Lower is usually better but higher may be acceptable if the robot yields to pedestrians more frequently.

Motion quality. The quality of the robot's motion is characterized by the smoothness and efficiency of the robot's movement. These metrics quantify the robot's energy expenditure, acceleration, and jerk.

(1) Average speed ($meters/second$): average speed of the robot over its entire trajectory.
(2) Average energy expenditure (*Joules*): robot energy is calculated as the integral of the squared robot velocity over the entire robot trajectory, assuming unit mass. Total robot energy expended is related to total path and velocity, but is slightly different from each. For example, two different algorithms may yield to pedestrians either by moving out of the way or by stopping before moving out of the way is necessary. The second algorithm may take the same overall time and have similar average velocity as the first but will expend lower energy in total. All other things equal, lower energy expenditure is better.
(3) Average acceleration ($meters/second^2$): average acceleration of the robot over its entire trajectory.
(4) Average jerk ($meters/second^3$): average jerk (time derivative of acceleration) of the robot over its entire trajectory. Lower indicates that the robot takes smoother paths, which are more predictable and legible for surrounding pedestrians while also consuming less energy.

Pedestrian-related. These metrics capture the robot's movements with respect to the surrounding pedestrians. They are mainly focused on the safety of the robot's navigation.

(1) Closest-pedestrian distance (*meters*): at each robot trajectory segment, this is the distance to the closest pedestrian. Pedestrian distances more than 10 m are saturated to 10 m.
(2) Time-to-collision (*seconds*): at each robot trajectory segment, this is the minimum time-to-collision to any pedestrian in the environment. The time-to-collision for any robot-pedestrian pair is computed by linearly extrapolating the robot and pedestrian trajectories using their instantaneous velocity. This is equivalent to the time it would take the robot-pedestrian pair to collide if, at the end of this trajectory segment, they continued moving at their current speed in their current heading. This is calculated for every robot-pedestrian pair in the environment and the minimum TTC is selected. Times larger than 10 s are saturated to 10 s.

The two metrics above are similar but not the same and are better interpreted together. For example, a robot may follow a crowd walking in the same direction very closely, which would have a low *Closest-pedestrian distance*. However, because their motions are in the same direction, the *Time-to-collision* will also be high, indicating overall a lower pedestrian disruption. Between two algorithms with similar *Time-to-collision*, users may prefer the higher *Closest-pedestrian distance* in scenarios where giving pedestrians a wider berth is preferred. These two metrics are range limited, since instantaneous values of Times-to-collision greater than 10 s and Closest-pedestrian distances greater than 10 m are unlikely to impact the robot's operation in any way. Leaving these outliers in would affect their averages, which may reflect an overall safer robot navigation than is actually the case.

Meta statistics. Additionally, the tool generates a selection of overall success meta-statistics for the entire collection of episodes:

(1) Overall success rate: the fraction of episodes successfully completed (robot reached goal position and never collided with any pedestrian). Episodes may be completed but not successful if they have pedestrian collisions but no timeouts or environment collisions.
(2) Total pedestrian collisions: the total number of pedestrian collisions across all episodes considered. Lower is better.
(3) Failure cases: the distribution of failure cases over *Timeout*, *Pedestrian Collision*, and *Environment Collision*. This is reported as a tuple (T/PC/EC). Lower numbers for each category are better.
(4) Average planning wait time (seconds): the average time the simulator waited for commands from the planner. If the algorithms are tested on identical hardware, then this provides a measure of the complexity of the algorithm and, hence, its feasibility for real-time, on-board robot operation. Lower is better.

In the above descriptions, we describe the interpretation of metrics individually with all other metrics being equal. However, this is rarely the case during social navigation where algorithms are constantly negotiating a safety versus efficiency trade-off. Hence, these metrics must be taken in context with each other, which is demonstrated in Section 5.1

4 EXPERIMENTS

4.1 Choice of Candidate Algorithms

We evaluate three existing social navigation algorithms and a simple pedestrian-unaware baseline on *SocNavBench*.

We chose Social Forces [19], ORCA [57], and SA-CADRL [10] as the three candidate social navigation algorithms, because they represent the evolution of the field over three decades. The Social Forces model [19] is an early influential work on modeling pedestrian navigation that motivates many subsequent contributions in the robot navigation domain [37, 43, 63]. ORCA [57] is a popular baseline method used by many recent social navigation algorithms [9, 10, 32] for performance comparisons. CADRL [10] is a recent, popular, and relatively state-of-the-art model using a deep reinforcement learning-based navigation method.

Social-forces-based pedestrian dynamics (henceforth, Social Forces) is a formulation to explain pedestrian navigation behavior as being under the influence of "social forces," including an attractive force to their goal, a repulsive force from other pedestrians and obstacles, and other attractive forces to interesting objects or other pedestrians in the environment. We use an open-source implementation of this method from a popular pedestrian crowd simulator PedSim [16]. This method does not follow an angular velocity constraint.

ORCA: Optimal Reciprocal Collision Avoidance [57] (henceforth, ORCA) is an efficient navigation planner that provides theoretical non-collision guarantees assuming all agents in the environment use the same planning policy. The core idea behind reciprocal velocity obstacles used by ORCA is reciprocal reaction from both the pedestrian and the robot. Because in our simulator the pedestrians are not reacting to the robot, we changed the collaboration coefficients of the pedestrians to be 0 and that of the robot to be 1. However, ORCA, like most existing social navigation frameworks, does not account for system kinematics and dynamics in its formulation. Because its performance guarantees are void when a post-hoc kinematic constraint is applied, we run ORCA in an unbounded angular velocity mode without kinematic constraints on linear velocities.

SA-CADRL: socially aware collision avoidance with deep reinforcement learning (henceforth, CADRL) [10] is a deep reinforcement learning-based social navigation method that formulates social navigation as a cooperative multi-agent collision avoidance problem with well-crafted rules to inject social awareness. We are using the four-agent version of the network. Because the number of agents to be fed into the network is fixed, we only take observations from the three nearest pedestrians to the robot. We used the pre-trained model provided by the authors, as described in the article. Similar to ORCA, CADRL does not consider kinematic constraints.

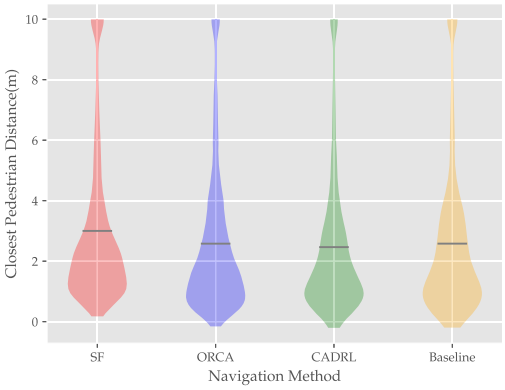
Additionally, we implemented a meta-level planner for ORCA and CADRL, because CADRL does not possess obstacle avoidance capabilities and the authors of CADRL and ORCA [58] endorsed the inclusion of a meta-level planner. The meta-level planner is responsible for identifying a near-future waypoint for the robot to reach, directing the robot away from obstacles in the process. The planner we use is a sampling planner (from Reference [50]) that samples trajectories from a connectivity graph and evaluates them via heuristic cost functions and returns the minimum cost trajectory within a budget. Our meta planner has heuristic-based obstacle avoidance and goal-seeking capabilities. In operation, the meta planner plans a sub-goal within a 6-second horizon each time a new checkpoint is requested. This corresponds to around 4–7 m. The robot is considered to have reached the checkpoint if it is within 1 m of the checkpoint, and then a new checkpoint is requested.

Pedestrian-unaware Baseline: We use the meta-planner described above as a baseline navigation algorithm, which does not take into account pedestrians around it and show its performance on the same set of navigation scenarios as the other three candidate algorithms. This planner can take into account kinematic constraints, so we apply a 3D (position and heading) unicycle model (Equation (1), Section 3.3.3). There is no kinematic constraint application on this planner when used as a meta-planner.

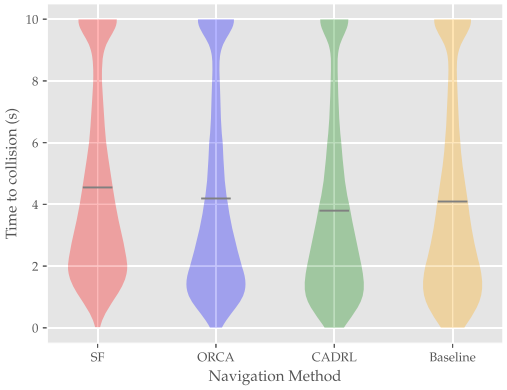
4.2 Experimental Protocol

In our experiments, we used the default simulation settings, i.e., *Synchronous* and *Schematic* modes. In evaluation of algorithms that assume no kinematic constraints, we turned the kinematic constraints off, leaving only the maximum linear speed constraint for the robot.

In total, we tested on the curated set of 33 episodes described in Section 3.2. For the sake of easily comparing between different algorithms, we only consider those episodes in our analysis in which candidate algorithms complete the curated episodes. This is because if some algorithms terminate early due to pedestrian collisions, some metrics (such as path length or total energy expended) are not comparable. Hence, we did not terminate episodes for a pedestrian collision and instead counted all pedestrian collisions throughout the episode. Twenty-nine of 33 episodes were completed by all four candidate algorithms, and all metrics were computed on those 29 completed episodes unless explicitly mentioned.



(a) Closest-pedestrian distance



(b) Time to collision

Fig. 6. Pedestrian-related statistics for the candidate algorithms. Gray lines indicate the distribution mean. CPD is saturated beyond 10 m and TTC beyond 10 s, which results in bimodality toward higher values CPD and TTC instead of a (very) long tail due to outliers.

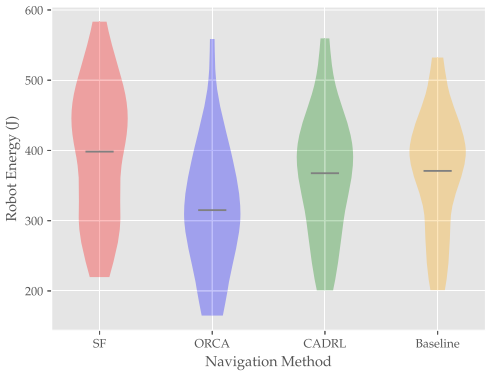


Fig. 7. Average robot motion energy expenditure per episode.

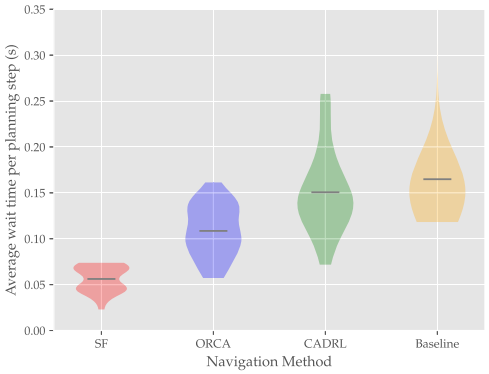


Fig. 8. Average wall-clock wait time per planning step.

5 RESULTS AND DISCUSSION

We ran the aforementioned social navigation methods on the curated set of episodes described in Section 3.2 and discuss the results below. Our focus is on how each metric reflects the overall navigation style of the algorithm with a particular focus on the preferences inherent in the algorithms with respect to the navigation efficiency versus pedestrian disruption trade-off. Figures 6–11 are violin plots, which represent the shape of the distribution of each metric, with the mean of each distribution marked with a horizontal gray tick on the violin. This helps us see outliers and modalities in addition to the information provided by error bars.

5.1 Experimental Results

5.1.1 Meta Statistics (Table 2). From Table 2, the Social Forces model is the most successful social navigation method with the highest success rate 32/33 and no pedestrian collisions. Hence, all completed episodes were also successful. It also has the fastest planning time, largely because it does not require a meta-planner.

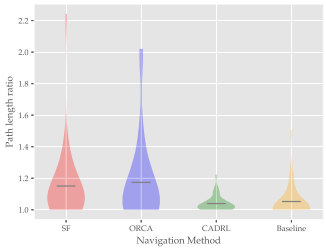


Fig. 9. Average path length ratio per episode.

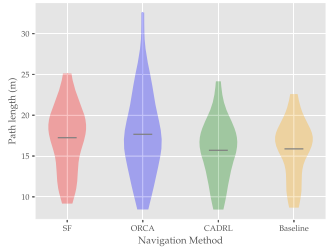


Fig. 10. Average path length per episode.

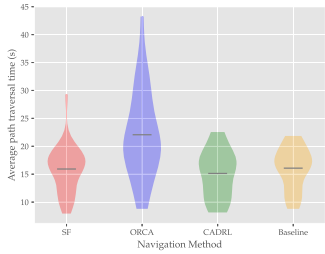


Fig. 11. Average path traversal in simulator time per episode.

Table 2. Average Meta Statistics for Candidate Algorithms Tested on *SocNavBench*

Candidate Algorithm	**Overall success rate**	**Failure cases (T/PC/EC)**	**Total pedestrian collisions**$^{+}$	**Planning wall time per episode (s)**$^{+}$
Social Forces [19]	**32/33**	**(1/0/0)**	**1**	**18.23 ± 7.41**
ORCA [57]	24/33	(1/8/0)	15	48.84 ± 23.06
SA-CADRL [10]	18/33	(0/14/1)	40	46.78 ± 21.38
Baseline (S. 4.1)	9/33	(1/23/0)	64	51.12 ± 16.21

The metrics marked with + are evaluated only on episodes completed by all algorithms.

ORCA is the second most successful model with 24 successful cases of the total 33 and 15 pedestrian collisions across eight episodes. It completed 32/33 episodes. The planning time per step for ORCA is close to the baseline (Table 2 and Figure 8), indicating that the computation bottleneck is the meta-planner.

SA-CADRL, despite being relatively state-of-the-art, frequently runs into pedestrians with 40 collisions across 14 episodes. Its overall completion rate is, however, the same as ORCA. The testing scenarios in the curated dataset are challenging, with sometimes more than three pedestrians moving relatively close to the robot. It is very likely that the robot frequently runs into situations unseen during training and causes collisions. Similar to ORCA, the planning time is close to the baseline, indicating the same computation bottleneck.

Unsurprisingly, the pedestrian-unaware baseline performs the worst with respect to pedestrian safety. While it did complete all but one episode by reaching the goal, it collided with a total of 60 pedestrians across 23 episodes.

Additionally, all candidate algorithms had one *Incomplete* episode each, which were all distinct episodes. This illustrates the diversity and difficulty of our curated set of episodes, which test different aspects of social navigation.

Meta-statistics reflect an overall reliability and computation time for each model. From Table 2, the Social Forces model is computationally fastest and most reliable model, because it passes the most test scenarios with the fewest pedestrian collisions.

5.1.2 Path Quality Metrics (Table 3). The Social Forces model takes very circuitous paths to goal, as reflected in long path lengths (Figure 10) and high path length ratios (Figure 9). However, judging by its high success rate, we can infer that it is being conservative about avoiding pedestrians. This conservative behavior is further demonstrated by high path irregularity, as it tends to take large

Table 3. Average Path Quality Metric Scores for Candidate Algorithms Tested on *SocNavBench*

Candidate Algorithm	Path length (m)	Path length ratio	Goal traversal ratio$^-$	Path irregularity (radians)	Path traversal time (s)
Social Forces [19]	17.25 ± 4.05	1.15 ± 0.23	0.52	1.66 ± 0.95	15.93 ± 4.17
ORCA [57]	17.66 ± 5.22	1.17 ± 0.26	0.21	**1.56 ± 1.01**	22.06 ± 7.78
SA-CADRL [10]	**15.70 ± 3.72**	**1.04 ± 0.05**	0.51	1.68 ± 1.04	**15.14 ± 4.21**
Baseline (S. 4.1)	15.88 ± 3.57	1.05 ± 0.11	**0.09**	1.65 ± 1.02	16.08 ± 3.73

The metrics marked with − are evaluated only on incomplete episodes for each algorithm (T/EC).

curved paths. Despite having long path lengths, the Social Forces model has low path traversal time, meaning it is consistently guiding the robot to move at full speed.

ORCA has the longest path lengths and the second highest success rate, which would seem to imply that ORCA also makes a significant effort to avoid pedestrians by taking roundabout paths. However, given that ORCA also has the lowest path irregularity, we can infer that it is actually trying to take relatively direct paths to goal and using local deviations while trying to to avoid pedestrians. ORCA's longest path traversal times also imply that ORCA is taking a greedy approach of heading toward the goal and using deviations while avoiding pedestrians along the way. A qualitative analysis of the trajectories, paired with the metrics data, indicates that ORCA tends to react late to pedestrians during avoidance maneuvers and then buy time for an avoidance maneuver by starting to move in the opposite direction of the pedestrian (Figure 12). This greedy approach often runs into problems when it is too late to react.

CADRL has the least path length and path traversal time, indicating aggressive goal-seeking behavior.

The baseline method has low path lengths overall, just behind CADRL. This is explained by the fact that the baseline method is subject to kinematic constraints via a unicycle model, which limits its liner and angular acceleration capabilities. It has a larger turning radius than the other candidate algorithms, which is indicated by its high path irregularity.

The path quality metrics mostly reflect a model's efficiency while path irregularity and path length ratio act as sanity checks for presence of large detours or frequent zigzagging patterns. We can see that SA-CADRL is the fastest and the most efficient model overall. However, its aggressive evasive maneuvers would mean a high path irregularity. ORCA produces paths that are direct with less aggressive evasive maneuvers at the cost of efficiency. The navigation driven by Social Forces tended to take large detours.

5.1.3 Motion Quality Metrics (Table 4). The inferences we made about the algorithms from the previous sections are further confirmed by the results in Table 4.

Social Forces has the fastest average speed. This confirms that despite having long paths, it is operating at almost maximum speeds, consistently leading to low path traversal times (Table 3). The smoothness of motion is further illustrated by low robot acceleration and jerk. Due to its behavior being conservative and its relatively longer paths traversed with high speeds, it also happens to consume the highest amount of energy.

ORCA takes relatively direct approaches to the goal, as shown by consuming the least amount of energy here. Low acceleration and jerk indicate that its motion is relatively smooth as well, albeit at not as smooth as the Social Forces model. This implies that ORCA's reactions to pedestrians are not very sudden or unpredictable.

Table 4. Average Motion Quality Metric Scores for Candidate Algorithms Tested on *SocNavBench*

Candidate Algorithm	Average speed (m/s) (max = 1.2m/s)	Average energy expenditure (J)	Average acceleration (m/s^2)	Average jerk (m/s^3)
Social Forces [19]	1.09 ± 0.27	398.33 ± 95.45	0.39 ± 1.43	**2.70 ± 28.23**
ORCA [57]	0.80 ± 0.27	**315.03 ± 89.89**	0.31 ± 1.26	6.28 ± 28.85
SA-CADRL [10]	1.04 ± 0.36	367.58 ± 87.24	0.93 ± 2.91	31.68 ± 87.71
Baseline (S. 4.1)	0.99 ± 0.42	370.88 ± 84.55	4.81 ± 9.07	180.86 ± 303.88

CADRL also is very fast, which is again explained by its aggressiveness in goal seeking. This aggressiveness is further shown by high acceleration and jerk, which indicate sudden movements to dodge pedestrians. Despite being aggressive in proceeding toward the goal, it expends plenty of energy, most of which is spent on accelerating and decelerating the robot.

The motion quality metrics reflect how predictable a model's produced robot actions are. We can see that SA-CADRL has large acceleration and jerk. Therefore, coupled with its highly direct paths, it is recommended over the other models in situations where the robot has fewer energy constraints such as small operating range. The Social Forces model produces the smoothest trajectories, while ORCA closely follows.

5.1.4 Pedestrian Disruption Metrics (Figure 6). We can see again from Figure 6 that the Social Forces model is the most conservative and safest model, because it has the furthest closest-pedestrian distances with no collisions and longest times-to-collision. ORCA tends to avoid pedestrians locally and will run into clusters of pedestrians hoping to find gaps, so it has a smaller closest distance and short times-to-collision. CADRL is very aggressive and only dodges pedestrians at the last second. This is shown by the shortest closest distance and short times-to-collision. This observation of CADRL's aggressiveness was also made in prior work [9]. For CADRL and ORCA, the closest-pedestrian distance has some negative values, because there are episodes with collisions between the simulated robot and pedestrians. Pedestrian disruption metrics indicate the navigational compliance of a robot by measuring how much it avoids or yields to pedestrians. We can infer that Social Forces is the most compliant model, because of its high minimum distance to pedestrians and large times to collision. Conversely, SA-CADRL is the most aggressive model, with ORCA in between.

5.2 Summary of Candidate Algorithm Performance

Overall, the Social Forces model produces robot motion that errs on the side of caution. In our testing, it produced long paths that take care to avoid pedestrians but also traversed them quickly. It was also the computationally quickest planner amongst all tested algorithms, although this is partly because it did not require a meta-level planner for obstacle avoidance. It also suffers from an issue where a large cluster of pedestrians away from the goal can cause it to overshoot the goal due a large repulsive force. This can be seen in the last frame for Social Forces in Figure 12, where it overshoots and loops around the goal. It also had one failure due to *Timeout* in which it was stuck in a local minimum because of the relative configuration of the goal position and the environmental obstacles (it was unable to navigate around a corner to a goal).

ORCA tried to strike a middle ground between the Social Forces model and CADRL in terms of goal-seeking aggressiveness. Its pedestrian avoidance was more locally focused than Social Forces and similar to SA-CADRL in that it tried to deal with individual pedestrians closest to its path

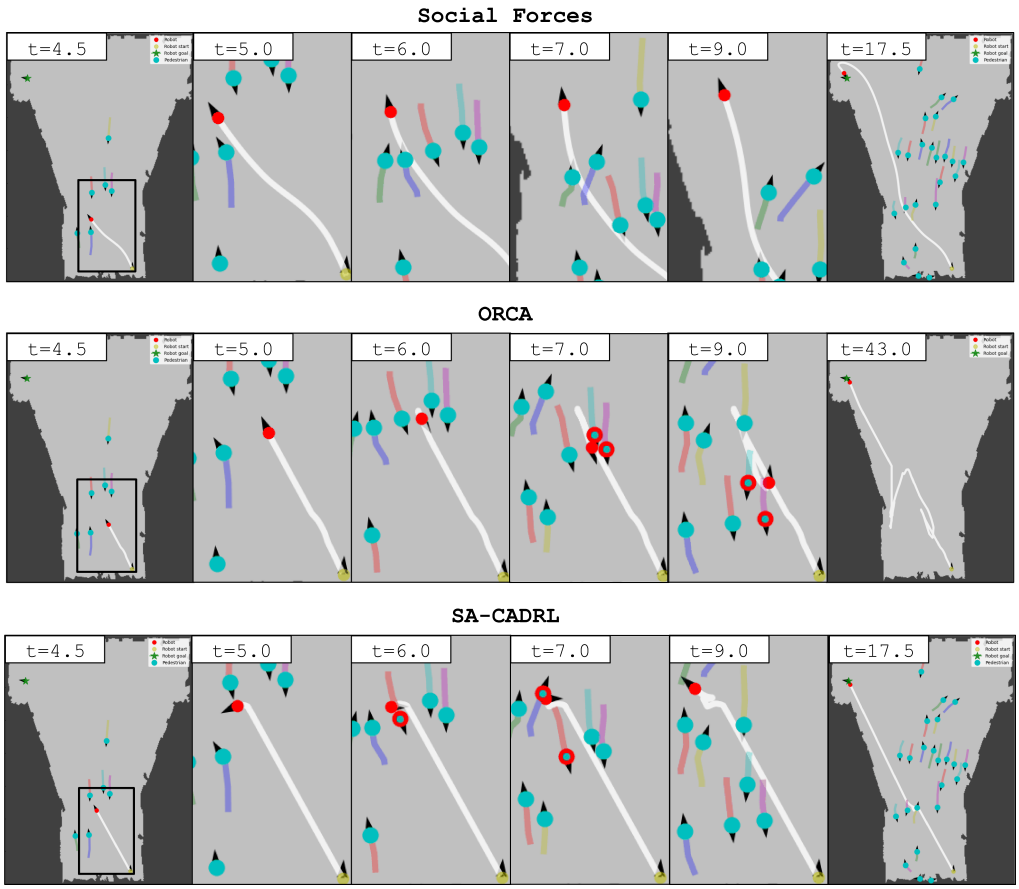


Fig. 12. Qualitative comparisons for candidate algorithms' pedestrian crossing behavior tested on *SocNavBench*. The leftmost image shows the simulated robots about to cross a dense pedestrian crowd to get to their goal. The rightmost image shows the simulated robots just about to reach the goal. Pedestrians with whom collisions occurred are indicated with a red border. A discussion of the algorithms' behavior can be found in Section 5.2.

rather than planning around pedestrians or groups of pedestrians as a whole. Most often, the local avoidance behaviors amounted to the robot trying to escape pedestrians by straying from its preferred trajectory. This behavior can be observed in Figure 12. At $t = 5$, the robot is still proceeding toward the goal, but turns around and starts to move away from the approaching pedestrians once they get too close. However, it reacts too late, and ends up with colliding with multiple pedestrians. This behavior is repeated with subsequent groups of approaching pedestrians as can be seen from the path history and long total time taken as indicated in the last frame. Once the pedestrians had passed and were no longer on the robot's subsequent path segments, it started to plan toward the goal.

SA-CADRL was the most aggressive of the three social navigation models in terms of goal-seeking behavior. It generated direct paths to goal and deals with pedestrians via local avoidance behaviors. Most often, it would stop and turn away from pedestrians but would not move to accommodate them if they kept moving on their trajectory (Figure 12). Therefore, in situations where the robot takes precedence over surrounding pedestrians while still trying to avoid them, such

as when ferrying emergency supplies through a crowded environment, SA-CADRL would be an appropriate choice due to its direct and efficient paths, which still have some pedestrian avoidance capabilities. In contrast, if the robot takes less precedence than pedestrians who share space with it (such as a laundry collecting robot in a hospital), Social Forces may be an appropriate model, since it yields to other agents with higher priority (such as healthcare workers or patients with limited mobility).

Our findings are partly corroborated by previous work. The experiments conducted by the authors of SA-CADRL [10] showed, in agreement with our findings, that SA-CADRL is faster than ORCA in terms of path traversal time. However, they showed that SA-CADRL has shorter minimum separation distance to pedestrians (analogous to our *Closest pedestrian distance*), while we observe similar measurements in this metric (see Figure 6(a)). One difference between our experiment and theirs is that they used a fixed two or four-agent setting where every agent follows the same policy whereas our agent is considered invisible to the real world pedestrians. Apart from traversal time and minimum separation distance, no additional metrics are mentioned in this work. In terms of the Social Forces policy, some works highlighted its fast computation time [48] and high success rates [12] similar to what we observed, but to the best of our knowledge, no prior work evaluated the Social Forces model in the same experimental setting as ORCA or CADRL. The lack of a descriptive suite of metrics and comparisons across different policies in a formal setting further highlights the importance of our benchmark.

5.3 Considerations Around Pedestrian Trajectory Replay

SocNavBench uses a replay style simulation, meaning that simulated pedestrians execute pre-recorded trajectories and are not responsive to the robot's actions. This replay style method introduces some limitations to the benchmark that we believe are mitigated in many scenarios. In this section, we describe our assumptions about scenarios where SocNavBench works well, and we detail the limitations of our benchmark. Additionally, we contrast the tradeoff between our approach and a reactive pedestrian simulation.

The key assumption that underlies SocNavBench metrics is that interrupting pedestrians by inducing deviations from their preferred trajectory is a sign of poor social navigation. For example, the distribution of the *Time-to-collision* metric (Figure 6(b)) quantifies how much the navigation algorithm puts the robot in "close to collision" scenarios. Algorithms that produce consistently small times-to-collision have a higher likelihood of inducing responsive humans to deviate from their originally preferred trajectory, and thus rank less well.

We believe the aforementioned assumption holds for pedestrians in larger, densely crowded areas where typically flows of pedestrians emerge, such as a large atrium or wide sidewalk. In such a situation, it would be less socially appropriate for a single agent (robot or human) to disrupt these emergent pedestrian flows by cutting off pedestrians or trying to "shoot the gap" and failing. Since our benchmark and curated episodes involves large outdoor maps with an average of 44 pedestrians per episode, this assumption is appropriate for the scenarios described in *SocNavBench*.

In contrast, consider a scenario involving a single pedestrian in a supermarket aisle or hospital hallway. This individual pedestrian may walk down the middle of the aisle in the absence of a second agent, but it is still socially acceptable for them them to move closer to one side to allow an opposing agent to pass. In this case, the robot inducing a deviation in the pedestrian's motion is not a sign of poor social navigation; on the contrary, it reflects effective negotiations of space. In these scenarios, SocNavBench metrics do not capture the social aspects of navigation well.

Our non-reactive agent assumption can still be useful for some real-world situations in which reactive modeling may seem imperative. Consider smaller environments, such as the aforementioned hospital scenario—if the other agents in the scene take much higher precedence over the

robot, then our approach of minimizing deviations of others at the expense of the robot is still useful. Such a precedence order would be common in a hospital where a robot such as the TUG [46] ferries laundry and other non-critical supplies between locations. In this situation, it would be undesirable for the TUG bot to interrupt the paths of critical operations such as care providers [39], patients being moved, or even individuals for whom path changes are expensive, such as wheelchair users and assisted walkers.

We chose not to include pedestrian reactivity in this version of SocNavBench, mainly because reactivity adds a different set of limitations, mainly in the form of introducing biases. To incorporate reactive pedestrians, we would need to use a pedestrian movement model that can react to a simulated robot. Since no perfect pedestrian model exists, any model will include assumptions about the nature of pedestrian walking policies, introducing biases on various axes including preferred speeds, accelerations, acceptable closeness to others, and so on. Mitigating these biases requires considerable human data and analysis, which would require a full, independent research project. Additionally, such a benchmark would be prescriptive of a particular type of pedestrian navigation as extensible to all situations. Instead, *SocNavBench* is descriptive and the metrics herein can be interpreted as appropriate for the particular navigation context of candidate algorithms.

Given the above considerations, we understand we cannot fully simulate the behavior of real *reactive* humans, so real-world experiments will still be necessary to provide a complete picture of social navigation performance. Hence, *SocNavBench* can serve as a complementary tool to assist in the development of social navigation algorithms by providing an interpretable performance benchmark that is cheaper to run and more consistent across algorithms than experiments in the real world, while being representative of real world pedestrian behavior—which is a well accepted paradigm in human-robot interaction [47]. This will also allow users to identify which parameters and algorithms to select prior to a real-world evaluation.

Although navigation scenarios in which it is important to model pedestrian reactivity are not well represented in *SocNavBench* as presently constructed, the benchmark is built to be extensible to these scenarios. SocNavBench has an object-oriented structure, and we provide a base `Human` class, which is extended by a `PrerecordedHuman` class to achieve our pedestrian replay. The `update` methods of this class can be overridden to support reactive pedestrians, if researchers concerned with a particular type of pedestrian behavior have a policy model for it.

5.4 Future Work

In the future, we plan to add an option to enable partial reactivity in the pre-recorded pedestrians so that they can react to imminent collisions by switching to a *situationally appropriate* pedestrian motion model with the same goal as the original pre-recorded pedestrian. This will allow us to add metrics to our evaluation suite that measure the number of times a particular social navigation algorithm causes pedestrian route changes as well as the amount of deviation caused. For added flexibility and ease of use in robot simulation, we also plan to provide a direct ROS interface via ROS messages (current sockets allow python and C++ ROS nodes to interact with *SocNavBench* via native types) as well as support for custom robots through a URDF specification.

6 CONCLUSION

We present *SocNavBench*, an integrated testing and evaluation framework for social navigation algorithms. *SocNavBench* comprises a photorealistic simulator and a curated set of social navigation scenarios based on real-world pedestrian trajectory data on which social navigation algorithms can be tested. We also provide an implementation of a suite of metrics that are useful for evaluating the performance of these algorithms along various criteria including robot navigation efficiency and pedestrian disruption. Finally, we demonstrate the use of *SocNavBench* by

evaluating a set of three existing social navigation methods. *SocNavBench* is publicly available at https://github.com/CMU-TBD/SocNavBench.

ACKNOWLEDGMENTS

We thank the anonymous reviewers of this work who helped improve the article in several areas, including clarity surrounding technical details and the arguments around the limits of pedestrian trajectory replay.

REFERENCES

[1] Adept Technology Inc. 2020. Pioneer3DX P3DX-RevA Specifications. Retrieved from https://www.generationrobots.com/media/Pioneer3DX-P3DX-RevA.pdf.

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*.

[3] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 2016. 3D semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1534–1543.

[4] Stefan Becker, Ronny Hug, Wolfgang Hübner, and Michael Arens. 2018. An evaluation of trajectory prediction approaches and notes on the trajnet benchmark. Retrieved from https://arxiv:1805.07663.

[5] Aniket Bera, Tanmay Randhavane, Emily Kubin, Austin Wang, Kurt Gray, and Dinesh Manocha. 2018. The socially invisible robot navigation in the social world using robot entitativity. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'18)*. IEEE, 4468–4475.

[6] A. Bera, T. Randhavane, R. Prinja, and D. Manocha. 2017. SocioSense: Robot navigation amongst pedestrians with social and psychological constraints. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*. 7018–7025.

[7] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Haenel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. 1999. Experiences with an interactive museum tour-guide robot. *Artific. Intell.* 114, 1 (1999), 3–55.

[8] C. Cao, P. Trautman, and S. Iba. 2019. Dynamic channel: A planning framework for crowd navigation. In *Proceedings of the International Conference on Robotics and Automation (ICRA'19)*. 5551–5557.

[9] C. Chen, Y. Liu, S. Kreiss, and A. Alahi. 2019. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *Proceedings of the International Conference on Robotics and Automation (ICRA'19)*. 6015–6022.

[10] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P. How. 2017. Socially aware motion planning with deep reinforcement learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*. IEEE, 1343–1350.

[11] N. E. Du Toit and J. W. Burdick. 2012. Robot motion planning in dynamic, uncertain environments. *IEEE Trans. Robot.* 28, 1 (2012), 101–115.

[12] Gonzalo Ferrer, Anais Garrell, and Alberto Sanfeliu. 2013. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1688–1694.

[13] Paolo Fiorini and Zvi Shiller. 1998. Motion planning in dynamic environments using velocity obstacles. *Int. J. Robot. Res.* 17, 7 (1998), 760–772.

[14] Amalia F. Foka and Panos E. Trahanias. 2010. Probabilistic autonomous robot navigation in dynamic environments with human motion prediction. *Int. J. Soc. Robot.* 2, 1 (2010), 79–94. https://doi.org/10.1007/s12369-009-0037-z

[15] D. Fox, W. Burgard, and S. Thrun. 1997. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* 4, 1 (1997), 23–33.

[16] Christian Gloor. 2020. *libPedSim*. Retrieved from http://pedsim.silmaril.org/.

[17] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*.

[18] Jérôme Guzzi, Alessandro Giusti, Luca M. Gambardella, Guy Theraulaz, and Gianni A. Di Caro. 2013. Human-friendly robot navigation in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 423–430.

[19] Dirk Helbing and Peter Molnar. 1995. Social force model for pedestrian dynamics. *Phys. Rev. E* 51, 5 (1995), 4282.

[20] P. Henry, C. Vollmer, B. Ferris, and D. Fox. 2010. Learning to navigate through crowded environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 981–986.
[21] Joshua Joseph, Finale Doshi-Velez, and Nicholas Roy. 2010. A bayesian nonparametric approach to modeling mobility patterns. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI'10)*. AAAI Press, 1587–1593.
[22] I. Karamouzas and M. Overmars. 2012. Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Trans. Visual. Comput. Graph.* 18, 3 (2012), 394–406.
[23] Beomjoon Kim and Joelle Pineau. 2016. Socially adaptive path planning in human environments using inverse reinforcement learning. *Int. J. Soc. Robot.* 8, 1 (2016), 51–66. https://doi.org/10.1007/s12369-015-0310-2
[24] Rachel Kirby. 2010. Social robot navigation. https://www.ri.cmu.edu/pub_files/2010/5/rk_thesis.pdf.
[25] Parth Kothari, Sven Kreiss, and Alexandre Alahi. 2021. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Trans. Intell. Transport. Syst.* (2021), 1–15. DOI: 10.1109/TITS.2021.3069362
[26] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. 2016. Socially compliant mobile robot navigation via inverse reinforcement learning. *Int. J. Robot. Res.* 35, 11 (2016), 1289–1307.
[27] T. Kruse, P. Basili, S. Glasauer, and A. Kirsch. 2012. Legible robot navigation in the proximity of moving humans. In *Proceedings of the IEEE Workshop on Advanced Robotics and Its Social Impacts (ARSO'12)*. 83–88.
[28] Steven M. LaValle and James J. Kuffner Jr. 2001. Randomized kinodynamic planning. *Int. J. Robot. Res.* 20, 5 (2001), 378–400.
[29] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. 2007. Crowds by example. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 655–664.
[30] Matthias Luber, Luciano Spinello, Jens Silva, and Kai O. Arras. 2012. Socially-aware robot navigation: A learning approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 902–907.
[31] Luis J. Manso, Pedro Nunez, Luis V. Calderita, Diego R. Faria, and Pilar Bachiller. 2019. SocNav1: A dataset to benchmark and learn social navigation conventions. Retrieved from https://arXiv:1909.02993.
[32] Christoforos Mavrogiannis, Alena M. Hutchinson, John Macdonald, Patrícia Alves-Oliveira, and Ross A. Knepper. 2019. Effects of distinct robot navigation strategies on human behavior in a crowded environment. In *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI'19)*. IEEE, 421–430.
[33] Christoforos I. Mavrogiannis, Valts Blukis, and Ross A. Knepper. 2017. Socially competent navigation planning by deep learning of multi-agent path topologies. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*. IEEE, 6817–6824.
[34] Christoforos I. Mavrogiannis and Ross A. Knepper. 2016. Decentralized multi-agent navigation planning with braids. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR'16)*.
[35] Christoforos I. Mavrogiannis, Wil B. Thomason, and Ross A. Knepper. 2018. Social momentum: A framework for legible navigation in dynamic multi-agent environments. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 361–369.
[36] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. 2020. Social-STGCNN: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*.
[37] Mehdi Moussaïd, Dirk Helbing, Simon Garnier, Anders Johansson, Maud Combe, and Guy Theraulaz. 2009. Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proc. Roy. Soc. B: Biol. Sci.* 276, 1668 (Aug. 2009), 2755–62. https://doi.org/10.1098/rspb.2009.0405
[38] Mehdi Moussaïd, Niriaska Perozo, Simon Garnier, Dirk Helbing, and Guy Theraulaz. 2010. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PLoS One* 5, 4 (Apr. 2010), 1–7. https://doi.org/10.1371/journal.pone.0010047
[39] Bilge Mutlu and Jodi Forlizzi. 2008. Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI'08)*. IEEE, 287–294.
[40] B. Okal and K. O. Arras. 2016. Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'16)*. 2889–2895.
[41] Sebastien Paris, Julien Pettre, and Stephane Donikian. 2007. Pedestrian reactive navigation for crowd simulation: A predictive approach. *Comput. Graph. Forum* 26, 3 (2007), 665–674. https://doi.org/10.1111/j.1467-8659.2007.01090.x
[42] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proceedings of the IEEE 12th International Conference on Computer Vision*. IEEE, 261–268.
[43] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. 2016. Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proceedings of the European Conference on Computer Vision (ECCV'16)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 549–565.

[44] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. 2019. SoPhie: An attentive GAN for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*.

[45] Masahiro Shiomi, Francesco Zanlungo, Kotaro Hayashi, and Takayuki Kanda. 2014. Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model. *Int. J. Soc. Robot.* 6, 3 (2014), 443–455. https://doi.org/10.1007/s12369-014-0238-y

[46] ST Engineering Aethon Inc. (2021). *TUG Robot Product Page*. Retrieved from https://aethon.com/products/.

[47] Aaron Steinfeld, Odest Chadwicke Jenkins, and Brian Scassellati. 2009. The oz of wizard: Simulating the human for interaction research. In *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction (HRI'09)*. Association for Computing Machinery, New York, NY, 101–108. https://doi.org/10.1145/1514095.1514115

[48] Avneesh Sud, Russell Gayle, Erik Andersen, Stephen Guy, Ming Lin, and Dinesh Manocha. 2008. Real-time navigation of independent agents using adaptive roadmaps. In *Proceedings of the ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH'08)*. Article 56, 10 pages.

[49] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. 1999. MINERVA: A second-generation museum tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3.

[50] Varun Tolani, Somil Bansal, Aleksandra Faust, and Claire Tomlin. 2020. Visual Navigation Among Humans with Optimal Control as a Supervisor. Retrieved from https://arXiv:2003.09354.

[51] Pete Trautman. 2017. Sparse interacting gaussian processes: Efficiency and optimality theorems of autonomous crowd navigation. In *Proceedings of the IEEE 56th Annual Conference on Decision and Control (CDC'17)*. IEEE, 327–334.

[52] Peter Trautman and Andreas Krause. 2010. Unfreezing the robot: Navigation in dense, interacting crowds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 797–803.

[53] Pete Trautman, Jeremy Ma, Richard M. Murray, and Andreas Krause. 2015. Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation. *Int. J. Robot. Res.* 34, 3 (2015), 335–356. https://doi.org/10.1177/0278364914557874

[54] Adrien Treuille, Seth Cooper, and Zoran Popović. 2006. Continuum crowds. ACM Trans. Graph. 25, 3 (July 2006), 1160–1168. https://doi.org/10.1145/1141911.1142008

[55] Xuan-Tung Truong and Trung-Dung Ngo. 2016. Dynamic social zone based mobile robot navigation for human comfortable safety in social environments. *Int. J. Soc. Robot.* 8, 5 (5 2016), 663–684. https://doi.org/10.1007/s12369-016-0352-0

[56] Xuan-Tung Truong and Trung Dung Ngo. 2017. Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model. *IEEE Trans. Autom. Sci. Eng.* 14, 4 (2017), 1743–1760.

[57] J. van den Berg, Ming Lin, and D. Manocha. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 1928–1935.

[58] Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, and Ming Lin. 2008. Interactive navigation of multiple agents in crowded environments. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'08)*. Association for Computing Machinery, New York, NY, 139–147. https://doi.org/10.1145/1342250.1342272

[59] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. 2017. Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 109–117.

[60] Dizan Vasquez, Billy Okal, and Kai O. Arras. 2014. Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1341–1346.

[61] A. Vemula, K. Muelling, and J. Oh. 2018. Social attention: Modeling attention in human crowds. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'18)*. 4601–4607.

[62] Shuai Yi, Hongsheng Li, and Xiaogang Wang. 2015. Understanding pedestrian behaviors from stationary crowd groups. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*.

[63] F. Zanlungo, T. Ikeda, and T. Kanda. 2011. Social force model with explicit collision prediction. *Europhys. Lett.* 93, 6 (Mar. 2011), 68005. https://doi.org/10.1209/0295-5075/93/68005

Received October 2020; revised June 2021; accepted July 2021