EU Horizon 2020 Research & Innovation Program

Advanced Robot Capabilities & Take-Up

ICT-25-2016-2017

Safe Robot Navigation in Dense Crowds

http://www.crowdbot.org

Technical Report

D 3.3: Local Interaction-Aware Motion Planning

Work Package 3 (WP 3)

Navigation

Task Lead: Swiss Federal Institute of Technology (ETHZ), Switzerland

WP Lead: Swiss Federal Institute of Technology (ETHZ), Switzerland

Table of Contents

Table of Figures

Executive Summary

State-of-the-art approaches for robot navigation among humans are typically restricted to planar movement actions. This work addresses the question of whether it can be beneficial to use interaction actions, such as saying, touching, and gesturing, for the sake of allowing robots to navigate in unstructured, crowded environments. To do so, we first identify challenging scenarios to traditional motion planning methods. Based on the hypothesis that the variation in modality for these scenarios calls for significantly different planning policies, we design specific navigation behaviors, as interaction planners for actuated, mobile robots. We further propose a high-level planning algorithm for multi-behavior navigation, named Interaction Actions for Navigation (IAN). Through both real-world and simulation experiments, we validate the selected behaviors and the high-level planning algorithm, and discuss the impact of our obtained results on our stated assumptions. Our IAN planner will be open sourced at https://github.com/ethz-asl/interaction_actions_for_navigation.

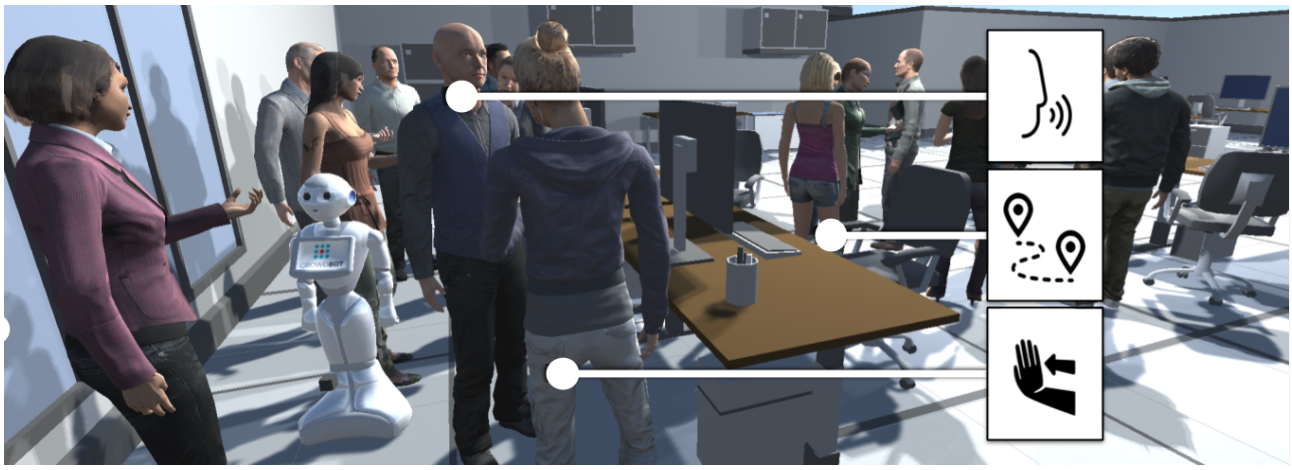

Figure 1. The IAN planning approach combines multiple interactive robot behaviors, which implement gestures, speech and assertive motion in order to interact with humans and move efficiently through dense crowds.

1. Introduction

From vacuum cleaner to store assistant: industry is exploring the idea of bringing robots into our daily lives. More ubiquitous robots implies more robust robots, but also more intuitive and interactive, more mobile, more lifelike robots. It also means being surrounded by humans, in partially known environments and contexts.

This has given rise to an increase in research interest in autonomous navigation among humans, and as pioneering work of many of our predecessors show, has led to the discovery of plenty of hard-to-solve challenges.

Though navigation among humans can be treated as a geometrical problem, where the robot is considered as a fixed shape moving through a dynamic world, we examine whether imitating life with gestures, speech, and interaction can lead to new ways of approaching the navigation problem. Here, we focus especially on robots with movable bodies, though not necessarily human-like.

Adding gestures and speech to the already complex navigation problem implies further increasing the dimensionality of the configuration, and planning space that we operate in. It follows that a single navigating, gesturing, interactive planner requires a tremendous amount of complexity. For clarity, we use the term **navigation behaviors** to refer to the interactive, motion + gesture + speech actions that a robot may perform in the context of navigation. An intuitive look at how humans solve such problems led to the hypothesis that it is beneficial to be able to switch between distinct, expert planners that specifically target different situations. This raised the following questions, which we address in this work: When is multi-modal navigation useful? How is the performance of multi-modal navigation in comparison to classical planning? Which navigation modes are necessary, and in what situations?

In this work, we:

1) Identify useful interaction behaviors and implement them as motion, gesture and speech planners.
2) Present the IAN planning approach: A high-level, multi-behavior, interaction-aware planning framework for navigation in unstructured, human-populated environments.
3) Evaluate the presented framework in real-world environments.

To precisely describe the characteristics of challenging scenarios, we define the terms of crowd perceptivity, permissivity and crowdedness, which describe the ability of participants in the crowd to perceive the robot's intended motion, the willingness of those participants to allow the robot to achieve its goal, and the overall density of the crowd.

We refer to our multi-behavior planning approach as IAN—Interactive Actions for Navigation. We implement and test three navigation behaviors: an optimistic behavior, referred to as *Intend*, a communicative behavior referred to as *Say*, and an assertive behavior, referred to as *Nudge*.

2. Related Work

Navigating robots alongside humans has been a goal and topic of research for several decades. One of the earliest motion planning approaches designed for dynamic environments is the velocity obstacles (VO) planner [1]. VO generates paths for a holonomic robot under the assumption that other agents in the space will continue to move with their current velocity. The planner behaves predictably and works well in simple environments. Nevertheless, VO is a reactive method and does not model the behavior of obstacles and agents in a realistic way. In addition, it cannot account for mutually beneficial behavior between agents, such as joint planning and communication. Indeed, several studies [2–7] have highlighted the potential for cooperative planning to improve not only the navigation efficiency but also human acceptance of the robot. In particular, Trautman et al. [8] observed that the baseline Dynamic Window Approach (DWA) [9] for local planning fails at crowd densities above 0.55 persons per square meter (p/m^2), whereas their cooperative planning approach was capable of operating in densities of up to $0.8p/m^2$. Yet, beyond these densities, even cooperative planning begins to break down.

To improve upon this, several works turn to human modelling and reciprocal planning. Alonso-Mora et al. [10] proposed reciprocal velocity obstacles (RVO), an extension of the VO approach to holonomic agents, implementing reciprocal planning with the assumption that agents share the collision-avoidance effort. Some use hand-crafted models, such as Rudenko et al. [11], who presented a prediction framework based on Monte Carlo simulation of various possibilities, where agent behaviors are computed using the social force model [12]. However, this requires known goals for the simulated agents, which is not necessarily available to a robot moving in an organic human crowd.

Recently, data-driven approaches have used real-world demonstrations to learn motion planners. For example, Pfeiffer et al. [13] train a long short-term memory (LSTM) neural network on several pedestrian datasets [14] to predict human trajectories. The assumption here is that the learned models are able to capture useful patterns in human behavior without having to specifically bake in social norms or human-motion models, which might be incomplete or partially superfluous. Other approaches have used interacting Gaussian processes (IGP) [15], apprenticeship learning [16] as well as maximum entropy reinforcement learning (RL) [17, 18]. However, a persistent challenge for data-driven methods is to ensure the prediction of reasonable paths, or to indicate uncertainty when an agent's trajectory history is only partially known. This can occur frequently in a crowd when agents occlude one another from the robot sensors, leading to lost tracks.

Deep RL approaches have also demonstrated the ability to learn control policies for navigating in human crowds. Some favor an end-to-end approach [19], while others first process the robot sensor data to extract pedestrian tracks, which form part of the input state [20]. Further, RL methods are able to incorporate social norms, such as the left-right passing rule mentioned in [21], by designing reward functions that encourage socially acceptable behavior [22–24]. Nevertheless, these methods focus on generating policies that only control the motion of the robot base. In our work, we hypothesize that the incorporation of multi-modal behaviors, beyond simply the movement of the base, can enable more meaningful robot-crowd interactions that ultimately allow the robot to reach its goal faster.

Recent works have begun to consider the problem of navigating among humans as a multi-behavior planning task. Approaches combine a learned navigation policy with manually designed recovery procedures for use when, for example, localization fails [25–27]. Chen et al. [28] introduced multiple travel modes, such as a follow mode or a probing mode, using a heuristic approach for the mode selection. In addition, each of these methods have only been tested in scenarios where the crowd participants, while not necessarily cooperative, still move in a goal-seeking fashion, without any active interaction with the robot. Yet, we have observed that compared with pedestrian-pedestrian interactions, when faced with a navigating robot, humans exhibit quite different, and sometimes antagonistic, behaviors such as aggressively blocking the robot's path [18]. In our work, we investigate the efficacy of more assertive robot behaviors, and the use of other interaction modalities, such as gestures, touch and verbal communication, for crowd navigation.

3. Navigation Behaviors

Previous work identified several scenarios that pose important challenges to planners when navigating robots in crowded environments. Among those are:

1) Static crowds with non-cooperative agents blocking the robot path, which can lead to the frozen robot problem.
2) Cooperative but clueless agents that can result in mismatched trajectories due to a lack of communicated intent from the robot.
3) Crowds presenting dense flow characteristics in which the neutral behavior should involve constant motion.

In this work, we focus on the first two scenarios, which already present significant variation in robot-crowd interactions. This motivates the need for separate movement and interaction modalities over a monolithic navigation method. To explore this, we identified a subset of simple navigation behaviors that can efficiently cover as much of this scenario space as possible. These are made available to our high-level behavior planning algorithm which selects behaviors based on the observed state.

3.1. Intend: Smooth Sailing in Sparsely Occupied Space

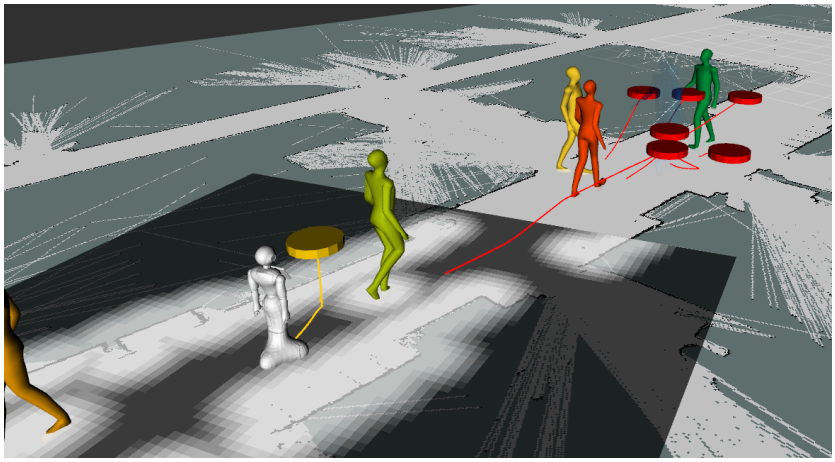

Figure 2. Predicted trajectories for the *Intend* behavior, which targets scenarios with few, goal-driven pedestrians.

In order to handle the general case, the first selected behavior consists in ordinary motion planning. To navigate to the goal, the planning model assumes a static or sparsely dynamic environment and returns safe trajectories. We use the term **crowdedness** to refer to the local crowd density from the point of view of state estimation and behavior selection. Many basic navigation approaches are relevant for this case and can be used to implement the behavior. Here we generate a local velocity field from instantaneous LiDAR data for static obstacles, and model dynamic agents using the RVO approach—setting their tracked velocities as desired velocities—to get predicted trajectories for the agents and to compute a corresponding trajectory for the robot. See Figure 2 for an example of the *Intend* behavior planning in a sparse environment.

3.1.1. Intend RVO Planner

In order to efficiently predict agent and robot paths, first, a desired velocity vector is obtained for both agents and robot. For the agents, a constant velocity model is used. Though simplistic, it can be a reasonable naïve prior given the lack of knowledge about actual agent goals and intentions. In the case of the robot, as the goal is known, a desired velocity field is computed for each point in 2D space, considering only static obstacles. To compute this field, the Fast Marching algorithm is used, and space close to obstacles is given a lower speed as penalty. This 'speed lowering factor' is computed as $S = \left(\frac{d}{0.7}\right)^2$, where d is the distance in meters to the nearest obstacle, and S is clipped to $[0.1,1]$.

Once desired velocities are obtained for robot and agents, they are passed as the input of an RVO simulation, which also includes mapped static obstacles. The resulting velocities are used to forward simulate one step. This is repeated until a resulting velocity sequence and trajectory is obtained for both robot and agents. The RVO implementation used is the widely available RVO2 library, and fine tuning of internal parameters was not necessary.

3.2. Say: Communicating the Robot's Goal

In order to help cooperative agents anticipate the robot's actions and intent, verbal and physical communication can be used. We use the term **perceptivity** to refer to the level of awareness that an agent has of the robot's goals and intentions. The goal of this behavior is therefore to increase this perceptivity, in order to decrease the likelihood of cooperation failures. In the *Say* mode, the robot verbally announces that it wants to move ("excuse me, I'm coming through"), indicates the direction with its hand while moving the base in the announced direction. The base motion planning used to implement this behavior is fundamentally the same as for the *Intend* behavior but with modified parameters, 25% lower desired velocity and a more optimistic estimation of nearby agent cooperation through 50% smaller agent radii in the RVO calculation. The goal of this behavior is to increase the perceptivity of the nearby crowd and thereby decrease the likelihood of cooperation failures.

3.3. Nudge: Unfreezing the Robot

Cooperative planning approaches such as [8, 10, 13] typically expect that other agents will assume part of the avoidance responsibility. However, an often-observed behavior is for humans to react with curiosity to the robot's presence by standing in front of it and interacting with it, effectively blocking its path. In this work, we refer to this behavior as 'playing' with the robot.

Due to these observations, we hypothesize that an extension to the cooperative/non-cooperative distinction is useful for navigating in dense crowds: Non-cooperative agents might have high **permissivity**. That is, a human might 'play' with the robot, blocking it (non-cooperative), and yet be willing to let it through once the robot shows clear intent to pass. Conversely, these same non-cooperative agents might be non-permissive. In which case, the *Intend* and *Say* behaviors will still be ineffective in allowing the robot to get past the agent. We propose an assertive behavior, *Nudge*, which targets scenarios of high crowdedness, and/or low (non-zero) permissivity.

Due to planning in close proximity to dynamic agents, we picked a low-latency reactive approach, where motion planning depends only on instantaneous LiDAR information. The DWA is used to select forward/lateral velocities, based on a mixed objective of distance keeping and progress towards goal: $J_{DWA} = p_o + 0.1p_g$, where p_o is the increase in minimum distance to obstacles in the LiDAR scan and p_g is the decrease in distance to goal in meters. Constraints are added such that p_o and p_g must be non-negative. When no non-zero solution is found, and humans are detected in front of the robot, the robot verbally announces "I'm passing through, thank you," while bringing one arm forward, reaching over its own base footprint. The velocity is greatly reduced (0.1m/s) and the p_o constraint is temporarily lifted, allowing the robot to 'push-through' nearby obstacles.

The algorithm is described in detail in CROWDBOT D31, Section 5.2 "Local, low-latency planning".

4. IAN: Interaction Actions for Navigation

In this section, we formulate the planning problem where the robot must not only decide the physical path that it will traverse but also the sequence of behaviors that it will execute along the path.

4.1. Problem Formulation

The navigation objective is modelled as a search problem: starting from an initial state (a combination of robot position and measured crowd state, which includes crowdedness, perceptivity and permissivity), find the sequence of transitions ending at the desired state, which minimizes some cost function. In practice the state can rarely be fully observed. For this reason, we model the state as a probabilistic distribution over possible states $p(S)$.

In the case of navigation behaviors, the transition to a new state S' can be modeled as depending on three aspects:

1) the initial state S,
2) the selected behavior b, and

3) the selected path Ξ, where the latter two are determined by the high-level planner.

Note that we are also specifically interested in the outcome probability $p(o)$ of executing b along Ξ given the initial state S. This can be computed as an intermediate step of the transition and aids in the computation of the transition cost.

With the state and transitions defined, Monte Carlo Tree Search (MCTS) can be applied to randomly sampled transitions from the current state. Each trial explores one branch of the possibility tree, ending when the goal, or some termination condition, is reached. The final cost of a single plan, from root to leaf node, can be obtained by adding the costs for each sequential transition. The result is a tree of explored states, as represented in Figure 3, along with trial statistics (i.e. how often a behavior achieves a successful outcome) for each node and transition in the tree. This allows us to compute, for any node or transition in the tree, the ratio of goal-reaching plans vs. total plans passing through this node/transition, and the average final cost of all goal-reaching plans passing through this node/transition.

Given some objective function for comparing expected final cost distribution of possible transitions, it is possible to greedily select a sequence of transitions which approximates the optimal plan as the sampling size increases.

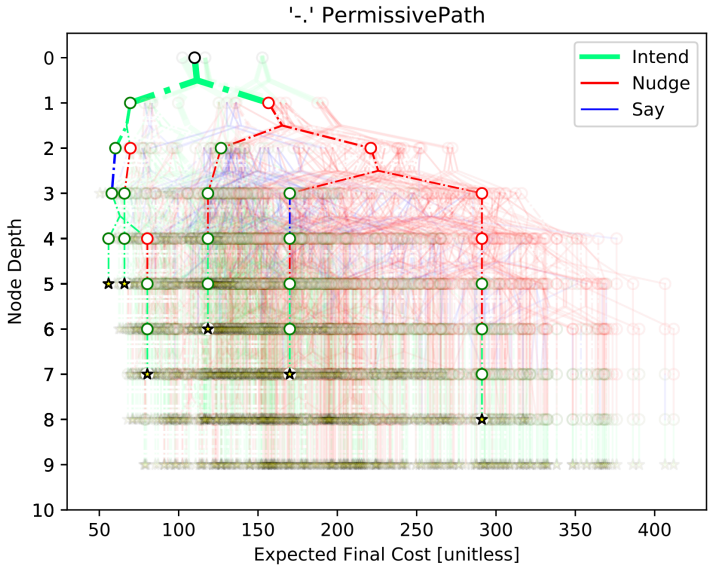

Figure 3. Tree representation of the POMDP states explored through MCTS (transparent). States where the goal is reached are shown with a star marker. The horizontal axis measures the mean final cost of all trials which transitioned through a given state. Overlaid is the final plan, consisting of the subset of the explored tree which is pruned by greedily picking a behavior at each state based on the MCTS statistics.

4.2. Our Implementation

4.2.1. State Representation

In this work, the state representation was selected empirically based on our set of behaviors, as well as available sensor modalities. It consists of the current (x, y) position of the robot and what we term the *local state features* s_i, which represents the crowdedness, perceptivity and permissivity at each cell i in the discretized grid map, where each feature is modeled independently as a normal distribution.

4.2.2. Discrete Paths, Behaviors, Outcomes

While the ideal space of possible paths, behaviors, outcomes, and thus transitions are continuous, in this work we reduce them to discrete approximations, for tractability. The space of paths was reduced to a discrete sampling of several path options, obtained based on the estimated state features of the initial map. This allows for

reduced computational cost, but constrains the planning to only considering local effects of actions. For more general problems it is necessary to re-sample paths after each transition.

Our approach for generating the path options was to use Fast Marching with a Euclidean signed distance field cost near static obstacles. We mark regions where selected features are below a certain threshold as non-traversable (i.e. treat them as static obstacles). The different path options were generated by varying these threshold values. This results in, for example, the 'naïve' path option, where only the static map is considered, or the 'permissive' path option, which treats only static obstacles and low permissivity regions as non-traversable, and so on. Examples of such path options are shown in Figure 4. To reduce confusion, we make a distinction between the path options Ξ, which are computed once for each planning loop, and the path segments or **subpaths** ξ on which behaviors are applied. These subpaths are re-sampled at every transition.

The set of possible behaviors $B = \{b_1, b_2, b_3\}$ contains those described in Section 3, i.e. *Intend*, *Say*, *Nudge*. However, the execution duration for each behavior is not fixed. Thus, at every state and for each behavior, a discrete set of subpaths must be defined to indicate the transition from one behavior to another. Rather than sampling randomly to obtain this set, we compute the regions along the set of path options (from the robot's current position in the state) where the behavior's estimated success probability is above a defined threshold. As a result, behaviors are only considered in regions of the state where they are relevant.

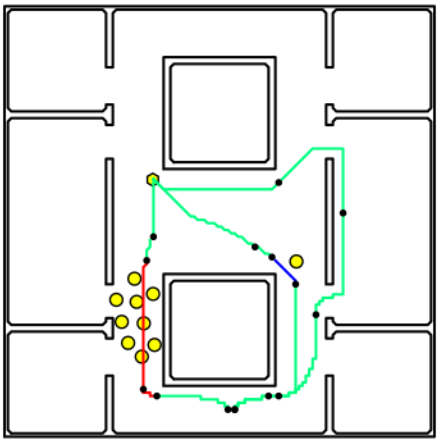

Figure 4. Example of sampled path options from the IAN planner. The robot starts in the top left and must move to the bottom. For each path, the optimal behavior sequence selected by the IAN algorithm is shown. Green, blue and red segments denote using the *Intend*, *Say* or *Nudge* behavior, respectively. Individual subpaths are delineated by black points.

The possible **outcomes** for a given S, b, and ξ are discretely modelled as a Bernoulli process, drawing from a distribution of two outcomes (success/failure) with probability $p_{success}$, $(1 - p_{success})$. While it may be interesting to allow for a more complex representation of outcomes, we note that in practice it implies the ability to differentiate those outcomes based on measurement. This is rather straightforward with a success/failure distinction: our approach consists of predicting a behavior duration (which is also used to predict behavior cost), and assuming a successful outcome if the behavior reaches its subpath destination within this time, otherwise assuming failure.

4.2.3. Behavior Transition Model

At this point, the deterministic transition function $T(S, b, \xi)$, the transition cost $c_o(S, b, \xi)$ and the outcome probability $p(o|S, b, \xi)$ need to be defined for each behavior. The deterministic transition functions $T(S, b, \xi)$ for each behaviour and outcome that we use in our experiments are given in Appendix A. To compute the outcome probability and cost along a subpath, we can consider transitions of the local state s_i across each discrete location i in the map. Thus, the outcome probability along the subpath is computed as the worst-case probability over each discrete grid cell along the subpath,

$$p(o|S, b, \xi) = \min_{i \in \xi} p(o|s_i, b), \tag{1}$$

while the cost function uses an addition of local costs along the subpath,

$$c_o(S, b, \xi) = \sum_{i \in \xi} c_o(x_i, y_i, b). \quad (2)$$

The definitions of the local outcome probabilities and local cost functions for each behavior are given in Appendix A. The probability of the next state $p(S')$ can then be computed in two steps. First, given a state estimate $p(S)$, a behavior and a sampled outcome, we can apply a Bayesian update to obtain a posterior estimate of the state,

$$p(S|o, b) \propto p(o|S, b)p(S), \quad (3)$$

which allows us to reuse terms from (1), i.e. the action worked well, therefore the state was more likely to be in the part of our estimate distribution where this action works well. Then the transition function can be directly applied through another Bayesian update to obtain the posterior estimate of the state after executing the behavior,

$$p(S'|S, o) = T(S, b, \xi)p(S|o). \quad (4)$$

Using these reductions, we can fully define the transitions for every behavior by determining only (1), (2) and (4). The tuple $\langle c_o(S, b), p(o|S, b, \xi), T(S, b, \xi)\rangle$ is referred to as the **behavior transition model**.

Behavior transition models are a critical part of the implementation. Building them requires, for example, testing a navigation behavior in various scenarios and measuring the outcome. However, one advantage of the proposed framework is that since it tracks the observed state and executed behaviors, by running the planner one naturally obtains state transition data for each behavior, which can be used to refine the behavior transition models online.

4.2.4. MCTS

With the behavior transition model defined, Monte Carlo sampling a transition at a single state S consists of first sampling a behavior and subpath from the discrete sets using uniform random sampling. Then sampling an outcome using the outcome probability $p(o|S, b, \xi)$ from (2). We then get the new state S' by applying (3) and (4). Finally, the cost is computed from (1). As stated in Subsection 4.1, MCTS yields the explored tree and final cost distributions over N_{total} trials. Due to the fact that not reaching the goal has a theoretically infinite, yet in practice unknown cost, it is necessary to define how to treat trials that do not reach the goal. We use a simple penalty heuristic where the objective is the mean expected final cost weighted by the ratio of trials that reached the goal state. The greedy objective function is,

$$J(node) = q \cdot \mathbb{E}[c], \quad (5)$$

where the expected cost is computed over all possible outcomes, and q is the penalty term,

$$q = \frac{N_{total}}{N_{goal}}. \quad (6)$$

4.2.5. Hardware System Description

The algorithm was run on the ETHZ CROWDBOT Pepper, which includes the sensing and hardware modifications described in D52 and D53. We used the outputs of two 2D LiDARs, mounted 17cm from the ground, providing 360° range measurements for static obstacle detection and SLAM. A RealSense D435 RGB-D camera positioned on the robot's forehead was used exclusively to detect and track humans in front of the robot. More details on the person detection and tracking algorithms can be found in CROWDBOT D22 Local Sensing 1st Prototype. These detections were used to compute the crowdedness, perceptivity and permissivity of the state. For state estimation, a uniform prior is used to initialize $p(S)$. For each incoming sensor measurement, $p(S)$ is updated through Bayesian inference using a manually designed likelihood model, see Appendix A for details.

In addition, to account for the fact that the state of unobserved areas can change, in the absence of measurement a 'forget' update was executed, updating $p(S)$ towards the uninformed prior. Finally, we consider the outcome of executing behaviors in the crowd as evidence, and update perceptivity and permissivity according to (4).

5. Experiments and Results

5.1. Preliminary Tests Using Nudge in a Real Human Crowd

One of our motivations for multi-behavior navigation planning was the hypothesis that real human-populated environments present different situations where specific behaviors are advantageous. For example, that a planner such as *Nudge*, which uses gestures and assertive motion planning, will successfully navigate in situations where planning to avoid contact provides no feasible solution, or plans inefficiently. To verify this hypothesis, we deployed our robot using the *Nudge* navigation behavior in a real, uncontrolled crowd presenting a density of approximately $2p/m^2$. Figures 5 and 6 show images of the crowd during the experiment.

The tests were set up as follows: Each run consisted of the robot starting at an arbitrary location and being given a goal at another arbitrary location. The robot was then allowed to navigate autonomously through the crowded scene until it either reached the goal or timed-out (after 300s on the simplex and complex maps, after 600s on the realistic map). 24 runs were executed, for a total run time of 37 minutes. To capture a meaningful average velocity, we compute the average progress towards goal for each run by applying a filter on the robot trajectory, thus removing local oscillations, then dividing the total filtered trajectory length by the total time. This metric is referred to as **average progress-towards-goal velocity**, and is measured in m/s. To obtain the weighted mean and standard deviation of average velocities over all runs, each run is assigned a weight proportional to its duration.

Figure 5. Crowd density during the unstructured crowd experiment. Interested, unaware participants are obstructing the robot's path. Nevertheless, the robot was able to reach its goal by using our assertive *Nudge* behavior.

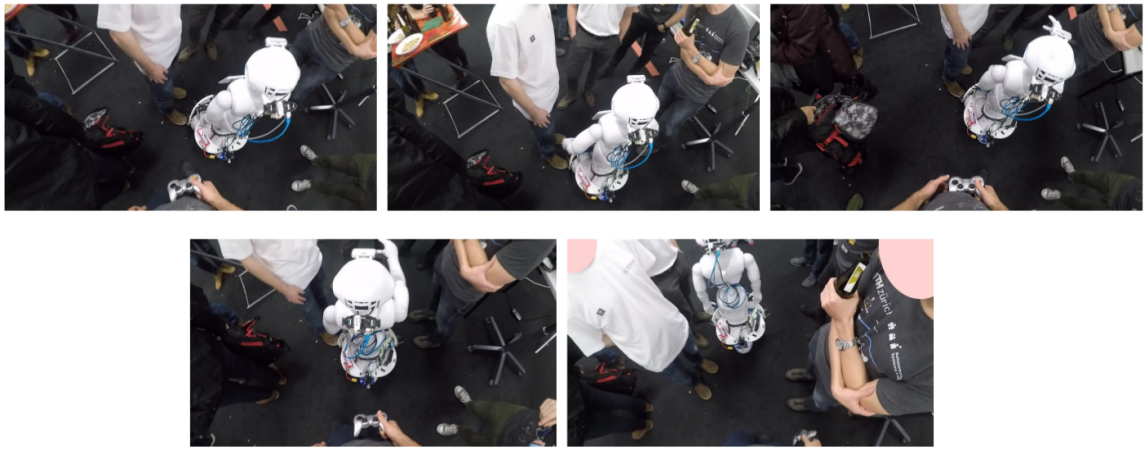

Figure 6. Our robot autonomously executing the *Nudge* behavior in a dense, crowded environment. In the first frame, the controller constraints yield no acceptable velocity, but humans are detected directly

in front of the robot. In the second and third frame, a gesture is performed to gently 'nudge' and indicate the robot's intended direction. In the fourth and fifth frame, the robot advances towards its goal at a constant yet severely reduced velocity.

Table 1. Assertive planner results in real scenario

	μ_w	σ_w^2	min	max
Progress towards goal [m/s]	0.074	0.0011	0.047	0.39
Velocity (unfiltered) [m/s]	0.105	0.002	0.065	0.53

All 24 runs were successful with the robot eventually reaching its goal. As shown in Table 1, the *Nudge* planner was able to reliably progress to the goal, at a reduced, albeit consistent velocity, despite both voluntary and involuntary blocking of its path by bystanders. In comparison, the cooperative planner in [8] could not safely make progress at crowd densities above 0.8p/m^2. We did not observe discomfort during human-robot interaction, as most crowd participants reacted with surprise and interest to the robot's assertive planning. This is however a qualitative observation.

5.2. Simulation Experiments

In this work, a simulation environment based on [29] was developed that models crowd response to the planning behaviors using simple, probabilistic interaction models. These models are based on observations from our real-world preliminary tests. During these experiments, detections are simulated directly for all agents which are within the robot's RGB-D sensor field-of-view.

Simulation scenarios were generated for the three maps shown in Figure 7. For each map, six scenes were designed, varying the crowd, robot and goal configuration (see Table 2). Thirteen trials were performed for each combination of planner, map and scene, giving a total of 1404 runs. For comparison, we tested a baseline timed elastic band (TEB) planner, which is a widely used dynamic obstacle-aware planner implemented in the ROS navigation stack. We also compare to Collision Avoidance with Deep Reinforcement Learning (CADRL) [23] as a state-of-the-art RL solution for navigating in crowded environments. We used the authors' open source implementation[1] with minimal hand-tuning of parameters. In addition, a single-behavior policy is evaluated for each of the proposed behaviors. Finally, the proposed IAN multi-behavior planning framework is evaluated. All planners were given the same velocity and acceleration limits, and simulated sensor data to operate with.

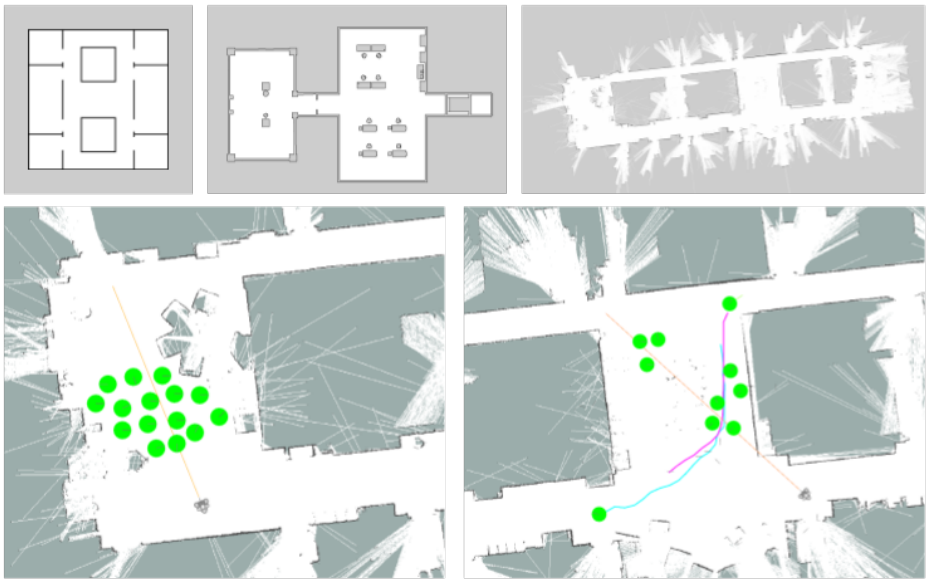

Figure 7. Top: from left to right, simple map, complex map, realistic map used in simulation experiments. Bottom left: realistic map, Scene 4. Bottom right: realistic map, Scene 3.

[1] https://github.com/mfe7/cadrl_ros

Table 2. Scene definitions used in our simulation experiments

Scene	Description
1	Empty map, static obstacles only
2	Mixed with both static and dynamic agents
3	Harder mixed scenario with more agents
4	Dense, static crowd
5	Agents move perpendicularly to the robot (both directions)
6	Agents move parallel to the robot (both directions)

We consider three main metrics, the time-to-goal, the planner success rate, and the time spent in discomfort. The latter is measured as the ratio between the time the robot spent below a certain distance threshold to another agent, and the total time for the trial. The distance thresholds for intimate (d_i = 0.45m) and personal space (d_p = 1.2m) are selected based on the definitions by Rios-Martinez et al. [30]. Each of these metrics are averaged for each scene, and the overall average across all scenes is also reported.

Results of the simulation experiments are shown in Tables 3[2], 4 and 5. Only the IAN planner and *Nudge* are able to find solutions for all scenes. However, it is worth noting that for most scenes, *Intend* achieves the best mean-time-to-goal, yet it performs very poorly in the harder mixed crowd (Scene 3) and fails altogether in the dense static crowd (Scene 4). In those scenes with the highest crowd density, *Nudge* performs best. This result supports the idea that, given knowledge about the modality of a scene, an expert planner for that modality will tend to outperform general-purpose planning approaches. Nevertheless, the IAN planner achieves second-best or equal top scores on all scenes for both time-to-goal and success-ratio metrics, demonstrating that its ability to reason over the crowd state to select the best expert behaviour for the current situation is advantageous for planning across all types of crowds.

Table 3. Per-scene mean time-to-goal in seconds (including standard deviations)

	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Overall
IAN (ours)	*76.11±20.7*	*116.43±47.0*	*123.12±45.62*	*95.45±59.5*	*38.14±15.65*	*91.45±35.1*	*89.68±48.7*
TEB	118.6±32.0	154.33±57.6	-	-	49.29±19.2	142.53±79.6	115.64±67.0
CADRL	91.26±20.6	209.05±31.8	-	-	55.11±21.1	110.29±52.3	97.81±55.0
Intend	**74.54±20.0**	**106.53±35.1**	245.57±35.4	-	**37.78±16.7**	**85.40±36.9**	**79.26±44.4**
Say	99.85±25.0	128.39±36.2	228.35±45.4	-	46.32±17.0	99.39±42.9	96.03±47.3
Nudge	140.81±39.3	148.96±45.2	**97.90±24.0**	**62.81±12.2**	53.72±18.0	100.70±20.0	100.82±45.9

Table 4. Per-scene mean success rate

	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Overall
IAN (ours)	1.00	1.00	0.95	0.85	1.00	1.00	0.97
TEB	0.92	0.95	0.00	0.00	1.00	1.00	0.65
CADRL	1.00	0.33	0.00	0.00	1.00	0.97	0.55
Intend	1.00	1.00	0.08	0.00	1.00	1.00	0.68
Say	1.00	1.00	0.08	0.00	1.00	1.00	0.68
Nudge	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Furthermore, although *Nudge* has a perfect success rate across all scenes, Table 5 shows that it scores worst overall for the time-spent-in-intimate-space metric. This result highlights the fact that despite a higher success rate, using an assertive-planning-only approach can be a poor choice if aspiring to design socially acceptable robots. To that end, it is in principle simple to add social or discomfort awareness to the IAN planner, by adding a term to the relevant behavior cost function. In doing so, it makes it possible to tune how often the high-level planner selects assertive behaviors.

[2] Note that the overall time-to-goal includes only successful runs, as the time cost of not reaching the goal is undefined.

Table 5. Per-scene ratio of time spent below distance threshold vs. total time for each planner

	Scene 1		Scene 2		Scene 3		Scene 4		Scene 5		Scene 6		Overall	
	t_{d_i}	t_{d_p}	t_{d_i}	t_{d_p}	t_{d_i}	t_{d_p}	t_{d_i}	t_{d_p}	t_{d_i}	t_{d_p}	t_{d_i}	t_{d_p}	t_{d_i}	t_{d_p}
IAN (ours)	0.00	0.00	*0.07*	0.40	**0.14**	*0.68*	**0.22**	**0.53**	**0.04**	0.33	*0.10*	0.43	0.11	0.47
TEB	0.00	0.00	**0.04**	*0.34*	-	-	-	-	**0.04**	**0.27**	0.13	0.45	**0.07**	**0.35**
CADRL	0.00	0.00	0.11	0.40	-	-	-	-	0.12	0.33	0.26	0.49	0.18	0.41
Intend	0.00	0.00	0.10	0.50	0.25	0.82	-	-	0.08	0.34	0.15	0.56	0.11	0.47
Say	0.00	0.00	0.09	0.41	0.31	0.79	-	-	0.06	0.33	**0.08**	*0.42*	*0.08*	*0.40*
Nudge	0.00	0.00	0.08	**0.31**	*0.18*	**0.47**	*0.40*	*0.77*	*0.05*	*0.31*	0.12	**0.38**	0.17	0.45

5.3. Real World Planner Validation

As proof-of-concept, we ran the entire planning framework on a humanoid robot platform in real scenarios. These scenarios included several encounters with pedestrians, some purposefully blocking the robot's path. No special modifications were made to the test environments, any objects and furniture in the scene were as found.

The first trial was performed in an office environment (same office floor from which the realistic simulation map was drawn). Twelve runs were executed for a total runtime of around 12 minutes. The planner was tested in both prompted and natural interactions with participants and passers-by, with the robot successfully reaching its goal in all runs. The *Intend* behavior was executed 24 times, of which 18 resulted in successful behavior outcomes. For *Say*, these counts were 12 and 6, while *Nudge* was successfully used 9 times out of 9. The average progress-to-goal velocity was 0.13m/s. For comparison, in the simulation experiments, the same metric for the IAN planner in a similar environment was measured as 0.14m/s (Scene 3, realistic map).

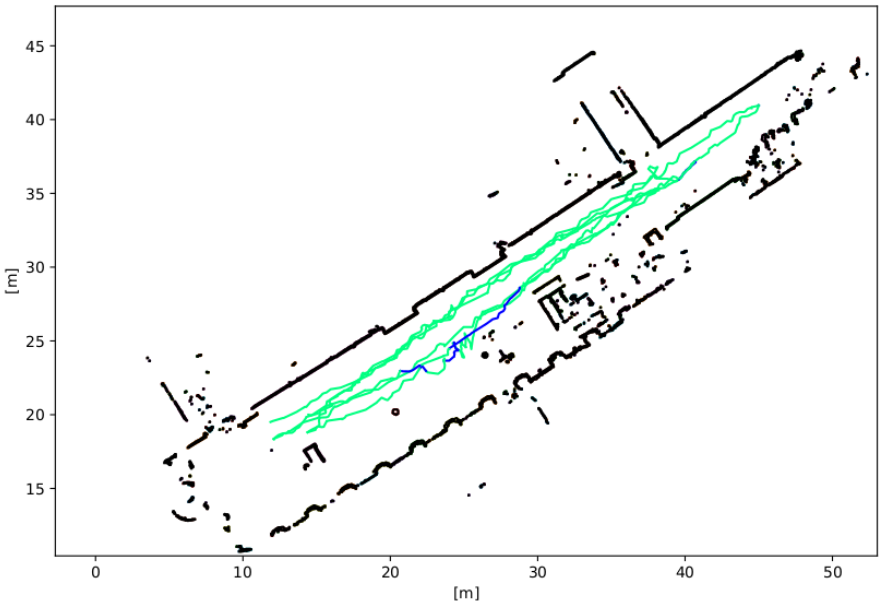

Figure 8. Map and executed behavior trajectories during one of the real-life validation experiments of the IAN planning approach.

The second trial was performed in a university hall. The map and executed behavior trajectories for this trial are shown in Figure 8. Most interactions were with unaware pedestrians, moving longitudinally to the robot trajectory. Six runs were executed for a total runtime of over 14 minutes. Again, the robot reached its goal successfully in all runs. Most pedestrians stopped for only a short time to interact with the robot or not at all. As a result of this, the high-level planner only selected *Intend* and *Say* behaviors. The *Intend* behavior was executed 33 times, of which 29 resulted in successful behavior outcomes. *Say* was executed 6 times with 5 successful behavior outcomes. The average progress-towards-goal velocity was 0.25m/s. For comparison, we note that in a similar simulation scenario (Scene 6), the average-progress-towards-goal velocities were 0.26m/s, 0.24m/s, and 0.18m/s for the simple map, the complex map, and the real map, respectively.

The entire pipeline, including person detection and tracking, LiDAR-based SLAM, high-level planning, and controllers was able to run on an onboard laptop in real-time (Intel core i7-4810MQ CPU, 8 threads, 3.8Ghz, and Quadro K2100M 2GB GPU used for person detection). Each high-level planning loop was measured to take on average 0.54s with 1000 MCTS trials (non-parallelized).

6. Conclusion

We proposed the IAN planner for navigating robots through human crowds using multi-modal behaviors. Our results showed that the planner, which selects behaviors to execute based on the observed state of the crowd, was able to successfully navigate in natural human crowds in excess of 1p/m^2. In particular, we demonstrated the need for assertive planning behaviors such as *Nudge* in cases of high crowd density where classic planners such as TEB and learned controllers such as CADRL are unable to find solutions to navigate through the crowd.

In this work, we introduced three behavior modes *Intend*, *Say* and *Nudge*. However, there is certainly potential to explore additional behaviors that may target specific crowd behaviors not explored here. For example, following another pedestrian in a dense flowing crowd could be a viable navigation strategy. In this way, the multi-behavior planning approach can be considered modular since it is simple to introspect on the usefulness of individual behaviors in different crowd scenarios.

Finally, many of the parameters associated with the individual behaviors are currently hand-tuned. In future, data driven methods can be used to learn the behavior transition model parameters online or to directly learn a feature representation of the crowd state. Nevertheless, the costs of applying certain behaviors, in metrics such as social acceptance, comfort, and reactions of nearby persons are still poorly understood and necessitate additional research into human-crowd interactions.

Appendix A: Behavior Transition Model Parameters

A.1. Behavior Outcome Probabilities

Each of the local state features of a cell i represent the value of that feature with respect to a reference path length (in our case 10m). This implementation allows us to avoid practical scaling issues related to the resolution of the underlying discretized grid map. That is, the probability of a successful outcome over a query subpath ξ is simply computed as the worst-case probability over all cells $i \in \xi$ rescaled according to the length of the query subpath, $length(\xi)$.

For the reference subpath ($length(\xi_{ref}) = 10$), the probability of a successful outcome after executing the *Intend* behavior given the local state features s_i is computed as,

$$p_{ref}(o|s_i, b_{Intend}) = \frac{permissivity_i \cdot perceptivity_i}{2 \cdot crowdedness_i}. \quad (7)$$

For the *Say* behavior,

$$p_{ref}(o|s_i, b_{Say}) = \frac{3 \cdot permissivity_i}{2 \cdot crowdedness_i}. \quad (8)$$

Note that the transition function for executing the *Say* behavior increases the perceptivity to the maximum value, thus $perceptivity_i = 1$ and so is excluded from (8). Similarly, the *Nudge* behavior is designed as an assertive action and so neither the perceptivity nor the crowdedness of the state will influence the success or failure of the action, which is dependent only on the permissivity,

$$p_{ref}(o|s_i, b_{Nudge}) = 5 \cdot permissivity_i. \quad (9)$$

Equations (7), (8) and (9) are clipped to [005, 0.99] as a heuristic for reducing model overconfidence and avoiding numerical issues when propagating the probabilities.

Rescaling to the query subpath length then gives,

$$p(o|s_i, b) = p_{ref}(o|s_i, b)^{\frac{length(\xi)}{length(\xi_{ref})}}. \quad (10)$$

A.2. Cost Functions

The local cost functions are designed to reflect the expected time to execute each behavior across a state transition from (x_i, y_i) to (x_i', y_i'). The local cost of executing the *Intend* behavior is computed as,

$$c(b_{Intend}) = \frac{\sqrt{(x_i - x_i')^2 + (y_i - y_i')^2}}{v} \cdot \varepsilon, \tag{11}$$

where the average velocity of the robot is given as $v = 0.3$m/s and ε is a noise term drawn from $\mathcal{U}(0.9,0.2)$.

The *Say* behavior uses a similar base motion planner but at a reduced speed and also includes time to execute the utterance. Thus, the cost is computed as,

$$c(b_{Say}) = 1.5 \cdot c(b_{Intend}). \tag{12}$$

Similarly, the cost for executing *Nudge* (slowest travel time) is computed as

$$c(b_{Nudge}) = 2 \cdot c(b_{Intend}). \tag{13}$$

A.3. Deterministic Transition Functions

The deterministic transition functions $T(S, b, \xi)$ used in IAN are shown in Table 6. Note that several of the behavior-outcome pairs do not result in a change in the belief of the local crowd state. For example, crowdedness does not change due to any behavior of the robot. Similarly, the success of the *Nudge* behavior does not inform us about any of the local crowd state variables, whereas a failure to progress along a path when executing *Nudge* decreases the belief of the permissivity of the local crowd state.

These transition functions were designed based on heuristic observations of robot-crowd interactions when executing each of the behaviors, *Intend*, *Say*, and *Nudge*. During online planner execution, additional observations of the crowd response can be observed. The potential to use these online observations to update the transition function parameters in real-time is a promising avenue for future investigation.

Table 6. Deterministic transition functions for each of the local crowd state variable given a success or failure outcome of executing a behavior.

		Success	Failure
Intend	Crowdedness	$s' = s$	$s' = s$
	Perceptivity	$s' = s$	$s' = 0.5s$
	Permissivity	$s' = 1 - 0.9(1 - s)$	$s' = 0.1s$
Say	Crowdedness	$s' = s$	$s' = s$
	Perceptivity	$s' = s$	$s' = 1$
	Permissivity	$s' = 1 - 0.9(1 - s)$	$s' = 0.1s$
Nudge	Crowdedness	$s' = s$	$s' = s$
	Perceptivity	$s' = s$	$s' = s$
	Permissivity	$s' = s$	$s' = 0.1s$

References

[1] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," The Int. J. of Robot. Res., vol. 17, no. 7, pp. 760–772, 1998.
[2] Macaluso, E. Ardizzone, A. Chella, M. Cossentino, A. Gentile, R. Gradino, I. Infantino, M. Liotta, R. Rizzo, and G. Scardino, "Experiences with CiceRobot, a museum guide cognitive robot," in Congr. of the Italian Assoc. for Artif. Intell., 2005, pp. 474–482.
[3] R. Nourbakhsh, C. Kunz, and T. Willeke, "The mobot museum robot installations: A five year experiment," in Proc. 2003 IEEE/RSJ Int. Conf. on Intell. Robots and Syst., 2003, pp. 3636–3641.
[4] A. Clodic, S. Fleury, R. Alami, R. Chatila, G. Bailly, L. Brethes, M. Cottret, P. Danes, X. Dollat, F. Elisei et al., "Rackham: An interactive robot-guide," in ROMAN 2006-the 15th IEEE Int. Symp. on Robot and Human Interactive Commun., 2006, pp. 502–509.
[5] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Probabilistic algorithms and the interactive museum tour-guide robot Minerva," The Int. J. of Robot. Res., vol. 19, no. 11, pp. 972–999, 2000.
[6] W. Burgard, A. B. Cremers, D. Fox, D. Ha ̈hnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," Artif. Intell., vol. 114, no. 1-2, pp. 3–55, 1999.
[7] K. O. Arras, N. Tomatis, and R. Siegwart, "Robox, a remarkable mobile robot for the real world," in Exp. Robot. VIII, 2003, pp. 178– 187.
[8] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," The Int. J. of Robot. Res., vol. 34, no. 3, pp. 335–356, 2015.
[9] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," IEEE Robot. & Automat. Mag., vol. 4, no. 1, pp. 23–33, 1997.
[10] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in Distrib. Autonom. Robotic Syst., 2013, no. 83, pp. 203–216.
[11] A. Rudenko, L. Palmieri, and K. O. Arras, "Joint long-term prediction of human motion using a planning-based social force approach," in 2018 IEEE Int. Conf. on Robot. and Automat., 2018, pp. 4571–4577.
[12] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," Physical Rev. E, vol. 51, no. 5, pp. 4282–4286, 1995.
[13] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in 2018 IEEE Int. Conf. on Robot. and Automat., 2018, pp. 5921–5928.
[14] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in 2009 IEEE 12th Int. Conf. on Comput. Vision, 2009, pp. 261–268.
[15] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in IEEE/RSJ Int. Conf. on Intell. Robots and Syst., 2010, pp. 797–803.
[16] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in Proc. of the 21st Int. Conf. on Mach. Learn., 2004, pp. 1–8.
[17] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," The Int. J. of Robot. Res., vol. 35, no. 11, pp. 1289–1307, 2016.
[18] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R.Siegwart, "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models," in IEEE/RSJ Int. Conf. on Intell. Robots and Syst., 2016, pp. 2096–2101.
[19] T. Fan, X. Cheng, J. Pan, D. Manocha, and R. Yang, "CrowdMove: Autonomous mapless navigation in crowded scenarios," in IROS Workshop on From freezing to jostling robots: Current challenges and new paradigms for safe robot navigation in dense crowds, 2018.
[20] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in 2017 IEEE Int. Conf. on Robot. and Automat., 2017, pp. 285–292.
[21] R. Kirby, R. Simmons, and J. Forlizzi, "COMPANION: A constraint-optimizing method for person-acceptable navigation," in 18th IEEE Int. Symp. on Robot and Human Interactive Commun., 2009, pp. 607– 612.
[22] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in IEEE/RSJ Int. Conf. on Intell. Robots and Syst., 2017, pp. 1343–1350.
[23] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in IEEE/RSJ Int. Conf. on Intell. Robots and Syst., 2018, pp. 3052– 3059.
[24] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in Int. Conf. on Robot. and Automat., 2019, pp. 6015– 6022.
[25] T. Fan, X. Cheng, J. Pan, P. Long, W. Liu, R. Yang, and D. Manocha, "Getting robots unfrozen and unlost in dense pedestrian crowds," IEEE Robot. and Automat. Lett., vol. 4, no. 2, pp. 1178–1185, 2019.
[26] K. Rana, B. Talbot, M. Milford, and N. Sünderhauf, "Residual reactive navigation: Combining classical and learned navigation strategies for deployment in unknown environments," arXiv preprint arXiv:1909.10972, 2019.

[27] S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," arXiv preprint arXiv:2001.06627, 2020.
[28] Z. Chen, C. Song, Y. Yang, B. Zhao, Y. Hu, S. Liu, and J. Zhang, "Robot navigation based on human trajectory prediction and multiple travel modes," Appl. Sci., vol. 8, no. 11, pp. 1–21, 2018.
[29] F. Grzeskowiak, M. Babel, J. Bruneau, and J. Pettré, "Toward virtual reality-based evaluation of robot navigation among people," in IEEE VR 2020-27th IEEE Conf. on Virtual Reality and 3D User Interfaces, 2020, pp. 1–9.
[30] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "From proxemics theory to socially-aware navigation: A survey," Int. J. of Social Robot., vol. 7, pp. 137–153, 04 2014.