CAAI Transactions on Intelligence Technology

*Research Article*

# Role playing learning for socially concomitant mobile robot navigation

ISSN 2468-2322
Received on 2nd January 2018
Revised on 20th January 2018
Accepted on 9th February 2018
doi: 10.1049/trit.2018.0008
www.ietdl.org

*Mingming Li ✉, Rui Jiang, Shuzhi Sam Ge, Tong Heng Lee*

*Department of Electrical and Computer Engineering, and the Social Robotics Lab, Smart System Institute (SSI), National University of Singapore, Singapore 117576, Singapore*
✉ *E-mail: li_mingming@u.nus.edu*

**Abstract:** In this study, the authors present the role playing learning scheme for a mobile robot to navigate socially with its human companion in populated environments. Neural networks (NNs) are constructed to parameterise a stochastic policy that directly maps sensory data collected by the robot to its velocity outputs, while respecting a set of social norms. An efficient simulative learning environment is built with maps and pedestrians trajectories collected from a number of real-world crowd data sets. In each learning iteration, a robot equipped with the NN policy is created virtually in the learning environment to play itself as a companied pedestrian and navigate towards a goal in a socially concomitant manner. Thus, this process is called role playing learning, which is formulated under a reinforcement learning framework. The NN policy is optimised end-to-end using trust region policy optimisation, with consideration of the imperfectness of robot's sensor measurements. Simulative and experimental results are provided to demonstrate the efficacy and superiority of the proposed method.

## 1 Introduction

The capability to navigate in densely populated and dynamic environments is one of the most important features that enable the deployment of mobile robots in unstructured environment, such as schools, shopping malls and transportation hubs. The key difference between the problem of navigating among humans and the traditional path planning and obstacle avoidance problems is that humans tend to smoothly evade each other interactively and cooperatively, rather than remaining static or maintaining an indifferent trajectory dynamics. In other words, there are social norms that need to be understood and complied to achieve maximum comfort of all involved pedestrians during navigation. We refer to this as the problem of social navigation, which aims to model such social norms and develop a robotic navigation policy that is socially acceptable to the pedestrians around.

For social navigation, the traditional approaches based on dynamic window approach (DWA) [1] or potential fields [2, 3] are usually of limited efficacy as pedestrians are simply regarded as uncooperative obstacles. An illustrative example is the freezing robot problem [4, 5], where a mobile robot will be stuck in a narrow corridor when facing a crowd of people if it lacks the ability to predict the joint collision avoidance behaviours of human pedestrians. To this end, researches have been done to understand the principles of humans' joint collision avoidance strategies and one of the pioneering works are the social force model (SFM) [6, 7]. Other joint collision avoidance model such as reciprocal velocity obstacles (RVOs) have been proposed in [8–10], with an underlying assumption that all involved agents adopt the same collision avoidance strategies. These ideas are also applied to visual tracking of pedestrians [11, 12]. More recently, several attempts are made to learn probabilistic models of pedestrians' trajectories during joint collision avoidance, based on which the robot's navigation decision is generated such that it is able to behave naturally and correctly in similar situations [5, 13–15].

In this paper, we propose to augment the dimensions of human–robot interaction in social navigation by further endowing robot with appropriate group behaviours when it is travelling with a human companion. This capability is highly desirable for assistive mobile robots [16–18], which serve as assistants and companions and are expected to travel along with their human partners in not only home environment but also possibly crowded public areas. In other words, apart from understanding the collision avoidance behaviours of pedestrians, the robot also needs to consider the motion of its companion so as to maintain a sense of affinity when they are travelling together towards a certain goal. We call this socially concomitant navigation (SCN) and it is more challenging than the aforementioned social navigation problem, where the robot is assumed to travel alone with a simpler pursuit of reaching a specific goal while being free of collision.

To address the problem of SCN, we develop a new learning scheme called role playing learning (RPL). Particularly, we formulate such problem under the framework of partially observable Markov decision process (POMDP) and reinforcement learning (RL). A neural network (NN) is used to parameterise the navigation policy of the robot, which is optimised to give proper steering commands for the next time instance based on the robot's current and previous observations to its surroundings. To facilitate the RL process, we create a simulative navigation environment by mirroring collections of real-world pedestrians data sets and develop an on-policy optimisation method called partially observable trust region policy optimisation (PO-TRPO). In each run in an optimisation iteration, the robot will attempt to play itself as a companion of a randomly chosen pedestrian by executing the NN navigation policy. The NN policy is then optimised using PO-TRPO based on a batch of collected trajectories. Compared to the existing analytically derived or data-driven approaches, our RPL scheme has the following advantages:

(i) RPL scheme is less restrictive. It does not rely on the assumption that the robot and other agents (pedestrians) share the same decision-making models [5, 8–10, 14] or that the navigation goals of pedestrians are known [5, 14].
(ii) The formulation of RPL scheme is more generalisable and flexible. Our formulation contains no manually-defined feature and domain knowledge (e.g. statistics of pedestrians' behaviours). It is not hardware specific and can be easily modified to incorporate kinematics of different mobile robot platforms, sensor specifications and navigation objectives. In addition, unlike

[14, 15], the learned navigation policy operates without access to the global map of the environment. Therefore, it is not environment specific and is well generalisable to unmet real-world scenarios.
(iii) We explicitly consider the noise and limitation of the robot's sensor measurements. Most approaches for social navigation assume that the robot has full and accurate knowledge of interested variables, such as positions or distance of pedestrians and obstacles [8–10, 14]. On the contrary, our RPL scheme is rooted from the situation where the robot can only perceive those lie within its sensor's field of view (FoV), with the existence of measurement noise.
(iv) As a RL-based approach, RPL is efficient. Although RPL aims at solving tasks that involve interaction among robot, humans and physical environment, it does not require participation of human in both data collection and learning, which is known to be tedious and time consuming. Instead, the learning process is safely automated in a simulative yet realistic environment with no human intervention.

We evaluate the performance of our approach in both simulations and real-world experiments, by comparing it with a baseline planner based on RVOs [8] and humans, respectively. We also show that, with some tricks, the learned navigation policy can still be effective when the navigation scenario is reduced to the aforementioned social navigation, which means the robot is travelling without human companion.

The rest of this paper is organised as follows. Related work is first reviewed in Section 2. In Section 3, the problem of SCN is formulated as a POMDP and associated definitions are given. RPL scheme and PO-TRPO algorithm are described in Section 4. Sections 5 and 6 provide extensive results of simulation and experiment, followed by some concluding remarks in Section 7.

## 2 Related work

The problems of robot navigation in populated and dynamic environment can be addressed from a number of angles, which can be largely classified into two groups as in the following subsections.

### 2.1 *Interactive behaviours models*

Many researches have been proposed to describe the interactive navigation behaviours of humans by fitting a computational model to the observed pedestrians trajectories [19]. In this way, the robot's path planner is able to understand pedestrians' intention during joint collision avoidance and actively calculate an optimal route towards its goal.

In the field of robotics, a majority of work in this direction is done via inverse RL (IRL) [20], which learns a cost function that explains the observed behaviours. For example, maximum entropy IRL [21] is adopted in a number of works [22–26] for discrete human behaviour prediction and route planning. However, discrete representation is less desirable when modelling trajectories, which are in nature continuous and has higher order dynamics, such as velocities and acceleration. Instead, Kim and Pineau [15] adopt maximum-a-posteriori Bayesian IRL [27] to learn appropriate navigation behaviour of a specific mobile robot from a set of demonstration trajectories. Note that, the demonstration data in [15] is specific to configurations of the robot and its sensor and has to be collected via human operation, which could be time consuming. On the other hand, the authors in [13, 14] learn probabilistic models of composite trajectories of pedestrians from video data by maximum entropy learning and IRL. To better capture the characteristics of observed trajectories, they propose to develop their models based on a set of features that are hand-crafted according to the domain knowledge from psychological studies. In addition, those features contain velocities and accelerations of pedestrians, which, in practice, are hard to precisely measure. Besides, interacting Gaussian process (IGP) is derived in [5] to model the joint trajectories of pedestrian while explicitly considering the effects of observation noise. Nevertheless, the design of IGP also requires several hand-crafted kernels that are formulated based on the priori information in a specific application scenario.

Other than researchers in robotics, the community of computer vision also possess great interest in pedestrian modelling. One of the important topics is trajectory prediction in video space. In [11], linear trajectory avoidance is developed as a dynamic model for pedestrians in video space for short-term trajectory prediction and it is integrated into visual tracking system. Gaussian process is adopted in [28] to learn the motion pattern of pedestrians. Recently, social long short term memory (LSTM) is proposed in [29] for human trajectory prediction in crowd space. Similarly, the feature of social sensitivity is developed in [30] to analyse trajectories of pedestrians and bicyclists. While the above methods can effectively predict the navigation intention of pedestrians in videos, it is still unclear how to apply these models to navigation of robot in real scenarios.

### 2.2 *Steering models*

In contrast to learning behaviour models of pedestrians, a more direct perspective is to develop a steering model that outputs the immediate navigation actions given the robot's current observation to the environment. One of the pioneering work in this direction is the SFM [6], which uses energy/potential functions to encode the social status of pedestrian. Then, the navigation motivation of a pedestrian can be derived by taking the gradients of these energy functions. Following this idea, subsequent works [12, 31–33] propose to infer the optimal parameters of the energy function by fitting them to video data. However, they are likely to produce suboptimal results if the demonstration data from humans are imperfect. In [34], the authors integrate a people tracker and an iterative $A^*$ planner, with which the robot actively follows the pedestrian travelling in a similar direction to navigate through crowded environment. Mehta *et al.* [35] follow the same idea and formulate the choice of a pedestrian to follow as a multi-policy decision making process. On the other hand, Foka and Trahanias [36] develop a hierarchical POMDP for predictive navigation in dynamic environment. The idea is to predict the motion of pedestrians and generate a environment-specific cost map for path planning and obstacle avoidance.

Other than navigating in a pedestrian-aware manner, several reactive collision avoidance techniques have also been developed, such as DWA [1, 37], velocity obstacles [38] and RVOs [8–10]. The common idea of these methods is to treat pedestrians as moving obstacles and reactively update the planner every short periods to achieve collision avoidance. As mentioned in Section 1, these methods are less effective for social navigation as they lack predictive abilities and are based on some restrictive assumptions, such as accurate knowledge of moving agents' velocities [37] and that all agents adopt the identical collision avoidance strategy [8–10].

Our proposed navigation policy belongs to the steering models. It takes an observation vector as input and outputs the navigation action through a stochastic NNs. During RPL, our policy is optimised by the PO-TRPO algorithm, which is derived based on the recent advances in deep RL (DRL) [39, 40]. DRL exploits the massive representation power of deep NNs (DNNs) [41] to build a complex yet sophisticated decision model, with which an agent can directly learn from raw signals instead of carefully crafted feature and tends to act more intelligently. Recently, there are several attempts in using DNN and DRL for robot navigation. For example, an end-to-end motion planner is learned in [42] to map raw sensor data of a laser range finder onto steering commands of a mobile robot. In [43], a decentralised multi-agent collision avoidance policy is learned via DRL, which can be thought as a DRL version of the original RVO approach [8]. Finally, a target-driven visual navigation policy for home environment is learned in [44] via DRL. They create a set of three-dimensional (3D) virtual home environments for effective and efficient training of the agent, which shares a similar idea with our proposed RPL scheme.

## 3 Problem formulation

To formulate the problem of SCN, we give the following rules of SCN:

(i) The robot should reach its goal as fast as possible;
(ii) The robot should not collide with any of the pedestrians or its companion, or run into any obstacle;
(iii) The robot should not run too far away from its companion.

The above rules serve as a generic description of the robot's desired performance during navigation. To give concrete definitions, consider the navigation process as an infinite-horizon discounted POMDP in discrete time, defined by the tuple $(\mathcal{S}, \mathcal{A}, F, \mathcal{O}, p_0, r, \gamma)$. $\mathcal{S}$ is a finite set of states $s$ reflecting the navigation status of the robot. $\mathcal{A}$ is a finite set of actions $a$. In this paper, it is defined as a twosome of the translational and rotational velocities of a synchro-drive mobile robot, i.e. $\boldsymbol{a} = [v_\mathrm{T}, v_\mathrm{R}]$. $F{:}\mathcal{S}\times\mathcal{A}\to\mathcal{S}$ is state-transition mapping, which is characterised by the dynamics of the robot, the other humans and the environment. Without loss of generality, we assume deterministic state transition, i.e. $\boldsymbol{s}_{i+1} = F(\boldsymbol{s}_i, \boldsymbol{a}_i)$, where $\boldsymbol{s}_i$, $\boldsymbol{a}_i$ are the state and action taken at time $t_i$. $\mathcal{O}$ is the set of the robot's observation $o$ to the state $\boldsymbol{s}$ and $\beta(\boldsymbol{o}|\boldsymbol{s})$ denotes the conditional observation probability distribution. Note that, in practice, the robot's observation has only incomplete access to $\boldsymbol{s}$ or is subject to certain measurement noise, which implies $\boldsymbol{o} \neq \boldsymbol{s}$. $p_0{:}\mathcal{S}\to\mathbb{R}$ is the initial state distribution, i.e. $\boldsymbol{s}_0 \sim p_0$. $r{:}\mathcal{S}\to\mathbb{R}$ is a scalar reward given to the robot and $\gamma \in (0, 1]$ is the reward discount factor.

*Robot motion dynamics:* In this paper, synchro-drive mobile robots are considered, whose motion equation can be approximated by assuming the robot's velocities to be constant within a certain short time period $[t_i, t_{i+1}]$ [1] with length $\Delta t = t_{i+1} - t_i$. Particularly, let $\phi_r(t_i)$ and $\rho_r(t_i) = [x_r(t_i), y_r(t_i)]$ denote the robot's heading and its positions in a 2D Cartesian space at time $t_i$, respectively. $v_\mathrm{T}(t_i) \in [0, \bar{v}_\mathrm{T}]$ and $v_\mathrm{R}(t_i) \in [-\bar{v}_\mathrm{R}, \bar{v}_\mathrm{R}]$ represent the robot's translational and rotational velocities. Define $\Delta x_r = x_r(t_{i+1}) - x_r(t_i)$ and $\Delta y_r = y_r(t_{i+1}) - y_r(t_i)$. When the robot has non-zero rotational velocity, i.e. $v_\mathrm{R}(t_i) \neq 0$, we have

$$\Delta x_r = -\frac{v_\mathrm{T}(t_i)(\sin\phi_r(t_i) - \sin(\phi_r(t_i) + v_\mathrm{R}(t_i)\Delta t))}{v_\mathrm{R}(t_i)} \tag{1}$$

$$\Delta y_r = \frac{v_\mathrm{T}(t_i)(\cos\phi_r(t_i) - \cos(\phi_r(t_i) + v_\mathrm{R}(t_i)\Delta t))}{v_\mathrm{R}(t_i)} \tag{2}$$

Otherwise, when $v_\mathrm{R}(t_i) = 0$

$$\Delta x_r = v_\mathrm{T}(t_i)\cos\phi_r(t_i) \tag{3}$$

$$\Delta y_r = v_\mathrm{T}(t_i)\sin\phi_r(t_i) \tag{4}$$

With the above formulations, our goal is optimising a stochastic navigation policy $P_{\boldsymbol{\theta}}{:}\mathcal{O}\times\mathcal{A}\to[0,1]$ with parameters $\boldsymbol{\theta}$ in order to maximise the expected discounted reward

$$\eta(P_{\boldsymbol{\theta}}) = \mathbb{E}_\tau\left[\sum_{i=0}^{\infty}\gamma^i r(\boldsymbol{s}_i, \boldsymbol{a}_i)\right] \tag{5}$$

where $\tau = (\boldsymbol{s}_0, \boldsymbol{o}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{o}_1, \boldsymbol{a}_1, \ldots)$ denotes the whole trajectory and $\boldsymbol{a}_i \sim P_\theta(\boldsymbol{a}_i|\boldsymbol{o}_i)$. The specific definitions of the above ingredients for SCN will be elaborated as follows.

*State:* Given $\rho_r$ and $\phi_r$, define the distance $d$ and direction $\phi$ of a point $\rho = [x, y]$ to the robot as follows:

$$d(\rho) = \sqrt{(x - x_r)^2 + (y - y_r)^2} \tag{6}$$

$$\phi(\rho) = \arctan\left(\frac{y - y_r}{x - x_r}\right) - \phi_r \tag{7}$$

Then, the robot's distance to the goal located at $\rho_g = [x_g, y_g]$ are computed as $d_g = d(\rho_g)$ and $\phi_g = \phi(\rho_g)$ denotes the offset angle between the robot's current heading $\phi_r$ and its goal. Similarly, we can define the twosomes $(d^j_\mathrm{ped}, \phi^j_\mathrm{ped})$, $(d^j_\mathrm{com}, \phi^j_\mathrm{com})$ or $(d^j_\mathrm{obs}, \phi^j_\mathrm{obs})$ to describe the relative position of a pedestrians $\rho^j_\mathrm{ped}$, a companion $\rho^j_\mathrm{com}$ or an obstacle $\rho^j_\mathrm{obs}$ to the robot. With such definitions, the state $\boldsymbol{s}$ is defined to incorporate the information related to the robot's navigation status as follows:

$$\boldsymbol{s} = [d_g, \phi_g, \boldsymbol{a}, \boldsymbol{p}_\mathrm{ped}, \boldsymbol{p}_\mathrm{com}, \boldsymbol{p}_\mathrm{obs}] \tag{8}$$

where $\boldsymbol{a}$ is the current action vector and

$$\boldsymbol{p}_\mathrm{ped} = [d^1_\mathrm{ped}, \phi^1_\mathrm{ped}, \ldots, d^{n_\mathrm{ped}}_\mathrm{ped}, \phi^{n_\mathrm{ped}}_\mathrm{ped}] \tag{9}$$

$$\boldsymbol{p}_\mathrm{com} = [d_\mathrm{com}, \phi_\mathrm{com}] \tag{10}$$

$$\boldsymbol{p}_\mathrm{obs} = [d^F_\mathrm{obs}, d^{L^-}_\mathrm{obs}, \phi^{L^-}_\mathrm{obs}, d^{R^-}_\mathrm{obs}, \phi^{R^-}_\mathrm{obs}, d^{L^+}_\mathrm{obs}, \phi^{L^+}_\mathrm{obs}, d^{R^+}_\mathrm{obs}, \phi^{R^+}_\mathrm{obs}] \tag{11}$$

The vector $\boldsymbol{p}_\mathrm{ped}$ includes the distances and directions of $n_\mathrm{ped}$ closest pedestrians while $\boldsymbol{p}_\mathrm{com}$ includes those of the robot's companion.

The vector $\boldsymbol{p}_\mathrm{obs}$ is a compact description to the robot's perception of the surrounding environment. Particularly, the boundaries of the occupied space (obstacles) in the environment are represented as a finite point set $\mathbb{Z} = \{\rho^1_\mathrm{obs}, \rho^2_\mathrm{obs}, \ldots, \rho^j_\mathrm{obs}, \ldots\}$. Then, the nine variables in $\boldsymbol{p}_\mathrm{obs}$ are defined based on the following assumption.

*Assumption 1:* An obstacle $\rho_\mathrm{obs} \in \mathbb{Z}$ has no effect on the robot's navigation decision if it satisfies $d(\rho_\mathrm{obs}) > \bar{d}_\mathrm{obs}$, where $\bar{d}_\mathrm{obs}$ is a predefined finite constant.

By Assumption 1, it is sufficient to consider only obstacles in $\mathbb{Z}$ that are closed enough to the robot, whose distances are less than $\bar{d}_\mathrm{obs}$. In practice, this limit may correspond to the robot's perception range. Let

$$\bar{\mathbb{Z}} = \{\rho | \rho \in \mathbb{Z}, d_\rho \le \bar{d}_\mathrm{obs}\} \tag{12}$$

The components in vector $\boldsymbol{p}_\mathrm{obs}$ are described as follows.

The distance to the nearest obstacle located at heading of the robot, i.e.

$$d^F_\mathrm{obs} = \min_{\rho\in\bar{\mathbb{Z}}\ \mathrm{and}\ |\phi(\rho)|\le\epsilon_\rho} d(\rho) \tag{13}$$

where $\epsilon_\rho$ is a small constant.

For $d^L_\mathrm{obs}, \phi^L_\mathrm{obs}, d^R_\mathrm{obs}, \phi^R_\mathrm{obs}$, they represent the distance and direction of the closest and farthest obstacles on the robot's left ($\rho^L_\mathrm{obs}$) and right side ($\rho^R_\mathrm{obs}$), respectively, which are defined mathematically as follows:

$$\rho^{L^-}_\mathrm{obs} = \operatorname*{argmin}_{\rho\in\bar{\mathbb{Z}}\ \mathrm{and}\ \phi(\rho)>\epsilon_\rho} d(\rho) \tag{14}$$

$$\rho^{R^-}_\mathrm{obs} = \operatorname*{argmin}_{\rho\in\bar{\mathbb{Z}}\ \mathrm{and}\ \phi(\rho)<-\epsilon_\rho} d(\rho) \tag{15}$$

$$\rho^{L^+}_\mathrm{obs} = \operatorname*{argmax}_{\rho\in\bar{\mathbb{Z}}\ \mathrm{and}\ \phi(\rho)>\epsilon_\rho} d(\rho) \tag{16}$$

$$\rho^{R^+}_\mathrm{obs} = \operatorname*{argmax}_{\rho\in\bar{\mathbb{Z}}\ \mathrm{and}\ \phi(\rho)<-\epsilon_\rho} d(\rho) \tag{17}$$

Then, the variables in $\boldsymbol{p}_\mathrm{obs}$ can be simply determined as the distance and directions of the above points according to (6) and (7). Fig. 1

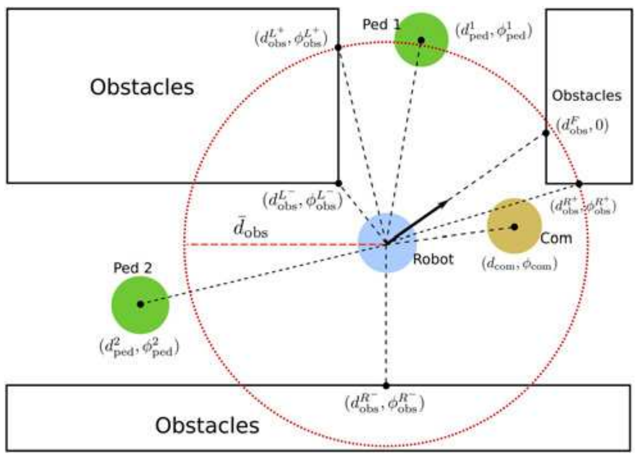

Obstacles
Ped 1
$(d_{\rm obs}^{L^+}, \phi_{\rm obs}^{L^+})$
$(d_{\rm ped}^1, \phi_{\rm ped}^1)$
Obstacles
$(d_{\rm obs}^F, 0)$
$(d_{\rm obs}^{L^-}, \phi_{\rm obs}^{L^-})$
$(d_{\rm obs}^{R^+}, \phi_{\rm obs}^{R^+})$
$\bar{d}_{\rm obs}$
Com
Robot
$(d_{\rm com}, \phi_{\rm com})$
Ped 2
$(d_{\rm ped}^2, \phi_{\rm ped}^2)$
$(d_{\rm obs}^{R^-}, \phi_{\rm obs}^{R^-})$
Obstacles

**Fig. 1** *Illustration of the state variables in (8). The blue, yellow and green circles represent the robot, its companion (Com.) and the pedestrians (Ped.) respectively. The red dashed circle with a radii $\bar{d}_{\rm obs}$ represents the boundary of the set $\mathbb{Z}$ in (12). The black arrow shows the current heading of the robot. Considering the robot's current position as the origin, the polar coordinates of the pedestrians, the companion, the closest (and the farthest) obstacles in each direction are compactly represented as vectors $\boldsymbol{p}_{\rm ped}, \boldsymbol{p}_{\rm com}, \boldsymbol{p}_{\rm obs}$*

provides a comprehensive illustration of the state variables $\boldsymbol{p}_{\rm ped}, \boldsymbol{p}_{\rm com}$ and $\boldsymbol{p}_{\rm obs}$.

*Observation*: As discussed in the previous sections, sensors mounted on the robot are always subject to various kinds of limitation and measurement noise, which must be taken into account in order to develop a robust and practical navigation system. To this end, we define $\boldsymbol{o}$ as the robot's observation to the true state $\boldsymbol{s}$ as follows:

$$o = [d_g, \phi_g, \boldsymbol{a}, \hat{\boldsymbol{p}}_{\rm ped}, \hat{\boldsymbol{p}}_{\rm com}, \hat{\boldsymbol{p}}_{\rm obs}] \tag{18}$$

By (18), we assume that the robot has accurate information about the goal position and its current taken action (i.e. the velocity commands output to the robot's motor) while its observations to $\boldsymbol{p}_{\rm ped}, \boldsymbol{p}_{\rm com}, \boldsymbol{p}_{\rm obs}$ may be imperfect. Particularly, consider the field of views (FoVs) for the robot's pedestrian and obstacle detectors illustrated as in Fig. 2.

Mathematically, let finite point sets $\mathbb{F}_{\rm ped}$ and $\mathbb{F}_{\rm obs}$ denote the current FoVs of pedestrian and obstacle detectors, characterised by threesomes $(\phi_{\rm ped}^+, \phi_{\rm ped}^-, d_{\rm ped}^+)$ and $(\phi_{\rm obs}^+, \phi_{\rm obs}^-, d_{\rm obs}^+)$, respectively. The robot's observations to the pedestrians' relative positions are

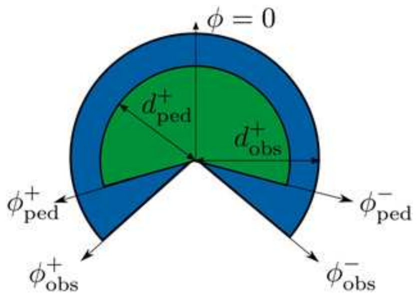


**Fig. 2** *FoVs of the pedestrians (green) and obstacles (blue) detectors. The arrow ($\phi = 0$) points towards the current heading of the robot. The constants $\phi_{\rm ped}^+, \phi_{\rm obs}^+$ and $\phi_{\rm ped}^-, \phi_{\rm obs}^-$ denote the maximum and minimum offset angles in the corresponding FoVs. Finally, $d_{\rm ped}^+$ and $d_{\rm obs}^+$ represent the maximum detection ranges for the pedestrian and obstacle detectors, respectively. The values of these constants should be determined according to the specific configurations of the robot's sensor and the corresponding detection algorithms. Any pedestrian/obstacle outside the FoVs is not observable and therefore will be omitted*

obtained as

$$\hat{\boldsymbol{p}}_{\rm ped} = [\hat{d}_{\rm ped}^1, \hat{\phi}_{\rm ped}^1, \ldots, \hat{d}_{\rm ped}^{n_{\rm ped}}, \hat{\phi}_{\rm ped}^{n_{\rm ped}}] \tag{19}$$

where

$$\hat{d}_{\rm ped}^j = \begin{cases} d_{\rm ped}^j + \tilde{d}_{\rm ped}, & \text{if } \rho_{\rm ped}^j \in \mathbb{F}_{\rm ped} \\ d_{\rm ped}^+, & \text{else} \end{cases} \tag{20}$$

and

$$\hat{\phi}_{\rm ped}^j = \begin{cases} \phi_{\rm ped}^j, & \text{if } \rho_{\rm ped}^j \in \mathbb{F}_{\rm ped} \\ \pi, & \text{if else} \end{cases} \tag{21}$$

for $j = 1, \ldots, n_{\rm ped}$, with $\tilde{d}_{\rm ped}$ being the measurement noise/error.

Similarly, define

$$\hat{\boldsymbol{p}}_{\rm obs} = [\hat{d}_{\rm obs}^F, \hat{d}_{\rm obs}^{L^-}, \hat{\phi}_{\rm obs}^{L^-}, \hat{d}_{\rm obs}^{R^-}, \hat{\phi}_{\rm obs}^{R^-}, \hat{d}_{\rm obs}^{L^+}, \hat{\phi}_{\rm obs}^{L^+}, \hat{d}_{\rm obs}^{R^+}, \hat{\phi}_{\rm obs}^{R^+}] \tag{22}$$

Compared to the states in (13) to (17), only the obstacles within $\mathbb{F}_{\rm obs}$ are observable. Thus, $\hat{d}_{\rm obs}^F$ is formulated as

$$\hat{d}_{\rm obs}^F = \min_{\rho \in \mathbb{Z} \cap \mathbb{F}_{\rm obs} \text{ and } |\phi(\rho)| \le \epsilon_\rho} d(\rho) + \tilde{d}_{\rm obs} \tag{23}$$

where $\tilde{d}_{\rm obs}$ is the measurement noise/error for obstacle detection. The closest observed obstacles on the robot's left and right sides are defined in a similar way as

$$\hat{\rho}_{\rm obs}^{L^-} = \arg\min_{\rho \in \mathbb{Z} \cap \mathbb{F}_{\rm obs} \text{ and } \phi(\rho) > \epsilon_\rho} d(\rho) + \tilde{d}_{\rm obs} \tag{24}$$

$$\hat{\rho}_{\rm obs}^{R^-} = \operatorname{argmin}_{\rho \in \mathbb{Z} \cap \mathbb{F}_{\rm obs} \text{ and } \phi(\rho) < -\epsilon_\rho} d(\rho) + \tilde{d}_{\rm obs} \tag{25}$$

$$\hat{\rho}_{\rm obs}^{L^+} = \operatorname{argmax}_{\rho \in \mathbb{Z} \cap \mathbb{F}_{\rm obs} \text{ and } \phi(\rho) > \epsilon_\rho} d(\rho) + \tilde{d}_{\rm obs} \tag{26}$$

$$\hat{\rho}_{\rm obs}^{R^+} = \operatorname{argmax}_{\rho \in \mathbb{Z} \cap \mathbb{F}_{\rm obs} \text{ and } \phi(\rho) < -\epsilon_\rho} d(\rho) + \tilde{d}_{\rm obs} \tag{27}$$

Then, their distance and directions to the robot are calculated using (6) and (7). For observation to the robot's companions, we rely on the following assumptions.

*Assumptions 2:* The companions $\rho_{\rm com}^1, \ldots, \rho_{\rm com}^{n_{\rm com}}$ are always observable to the robot.

Then, $\hat{\boldsymbol{p}}_{\rm com} = [\hat{d}_{\rm com}, \phi_{\rm com}]$, where

$$\hat{d}_{\rm com} = d_{\rm com} + \tilde{d}_{\rm com} \tag{28}$$

*Remark 1:* By (20) and (23)–(28), it is implied that the observation/measurement noises $\tilde{d}_{\rm ped}, \tilde{d}_{\rm obs}$ and $\tilde{d}_{\rm com}$ are additive and independent in different observations. A typical example of such noise is the additive Gaussian white noise.

*Remark 2:* Our general formulations of states (8) and observations (18) are applicable to various types of onboard sensors, such as range sensors [45, 46], RGB-D [47], time-of-flight [48] and omnidirectional cameras [49], as long as the interested positions can be extracted/estimated from the sensor's raw measurements.

*Remark 3:* The mathematical definitions of the variables in observations $\hat{\boldsymbol{p}}_{\rm ped}, \hat{\boldsymbol{p}}_{\rm com}, \hat{\boldsymbol{p}}_{\rm obs}$ are given for better understanding and are required only in the simulative RPL process. In practice, it is clear that these values can be directly measured via the robot's onboard sensors without accessing the actual 2D Cartesian coordinates $[x, y]$ of the considered point sets (e.g. $\mathbb{Z}$, $\mathbb{F}_{\rm ped}$ and

$\mathbb{F}_{\text{obs}}$). For example, consider a robot equipped with a laser range finder. These distances and offset angles can be easily obtained from the returned ranges array [50].

*Reward function*: A scalar reward will be given to the robot as an award of reaching the goal or a penalty of colliding with obstacles/pedestrians/companions or losing its companions. Particularly, at time $t_i$, the process of SCN will be terminated if any of the following three termination conditions are true.

(i) Goal reaching condition

$$d_g(t_i) \leq 0.8 \tag{29}$$

(ii) Collision conditions

$$\min_j d^j_{\text{ped}}(t_i) \leq 0.4 \tag{30}$$

$$d_{\text{com}}(t_i) \leq 0.4 \tag{31}$$

$$\min(d^F_{\text{obs}}(t_i), d^{L^-}_{\text{obs}}(t_i), d^{R^-}_{\text{obs}}(t_i)) \leq 0.2 \tag{32}$$

(iii) Stray condition

$$d_{\text{com}}(t_i) \geq 2 \tag{33}$$

Based on the above three terminal conditions, a reward $r$ will be given to the robot as follows:

$$r = \begin{cases} 10000, & \text{if (29)} \\ -10000, & \text{if (30) or (31) or} \\ & \quad \text{((32) or (33)} \\ -10|v_{\text{R}}|, & \text{else} \end{cases} \tag{34}$$

Clearly, a positive reward will be given to the robot if it reaches its goal and it will receive a large negative reward if it collides with anything or be stray from its companion. Otherwise, the robot will receive an intermediate reward $-10|v_{\text{R}}|$, which penalises the robot for its rotational velocity to encourage a smoother trajectory with less turning behaviours.

## 4 Role playing learning

In this section, we described the RPL scheme to learn an effective navigation policy $P_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{o})$ for SCN in an efficient data-driven manner. The core idea is to transform the crowd trajectories data collected from real world into a simulative and dynamic navigation environment, where the robot can play itself as a virtual pedestrian and iteratively improve the performance of $P_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{o})$ via PO-TRPO.

Consider a set of simulative navigation environment $\mathbb{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_j, \ldots\}$. Each environment $\mathcal{E}_j = (\mathbb{T}_j, \mathcal{M}_j)$ contains a set of pedestrian trajectories $\mathbb{T}_j = \{\rho^k_{0:T_k}\}$ and a binary map $\mathcal{M}_j$ that annotates the 2D Cartesian coordinates of obstacles/occupied space in the environment. With $\mathbb{E}$, the abstract process of RPL is described by the following pseudo-codes in Algorithm 1 (see Fig. 3).

*Companion synthesisation in non-SCN mode*: As described in Algorithm 1 (Fig. 3), RPL actually incorporates two different navigation scenarios: the SCN proposed in this paper and the traditional social navigation scenario, where the robot has no human companion. This helps develop a navigation policy adaptable to both situations, with no restrictive assumption on the existence of companion. Particularly, the companion position vector $\boldsymbol{p}_{\text{com}}$ and its observation $\hat{\boldsymbol{p}}_{\text{com}}$ are synthesised, with $d_{\text{com}} = \hat{d}_{\text{com}} = 0.8$ for every time step while $\phi_{\text{com}} = \phi_g$. It is clear that the synthesised $\boldsymbol{p}_{\text{com}}$ is equivalent to the situation where the companion is travelling non-distractively along the robot with a constant distance and guarantee that termination conditions (31) and (33) are always false.

```
Initialize navigation policy $P_\theta$
for Iter $= 0, 1, \cdots$, MaxIter do
  while Number of collected sample time steps $\leq$ Batch_size do
    Randomly choose an environment $\mathcal{E}_j$ from $\mathbb{E}$ and then a trajectory $\rho^k_{0:T_k}$ from $\mathbb{T}_j$.
    Initialize the robot's position at $\rho^k_0$ and initial velocities $[v_T, v_R] = [0, 0]$. Set $\rho_g = \rho^k_{T_k}$. Choose the robot's heading such that $\phi_g = 0$.
    Choose SCN Mode with probability 0.5
    if SCN Mode then
      Assign $\rho^k_{T_0:T_k}$ as the trajectory of the robot's companion, where $T_0 = \arg\min_{T'} \|\rho^k_{T'} - \rho^k_{T_k}\| \geq 0.6$
    else
      Create a synthesized companion that moves along the robot
    end if
    Assign all other trajectories in $\mathbb{T}_j$ as pedestrians.
    while None of the termination conditions in (29) to (33) is satisfied do
      Update the states and observations of the robot according to Eqs. (8) and (18).
      Let the robot execute its policy $P_\theta$.
      Update the robot's position according to dynamics (1) to (4)
      Calculate the current reward from Eq. (34)
      Update the positions of the companion and pedestrians according to the trajectories in $\mathbb{T}_j$.
    end while
  end while
  Update $P_\theta$ using PO-TRPO.
end for
```

**Fig. 3** *Algorithm 1: role playing learning*

On the other hand, in SCN mode, the companion is assigned with a truncated trajectory $\rho^k_{T_0:T_k}$ such that the initial robot-companion distance is sufficiently large.

In this paper, we construct a deep policy NN to parameterise the navigation policy $P_{\boldsymbol{\theta}}$, whose structure is shown in Fig. 4. The policy network $P_{\boldsymbol{\theta}}$ is to be trained with the trust region policy optimisation (TRPO) [40] method. However, the original TRPO method is derived based on fully observable MDP, which cannot

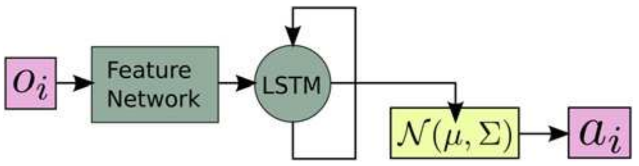


**Fig. 4** *Structure of the deep policy network $P_{\boldsymbol{\theta}}$. At time $t_i$, the observation vector $\boldsymbol{o}_i$ is input to the feature network, which is a feed-forward multi-layer perceptron. The output of the feature network is then fed to a LSTM network [51], a recurrent network for aggregation of the information collected through the navigation process. The LSTM network's outputs are assigned as the mean vector $\boldsymbol{\mu} \in \mathbb{R}^2$ of the diagonal Gaussian unit $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ on the right. The covariance matrix $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I} \in \mathbb{R}^{2\times 2}$, however, is independent of $\boldsymbol{o}_i$ and it is designed to be gradually decreasing during training and fixed during tests and experiments. Finally, the actions $\boldsymbol{a}_i = [v_T, V_R]$ are drawn according to $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$*

be directly applied to our problem due to the imperfect observation in our formulation and practice. Thus, we proposed to extend the original TRPO algorithm as PO-TRPO, which will be described in the following subsections.

## 4.1 Trusted region policy optimisation

The TRPO [40] algorithm is an effective on-policy optimisation method for large non-linear policies and tends to give monotonic improvement during the iterative optimisation process. To be specific, a fully observable MDP is considered by TRPO and therefore the policy to be optimised is formulated as $P^*_{\boldsymbol{\xi}}(\boldsymbol{a}|\boldsymbol{s})$, where $\xi$ is the parameter vector of the policy $P^*$. Note that, $P^*_{\boldsymbol{\xi}}(\boldsymbol{a}|\boldsymbol{s})$ determines the action $a$ directly from the true state $s$, which differs from our observation-based policy $P_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{o})$. Let us consider the following standard definitions of the state-action value function $Q_{\boldsymbol{\xi}}(\boldsymbol{s}_i, \boldsymbol{a}_i)$, the value function $V_{\boldsymbol{\xi}}(\boldsymbol{s}_i)$ and the advantage function $A_{\boldsymbol{\xi}}(\boldsymbol{s}_i, \boldsymbol{a}_i)$:

$$Q_{\boldsymbol{\xi}}(\boldsymbol{s}_i, \boldsymbol{a}_i) = \mathbb{E}_{\boldsymbol{s}_{i+1},\boldsymbol{a}_{i+1},\dots}\left[\sum_{l=0}^{\infty} \gamma^l r(\boldsymbol{s}_{i+l})\right], \tag{35}$$

$$V_{\boldsymbol{\xi}}(\boldsymbol{s}_i) = \mathbb{E}_{\boldsymbol{a}_i,\boldsymbol{s}_{i+1},\dots}\left[\sum_{l=0}^{\infty} \gamma^l r(\boldsymbol{s}_{i+l})\right], \tag{36}$$

$$A_{\boldsymbol{\xi}}(\boldsymbol{s}_i, \boldsymbol{a}_i) = Q_{\boldsymbol{\xi}}(\boldsymbol{s}_i, \boldsymbol{a}_i) - V_{\boldsymbol{\xi}}(\boldsymbol{s}_i) \tag{37}$$

where

$$\boldsymbol{a}_i \sim P^*_{\boldsymbol{\xi}}(\boldsymbol{a}|\boldsymbol{s}), \quad \boldsymbol{s}_{i+1} = F(\boldsymbol{s}_i, \boldsymbol{a}_i) \tag{38}$$

In addition, define $\nu_{\xi}$ as the discounted visitation frequencies

$$\nu_{\boldsymbol{\xi}}(\boldsymbol{s}) = p(\boldsymbol{s}_0 = \boldsymbol{s}) + \gamma p(\boldsymbol{s}_1 = \boldsymbol{s}) + \gamma^2 p(\boldsymbol{s}_2 = \boldsymbol{s}) + \cdots \tag{39}$$

where $\boldsymbol{s}_0 \sim \boldsymbol{p}_0$, $\boldsymbol{a}_i$ and $\boldsymbol{s}_{i\geq 1}$ are generated according to $P^*_{\boldsymbol{\xi}}$ and $F$. Let $\boldsymbol{\xi}^-$ denote the old parameters in last iteration. TRPO proposes to optimise the parameters $\boldsymbol{\xi}$ iteratively regarding the following objective function:

$$\text{maximise} \quad \mathbb{E}_{s\sim\nu_{\xi^-},\boldsymbol{a}\sim q^*}\left[\frac{P^*_{\boldsymbol{\xi}}(\boldsymbol{a}|\boldsymbol{s})}{q(\boldsymbol{a}|\boldsymbol{s})} A_{\boldsymbol{\xi}^-}(\boldsymbol{a}|\boldsymbol{s})\right] \tag{40}$$

$$\text{subject to} \quad \mathbb{E}_{s\sim\nu_{\xi^-}}[D_{\text{KL}}(P^*_{\boldsymbol{\xi}^-}(\cdot|\boldsymbol{s})\|P^*_{\boldsymbol{\xi}}(\cdot|\boldsymbol{s}))] \leq \epsilon \tag{41}$$

where $q^*(\boldsymbol{a}|\boldsymbol{s})$ is the importance sampling distribution and $D_{\text{KL}}(P^*_{\boldsymbol{\xi}^-}\|P^*_{\boldsymbol{\xi}})$ is the Kullback–Leibler divergence between the old and current policies.

## 4.2 Partially observable TRPO

As mentioned, our navigation problem is considered as a POMDP. The policy $P_{\boldsymbol{\theta}}(\boldsymbol{a}_i|\boldsymbol{o}_i)$ depends on the observation $\boldsymbol{o}_i$ instead of the true state. Therefore, we write the objective function (40) and the constraint (41) as

$$\text{maximise}\, \mathbb{E}_{s\sim\nu_{\theta^-},\boldsymbol{a},\boldsymbol{o}\sim q}\left[\frac{\sum_{o}\beta(\boldsymbol{o}|\boldsymbol{s})P_{\theta}(\boldsymbol{a}|\boldsymbol{o})}{q(\boldsymbol{a},\boldsymbol{o}|\boldsymbol{s})} A_{\theta^-}(\boldsymbol{a}|\boldsymbol{s})\right] \tag{42}$$

$$\text{subject to} \quad \mathbb{E}_{s\sim\nu_{\theta^-}}[D_{\text{KL}}(P_{\boldsymbol{\theta}^-}(\cdot|\boldsymbol{o})\|P_{\boldsymbol{\theta}}(\cdot|\boldsymbol{o}))] \leq \epsilon \tag{43}$$

For PO-TRPO, samples are collected by executing the old policy $P_{\boldsymbol{\theta}^-}(\boldsymbol{a}|\boldsymbol{o})$ to generate a set of trajectories, such as $\boldsymbol{s}_0, \boldsymbol{o}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{o}_1, \boldsymbol{a}_1, \dots, \boldsymbol{s}_{T-1}, \boldsymbol{o}_{T-1}, \boldsymbol{a}_{T-1}, \boldsymbol{s}_T$. Therefore

$$q(\boldsymbol{a}_i, \boldsymbol{o}_i|\boldsymbol{s}_i) = \beta(\boldsymbol{o}_i|\boldsymbol{s}_i)P_{\boldsymbol{\theta}^-}(\boldsymbol{a}_i|\boldsymbol{o}_i) \tag{44}$$

where $i = 0, \dots, T-1$.

Next, for a trajectory $s_{0:T}$, we use the generalised advantage estimation (GAE) [39] to construct an empirical estimation $\hat{A}$ of the advantage function $A_{\boldsymbol{\theta}^-}(\boldsymbol{a}_i|\boldsymbol{s}_i)$ as the following:

$$\hat{A}_i = \sum_{l=0}^{T-i} (\gamma\lambda)^l \delta^V_{i+l} \tag{45}$$

where

$$\delta^V_i = r_i + \gamma\hat{V}_{\zeta}(\boldsymbol{s}_{i+1}) - \hat{V}_{\zeta}(\boldsymbol{s}_i) \tag{46}$$

and $\hat{V}_{\zeta}(\boldsymbol{s}_i)$ is the estimation of the value function (36) with parameters $\boldsymbol{\zeta}$ (and $\boldsymbol{\zeta}^-$ being the old parameters). By collecting a set of $K$ trajectories $\{\boldsymbol{s}^k_{0:T_k}, \boldsymbol{o}^k_{0:T_k}, \boldsymbol{a}^k_{0:T_k}\}^K_{k=1}$, $\hat{V}_{\boldsymbol{\zeta}}$ is obtained by solving the following constrained regression problem [39]:

$$\text{minimise} \quad J_{1\zeta} = \sum_{k=1}^{K}\sum_{i=0}^{T_k} \|\hat{V}_{\zeta}(\boldsymbol{s}^k_i) - \sum_{l=0}^{T_k-i} \gamma^l r^k_{i+l}\|^2 \tag{47}$$

$$\text{subject to} \quad \sum_{k=1}^{K}\sum_{i=0}^{T_k} \frac{\|V_{\zeta}(\boldsymbol{s}^k_i) - V_{\zeta^-}(\boldsymbol{s}^k_i)\|}{2J_{1\zeta^-}} \leq \epsilon_1 \tag{48}$$

Finally, as the conditional observation probability distribution $\beta(\boldsymbol{o}|\boldsymbol{s})$ is independent of parameters $\boldsymbol{\theta}$ and time, we obtain an estimation of the objective function (42) and the constraints (43) by replacing the expectations with sample averages as

$$\text{maximise} \quad J_{\boldsymbol{\theta}} = \frac{1}{\sum_{k=1}^{K} T_k}\sum_{k=1}^{K}\sum_{i=0}^{T_k} \frac{P_{\boldsymbol{\theta}}(\boldsymbol{a}^k_i|\boldsymbol{o}^k_i)}{P_{\boldsymbol{\theta}^-}(\boldsymbol{a}^k_i|\boldsymbol{o}^k_i)}\hat{A}^k_i \tag{49}$$

$$\text{subject to} \quad \bar{D}^{\boldsymbol{\theta}^-}_{\boldsymbol{KL}}(P_{\boldsymbol{\theta}^-}, P_{\boldsymbol{\theta}}) \leq \epsilon \tag{50}$$

where

$$\bar{D}^{\boldsymbol{\theta}^-}_{\text{KL}}(P_{\boldsymbol{\theta}^-}, P_{\boldsymbol{\theta}}) = \frac{1}{\sum_{k=1}^{K} T_k}\sum_{k=1}^{K}\sum_{i=0}^{T_k} D_{\text{KL}}(P_{\boldsymbol{\theta}^-}(\cdot|\boldsymbol{o}^k_i)\|P_{\boldsymbol{\theta}}(\cdot|\boldsymbol{o}^k_i))] \tag{51}$$

which has the same form as the one obtained in [39], except that the policy $P_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{o})$ is conditioned on observation $o$ instead.

Finally, the constrained optimisation problem described in (49) and (50) is solved by conjugate gradient algorithm [52]. To summarise, the pseudo-code for PO-TRPO update in Algorithm 1 (Fig. 3) is given as below.

```
Algorithm 2: PO-TRPO
  Compute the estimated advantages $\hat{A}_i$ for all time steps using GAE with the estimated value function $\hat{V}_{\zeta}$.
  Update $\boldsymbol{\theta}$ with objective function (49) and constraints (50)
  Update $\boldsymbol{\zeta}$ with objective function (47) and constraints (48)
```

## 5 Simulation

As a data-driven approach, our DNN policy requires a massive amount of data to learn the SCN behaviour. In this section, we describe how to construct a simulative environment according to the proposed RPL scheme. Particularly, the environments, the DNN policy and the PO-TRPO algorithm (Algorithm 2) are developed under the framework of RLLAB [53]. We make use of trajectories of interacting pedestrians collected from five different data sets, which includes the ETH and Hotel video clips from the ETH Walking Pedestrians (EWAP) [11], the motion capture (MC) data set from [14], as well as the Zara and UCY video clips from [32]. Note that, the Zara and UCY data sets have multiple subsets: Zara01, Zara02, Zara03, UCY01 and UCY03. Thus, there are totally eight different RPL environments, i.e. $\mathbb{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_8\}$. The details of these eight environments are summarised in Table 1.

Each trajectory in these environment provides the ID and a sequences of 2D Cartesian positions of a pedestrian with a sampling period $\Delta t = 0.1$ s. In addition, eight binary grid maps $\mathcal{M}_1, \ldots, \mathcal{M}_8$ representing the occupied space/static obstacles are given. However, these maps are kept unknown to the robot throughout training and evaluation. They are only used to simulate the robot's perception to the environment as the state $\boldsymbol{p}_{\text{obs}}$ and observation $\hat{\boldsymbol{p}}_{\text{obs}}$. Without loss of generality, we use the ETH data set as the evaluation environment and all other data sets in Table 1 as training environments. In other words, the learned policy's performance will be assessed in an RPL environment that is excluded during training, which reflects whether it can properly generalise to uncovered situations.

As some of the trajectories in these environments are of people who were wandering or remained approximately stationary, they are excluded from the candidates of the robot's companion but will still be considered as pedestrians when the robot is navigating in the same environment.

We use a feed-forward NN with two hidden layers as the feature network in our NN policy, containing 256 and 64 Tanh units, respectively. Its output is then fed to a LSTM network with 64 units. The variance of the Gaussian output unit $\sigma$ is chosen to be linearly decaying from 0.5 to 0.05 in 100 training iterations, which effectively encourages exploration during the early stage of learning and ensures convergence of the navigation policy. For GAE, a three-layer feed-forward network with 256, 64 and 16 Tanh units are used, with $\gamma = 0.995$ and $\lambda = 0.96$. The update step size for policy network is adaptively chosen as $\epsilon = 0.01/\sigma$. For GAE update, a fixed step size $\epsilon_1 = 0.1$ is used. The update batch size (Batch_size in Algorithm 1 (Fig. 3)) is 150,000.

In RPL, we consider at most three pedestrians (i.e. $n_p = 3$). Thus, the state $\boldsymbol{p}_{\text{ped}}$ and observation $\hat{\boldsymbol{p}}_{\text{ped}}$ will only describe the three closest pedestrians and omit the others. For the situation where less than three pedestrians are perceived, dummy static pedestrians will be created in the remote corner of the environment so as to maintain the dimensions of $\boldsymbol{p}_{\text{ped}}$ and $\hat{\boldsymbol{p}}_{\text{ped}}$.

Considering a Kobuki Turtlebot 2 with a Hokuyo URG-04LX laser range finder [50] mounted on its top, we specify the sensor limitation of the robot in simulation as follows:

$$\phi^{+}_{\text{ped}} = \phi^{+}_{\text{obs}} = \frac{2\pi}{3} \tag{52}$$

$$\phi^{-}_{\text{ped}} = \phi^{-}_{\text{obs}} = -\frac{2\pi}{3} \tag{53}$$

The measurement noises $\tilde{d}_{\text{ped}}, \tilde{d}_{\text{com}}$ and $\tilde{d}_{\text{obs}}$ are modelled by zero-mean Gaussian $\mathcal{N}(0, \sigma^2_{\text{ped}}), \mathcal{N}(0, \sigma^2_{\text{com}})$ and $\mathcal{N}(0, \sigma^2_{\text{obs}})$ with their variances specified as follows:

$$\sigma_{\text{ped}} = 0.01 d^{j}_{\text{ped}} \tag{54}$$

$$\sigma_{\text{com}} = 0.01 d^{j}_{\text{com}} \tag{55}$$

$$\sigma_{\text{obs}} = 0.01 d^{j}_{\text{obs}} \tag{56}$$

Finally, the maximum translational and rotational velocities are assigned as 0.7 m/s and $(\pi/3)$rad/s, i.e. $0 \le v_{\text{T}} \le 0.7$ and $|v_{\text{R}}| \le (\pi/3)$ and $\Delta t = 0.1$. An example of our RPL environment constructed from the ETH data set is illustrated in Fig. 5.

**Table 1** Details of the 8 RPL environments

| Name | ETH | Hotel | MC | Zara01 |
|---|---|---|---|---|
| no. of trajectories | 365 | 420 | 324 | 148 |

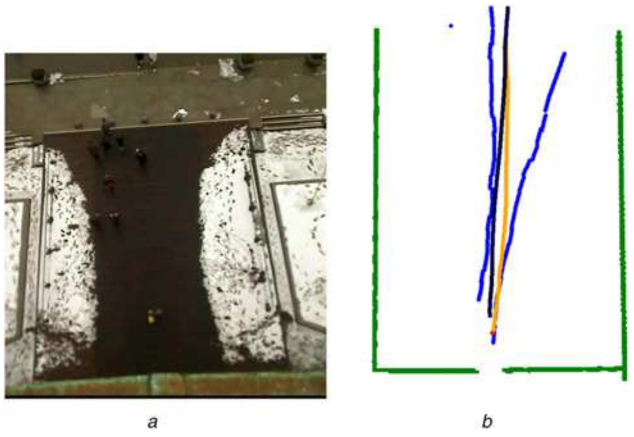


**Fig. 5** *Illustrative example of our RPL simulative environment. The black curve represents the trajectory of the robot navigating towards its goal (the red dot). The yellow curve denotes the trajectory of the robot's companion. Besides, there are a number of blue curves representing the pedestrians perceived by the robot and the green lines denote the fences around the entrance of the university (bottom centre). Note that, all trajectories of pedestrians are not synthesised but captured from the video. Thus, the robot can be thought as playing a role as an extra person in an realistic environment*
*a* Real-world environment
*b* Simulative environment for RPL

## 5.1 Results

The policy network is trained with the data from RPL environments except for the held-out ETH environment. The policy is trained for around 300 iterations. To validate benefits and necessity of using importance sampling in our PO-TRPO algorithm to train the policy, a policy network with gradients estimated by Monte Carlo sampling has been trained for comparison. Particularly, in implementation, the likelihood ratio in (49) is fixed to be 1 through the whole experiment and therefore the original importance sampling is reduced to a Monte Carlo sampling method. Other parameters and training configurations are unchanged and the resulted algorithm is called MC-PO-TRPO. To compare the performance of these two methods, the average discounted returns obtained in each iteration are computed as a measurement of how well the training has progressed. Fig. 6 shows their curves for policies trained by PO-TRPO and MC-PO-TRPO, respectively.

From Fig. 6, it is clear that the policy network trained by MC-PO-TRPO algorithm fails to learn an effective SCN policy in more than 300 iterations. There is little improvement of its average discounted returns, which tells how often the agent succeeds in an SCN episode. On the contrary, the average discounted returns of the policy trained by PO-TRPO quickly rise to a much higher level in <100 epochs. This observation clearly demonstrates the advantages of our importance-sampling-based PO-TRPO

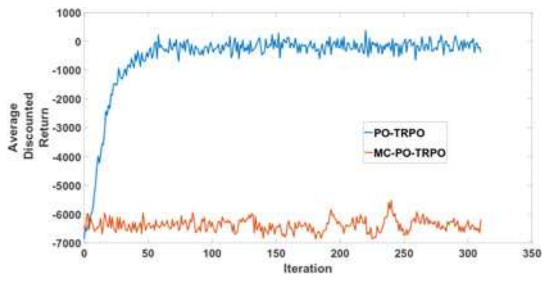


**Fig. 6** *Average discounted returns of policies trained by PO-TRPO and MC-PO-TRPO*

algorithm, which estimates a gradient of parameters $\boldsymbol{\theta}$ that effectively improves the policy network's performance in terms of the SCN objectives formulated in (34).

A more detailed evaluation of our policy's performance is conducted against a planner based on RVO [8], where the robot, its companion and the surrounding pedestrians are treated agents. In every time steps, the positions and velocities of all agents are given to the planner. Note that, for fair comparison, the agents' positions are subject to noise described in (54) and (55). For observations to obstacles, the planner is assumed to have full and perfect knowledge as required in the original RVO algorithm. With this protocol, the robot's position is updated according to the planner's output and updates the positions of the other agents according to their own trajectories in the RPL environments. The same termination conditions in Section 3 are applied to the robot directed by the RVO-based planner to determine whether the robot has conducted an successful navigation. For both of our policy and the RVO-based planner, 300 trials are conducted in the evaluation environment and compute the rates (in percentages) of different terminal conditions (RG: the robot reaches the goal successfully; HC/HP/HO: the robot hits a companion/pedestrian/obstacle; and LC: the robot loses its companion). The performance statistics of our policy and the RVO-based planner in SCN scenarios are listed in Table 2. As for computational efficiency in simulation, both methods can operate at more than 100 Hz, which is much faster than the sampling frequency (10 Hz). We keep the sampling frequency for simulations as low as 10 Hz to maintain the consistency between settings of simulations and experiments, which is to be elaborated in the next section.

It can be seen from Table 2 that our policy performs much better than the RVO-based planner in SCN. The RVO-based planner has a much lower success rate (29.7%) while its rate of LC is 47%, suggesting that it frequently losses its companion in SCN. Clearly, this is due to the fact that RVO is in nature a collision avoidance algorithm. Thus, it simply takes the robot's companion as another normal agent and the robot tends to stay far behind its companion to avoid collision instead of actively following it. On the contrary, our policy achieves a much higher success rate (83.6%). This indicates that it learns to effectively balance the objectives of SCN so that the robot is able to reach the prescribed goal while maintaining its distance to its companion and avoiding collision with other agents in the environment.

In addition to SCN, the scenarios without companion are also tested, which, as analysed in the previous sections, reduces to the traditional social navigation scenarios. The comparative results are shown in Table 3.

For situations without companion, our policy still outperforms the RVO-based planner with higher success rate (85 versus 80%) and lower HP rate (14.7 versus 18%).

Finally, it is worth noting that the RVO-based planner requires velocities of the companion/pedestrians and an accurate global map of the static obstacles. Conversely, our policy depends only on position measurements that are directly accessible from the robot's onboard sensors, which is therefore much simpler and more practical.

**Table 2** Rates (in percentages) of different terminal conditions of our policy and RVO-based planner in SCN scenarios

| Terminal condition | RG | LC | HC | HP | HO |
|---|---|---|---|---|---|
| our policy | 83.6 | 3.3 | 4.3 | 8.7 | 0 |
| RVO | 29.7 | 47 | 0.3 | 23 | 0 |

**Table 3** Rates (in percentages) of different terminal conditions of our policy and RVO-based planner in traditional social navigation scenarios

| Terminal condition | RG | HP | HO |
|---|---|---|---|
| our policy | 85 | 14.7 | 0.3 |
| RVO | 80 | 18 | 2 |

## 6 Experiments

In experiments, we assess the performance of our developed navigation policy by comparing it with humans in the same scenarios. Particularly, a robot and a human are to repeat each specific navigation scenario for ten times, respectively. Then, the following two metrics are calculated:

(i) Average minimum distance to the pedestrians ($\bar{D}_{\text{ped}}$): the average of the minimum distance between the robot/compared human to other pedestrians throughout a trajectory.
(ii) Average maximum distance to the companion ($\bar{D}_{\text{com}}$): the average of the maximum distance between the robot/compared human to its/his companion throughout a trajectory.

We use the same mobile platform (a synchron-drive Turtlebot 2 with a Kobuki base) and the same laser range finder (Hokuyo URG-04LX) simulated in last section. For pedestrian detection and localisation, we adopt the robot operation system (ROS)-compatible leg tracker in [54]. We use an ultra-wideband (UWB) indoor positioning system to localise the companion and the navigation goal, which can then be easily mapped to the observations $\hat{\boldsymbol{p}}_{\text{com}}$ and $d_g$, $\phi_g$ based on the odometry of the robot. Finally, a laptop is placed onboard as the processing unit and the policy is operated with a period of 0.1 s. The experiments are conducted in a narrow corridor with width of 1.56 m as shown in Fig. 7, which is a typical scenario that requires pedestrians to navigate cooperatively.

### 6.1 Scenario 1: traditional social navigation

In this subsection, we examine our method's performance in traditional social navigation scenario. Particularly, the robot is required to pass the corridor with two oncoming pedestrians and arrive at a goal that is 7 m ahead. In addition, a control experiment of three humans (one as the compared human and the other two as pedestrians) is conducted in the same space. The metric $\bar{D}_{\text{ped}}$ is computed. Example trajectories of the robot and the human control are shown in Fig. 8. In the robotic experiments, the trajectories of pedestrians are obtained from the robot's laser range finder while the robot's trajectory is based on its own odometry sensor. On the other hand, all trajectories in the human control experiments are captured using the UWB localisation system.

From Fig. 8, it is clear that the robot with our policy is able to understand human's cooperative behaviour for collision avoidance and navigate in an appropriate manner such that both itself and the other two pedestrians can successfully pass through the corridor, specifically, when observing the two pedestrians (blue and purple) 4 m ahead. The robot started to approach the wall on its left side

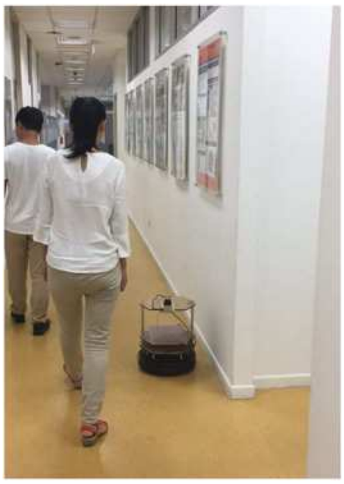
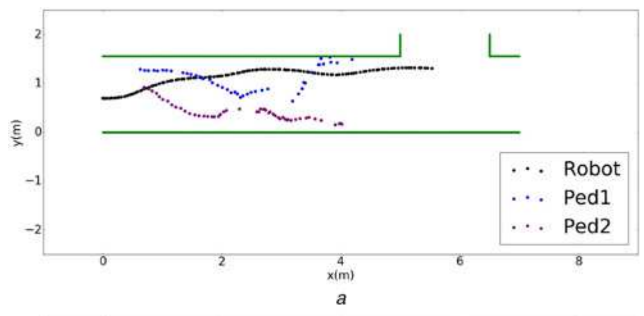


**Fig. 7** *Narrow corridor where experiments are performed*



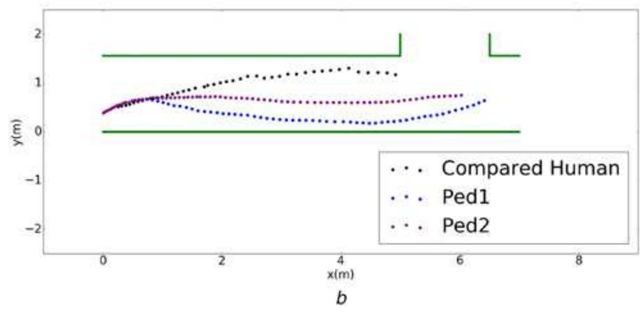
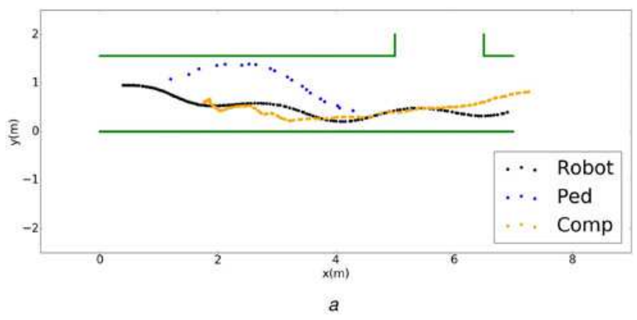


**Fig. 8** *Comparison between the robot with our policy and human control experiment in a social navigation scenario*
*a* Trajectories of the robot (moving from left to right) and other two pedestrians (moving from right to left)
*b* Human control experiment in a similar navigation scenario. The black trajectory is from left to right and the other two are from right to left



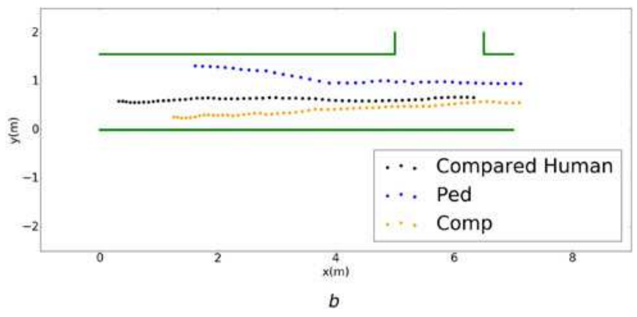


**Fig. 9** *Comparison between the robot with our policy and human control experiment in a SCN scenario*
*a* Trajectories of the robot and its companion (moving from left to right) and a pedestrian (moving from right to left)
*b* Human control experiment in a similar SCN. The black (compared human) and orange (companion) trajectories are from left to right and the blue (pedestrian) trajectory is from right to left

so as to create free space on the right for the pedestrians to smoothly walk through. By comparing both figures in Fig. 8, we can see that the robot is as proactive as human since both black trajectories in Figs. 8*a* and *b* started to make space for the oncoming pedestrians at the early stage of cooperative avoidance process. As for the performance metrics, the average minimal distance to pedestrians for our robot is $\bar{D}_{\text{ped}} = 0.35$ m. Although it is smaller than that of the human control experiments ($\bar{D}_{\text{ped}} = 0.56$ m), this value still indicates a safe and decent navigation behaviour of our robot as its radius is only 0.17 m.

### 6.2 Scenario 2: SCN

In this subsection, the scenario of SCN is studied. A human companion initially standing in front of the robot will start to walk through the same corridor while another pedestrian is passing from the other end. As described in the previous sections, the robot with our policy should closely navigate with its companion and avoid the oncoming pedestrian cooperatively. An additional metric $\bar{D}_{\text{com}}$ is used to evaluate the performance of our policy by comparing with the statistics obtained from another ten human control experiments. Example trajectories are shown in Fig. 9 and the performance metrics $\bar{D}_{\text{ped}}$ and $\bar{D}_{\text{com}}$ are summarised in Table 4.

As shown in Fig. 9 and Table 4, the robot is able to achieve both objectives of SCN. On one hand, it is effectively engaged into the joint collision avoidance process. The resulted behaviour is similar to that observed in the last subsection and the robot even has a slightly larger $\bar{D}_{\text{ped}}$. On the other hand, the average maximum distance $\bar{D}_{\text{com}}$ is 1.05 m, which is within the limit (2 m) we specified in the learning process and nearly the same as that of the compared human, showing that the robot can actively navigate along with its companion instead of deviating to other areas or lagging itself behind. This shows that the robot driven by our policy is able to understand the pace of its companion and achieve a similar sense of companionship in terms of distance.

In sum, the above results demonstrate the practical efficacy of our methods for both the traditional social navigation and the more complicated SCN scenarios. It proves that the policy learned from our RPL simulative environment is transferable to uncovered real-world situations.

**Table 4** Performance metrics of the robot and human controls in SCN scenarios

| | $\bar{D}_{\text{ped}}$, m | $\bar{D}_{\text{com}}$, m |
|---|---|---|
| robot | 0.49 | 1.05 |
| compared human | 0.37 | 1 |

| Name | Zara02 | Zara03 | UCY01 | UCY03 |
|---|---|---|---|---|
| no. of trajectories | 204 | 137 | 413 | 434 |

## 7 Conclusions

In this paper, the problem of SCN has been investigated and formulated under a POMDP framework, with explicit considerations of the limitation and inaccuracy of mobile robots' onboard sensors. The PO-TRPO algorithm has been proposed for optimisation of navigation policies. The RPL scheme has been developed to enable efficient and safe RL of navigation policies by mirroring a large amount of real-world pedestrian trajectories into simulative environments. Comparative simulation and experiment studies have demonstrated the efficacy and superiority of our policy in both SCN and traditional social navigation scenarios.

## 8 References

[1] Thrun, D.F.W.B.S., Fox, D., Burgard, W.: 'The dynamic window approach to collision avoidance', *IEEE Trans. Robot. Autom.*, 1997, **4**, p. 1
[2] Hwang, Y.K., Ahuja, N.: 'A potential field approach to path planning', *IEEE Trans. Robot. Autom.*, 1992, **8**, (1), pp. 23–32
[3] Ge, S.S., Cui, Y.J.: 'New potential functions for mobile robot path planning', *IEEE Trans. Robot. Autom.*, 2000, **16**, (5), pp. 615–620
[4] Trautman, P., Krause, A.: 'Unfreezing the robot: navigation in dense, interacting crowds'. 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 2010, pp. 797–803

[5] Trautman, P., Ma, J., Murray, R.M., *et al.*: 'Robot navigation in dense human crowds: statistical models and experimental studies of human–robot cooperation', *Int. J. Robot. Res.*, 2015, **34**, (3), pp. 335–356

[6] Helbing, D., Molnar, P.: 'Social force model for pedestrian dynamics', *Phys. Rev. E*, 1995, **51**, (5), p. 4282

[7] Helbing, D., Farkas, I., Vicsek, T.: 'Simulating dynamical features of escape panic', *Nature*, 2000, **407**, (6803), pp. 487–490

[8] Van den Berg, J., Guy, S.J., Lin, M., *et al.*: 'Reciprocal n-body collision avoidance'. Robotics Research, Lucerne, Switzerland, 2011, pp. 3–19

[9] Van den Berg, J., Abbeel, P., Goldberg, K.: 'Lqg-mp: optimized path planning for robots with motion uncertainty and imperfect state information', *Int. J. Robot. Res.*, 2011, **30**, (7), pp. 895–913

[10] Van den Berg, J., Lin, M., Manocha, D.: 'Reciprocal velocity obstacles for real-time multi-agent navigation'. IEEE Int. Conf. on Robotics and Automation, Pasadena, USA, 2008, pp. 1928–1935

[11] Pellegrini, S., Ess, A., Schindler, K., *et al.*: 'You'll never walk alone: modeling social behavior for multi-target tracking'. 2009 IEEE 12th Int. Conf. on Computer Vision, Kyoto, Japan, 2009, pp. 261–268

[12] Yamaguchi, K., Berg, A.C., Ortiz, L.E., *et al.*: 'Who are you with and where are you going?'. 2011 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, USA, 2011, pp. 1345–1352

[13] Kuderer, M., Kretzschmar, H., Sprunk, C., *et al.*: 'Feature-based prediction of trajectories for socially compliant navigation'. Robotics: Science and Systems, Sydney, Australia, 2012

[14] Kretzschmar, H., Spies, M., Sprunk, C., *et al.*: 'Socially compliant mobile robot navigation via inverse reinforcement learning', *Int. J. Robot. Res.*, 2016, **35**, (11), doi: 10.1177/0278364915619772

[15] Kim, B., Pineau, J.: 'Socially adaptive path planning in human environments using inverse reinforcement learning', *Int. J. Soc. Robot.*, 2016, **8**, (1), pp. 51–66

[16] Bicchi, A., Fagiolini, A., Pallottino, L.: 'Towards a society of robots', *IEEE Robot. Autom. Mag.*, 2010, **17**, (4), pp. 26–36

[17] Gross, H.-M., Schroeter, C., Mueller, S., *et al.*: 'Progress in developing a socially assistive mobile home robot companion for the elderly with mild cognitive impairment'. 2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), San Francisco, USA, 2011, pp. 2430–2437

[18] Wang, H., Liu, X.P.: 'Adaptive shared control for a novel mobile assistive robot', *IEEE/ASME Trans. Mechatronics*, 2014, **19**, (6), pp. 1725–1736

[19] Argall, B.D., Chernova, S., Veloso, M., *et al.*: 'A survey of robot learning from demonstration', *Robot. Auton. Syst.*, 2009, **57**, (5), pp. 469–483

[20] Abbeel, P., Ng, A.Y.: 'Apprenticeship learning via inverse reinforcement learning'. Proc. of the Twenty-first Int. Conf. on Machine Learning, Alberta, Canada, 2004, p. 1

[21] Ziebart, B.D., Maas, A.L., Bagnell, J.A., *et al.*: 'Maximum entropy inverse reinforcement learning'. Association for the Advancement of Artificial Intelligence, Palo Alto, USA, 2008, pp. 1433–1438

[22] Ratliff, N.D., Bagnell, J.A., Zinkevich, M.A.: 'Maximum margin planning'. Proc. of the 23rd Int. Conf. on Machine Learning, Pittsburgh, USA, 2006, pp. 729–736

[23] Ziebart, B.D., Ratliff, N., Gallagher, G., *et al.*: 'Planning-based prediction for pedestrians'. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, St. Louis, USA, 2009, pp. 3931–3936

[24] Henry, P., Vollmer, C., Ferris, B., *et al.*: 'Learning to navigate through crowded environments'. 2010 IEEE Int. Conf. on Robotics and Automation (ICRA), Anchorage, USA, 2010, pp. 981–986

[25] Vernaza, P., Bagnell, D.: 'Efficient high dimensional maximum entropy modeling via symmetric partition functions'. Advances in Neural Information Processing Systems, Lake Tahoe, USA, 2012, pp. 575–583

[26] Kitani, K.M., Ziebart, B.D., Bagnell, J.A., *et al.*: 'Activity forecasting'. European Conf. on Computer Vision, Florence, Italy, 2012, pp. 201–214

[27] Choi, J., Kim, K.-E.: 'Map inference for Bayesian inverse reinforcement learning'. Advances in Neural Information Processing Systems, Granada, Spain, 2011, pp. 1989–1997

[28] Kim, K., Lee, D., Essa, I.: 'Gaussian process regression flow for analysis of motion trajectories'. 2011 IEEE Int. Conf. on Computer vision (ICCV), Barcelona, Spain, 2011, pp. 1164–1171

[29] Alahi, A., Goel, K., Ramanathan, V., *et al.*: 'Social lstm: human trajectory prediction in crowded spaces'. Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016, pp. 961–971

[30] Robicquet, A., Sadeghian, A., Alahi, A., *et al.*: 'Learning social etiquette: human trajectory understanding in crowded scenes'. European Conf. on Computer Vision, Amsterdam, The Netherlands, 2016, pp. 549–565

[31] Johansson, A., Helbing, D., Shukla, P.K.: 'Specification of the social force pedestrian model by evolutionary adjustment to video tracking data', *Adv. Complex Syst.*, 2007, **10**, (supp02), pp. 271–288

[32] Lerner, A., Chrysanthou, Y., Lischinski, D.: 'Crowds by example', *Comput. Graph. Forum*, 2007, **26**, (3), pp. 655–664

[33] Helbing, D., Johansson, A.: 'Pedestrian, crowd and evacuation dynamics'. Encyclopedia of Complexity and Systems Science, 2009, pp. 6476–6495

[34] Müller, J., Stachniss, C., Arras, K., *et al.*: 'Socially inspired motion planning for mobile robots in populated environments'. Proc. of Int. Conf. on Cognitive Systems, Karlsruhe, Germany, 2008

[35] Mehta, D., Ferrer, G., Olson, E.: 'Autonomous navigation in dynamic social environments using multi-policy decision making'. 2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Deajeon, South Korea, 2016, pp. 1190–1197

[36] Foka, A.F., Trahanias, P.E.: 'Probabilistic autonomous robot navigation in dynamic environments with human motion prediction', *Int. J. Soc. Robot.*, 2010, **2**, (1), pp. 79–94

[37] Seder, M., Petrovic, I.: 'Dynamic window based approach to mobile robot motion control in the presence of moving obstacles'. 2007 IEEE Int. Conf. on Robotics and Automation, Roma, Italy, 2007, pp. 1986–1991

[38] Fiorini, P., Shiller, Z.: 'Motion planning in dynamic environments using velocity obstacles', *Int. J. Robot. Res.*, 1998, **17**, (7), pp. 760–772

[39] Schulman, J., Moritz, P., Levine, S., *et al.*: 'High-dimensional continuous control using generalized advantage estimation', arXiv preprint arXiv:1506.02438, 2015

[40] Schulman, J., Levine, S., Moritz, P., *et al.*: 'Trust region policy optimization', CoRR, abs/1502.05477, 2015

[41] LeCun, Y., Bengio, Y., Hinton, G.: 'Deep learning', *Nature*, 2015, **521**, (7553), pp. 436–444

[42] Pfeiffer, M., Schaeuble, M., Nieto, J., *et al.*: 'From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots', arXiv preprint arXiv:1609.07910, 2016

[43] Chen, Y.F., Liu, M., Everett, M., *et al.*: 'Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning', arXiv preprint arXiv:1609.07845, 2016

[44] Zhu, Y., Mottaghi, R., Kolve, E., *et al.*: 'Target-driven visual navigation in indoor scenes using deep reinforcement learning', arXiv preprint arXiv:1609.05143, 2016

[45] Choi, D.-G., Bok, Y., Kim, J.-S., *et al.*: 'Extrinsic calibration of 2-d lidars using two orthogonal planes', *IEEE Trans. Robot.*, 2016, **32**, (1), pp. 83–98

[46] Miller, L.M., Murphey, T.D.: 'Optimal planning for target localization and coverage using range sensing'. 2015 IEEE Int. Conf. on Automation Science and Engineering (CASE), Gothenburg, Sweden, 2015, pp. 501–508

[47] Endres, F., Hess, J., Sturm, J., *et al.*: '3-d mapping with an rgb-d camera', *IEEE Trans. Robot.*, 2014, **30**, (1), pp. 177–187

[48] Foix, S., Alenya, G., Andrade-Cetto, J., *et al.*: 'Object modeling using a tof camera under an uncertainty reduction approach'. 2010 IEEE Int. Conf. on Robotics and Automation (ICRA), Anchorage, USA, 2010, pp. 1306–1312

[49] Liu, M., Siegwart, R.: 'Topological mapping and scene recognition with lightweight color descriptors for an omnidirectional camera', *IEEE Trans. Robot.*, 2014, **30**, (2), pp. 310–324

[50] Kneip, L., Tâche, F., Caprari, G., *et al.*: 'Characterization of the compact hokuyo urg-04lx 2d laser range scanner'. IEEE Int. Conf. on Robotics and Automation, Kobe, Japan, 2009, pp. 1447–1454

[51] Hochreiter, S., Schmidhuber, J.: 'Long short-term memory', *Neural Comput.*, 1997, **9**, (8), pp. 1735–1780

[52] Nocedal, J., Wright, S.J.: 'Numerical Optimization' (Springer-Verlag, New York, 1999)

[53] Duan, Y., Chen, X., Houthooft, R., *et al.*: 'Benchmarking deep reinforcement learning for continuous control', arXiv preprint arXiv:1604.06778, 2016

[54] Leigh, A., Pineau, J., Olmedo, N., *et al.*: 'Person tracking and following with 2d laser scanners'. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA), Seattle, USA, 2015, pp. 726–733