Learning to Navigate in Human Environments via Deep Reinforcement Learning

Xingyuan Gao[1,2], Shiying Sun[1(✉)], Xiaoguang Zhao[1], and Min Tan[1]

[1] The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
{gaoxingyuan2016,sunshiying2013,xiaoguang.zhao,min.tan}@ia.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Mobile robots have been widely applied in human populated environments. To interact with humans, the robots require the capacity to navigate safely and efficiently in complex environments. Recent works have successfully applied reinforcement learning to learn socially normative navigation behaviors. However, they mostly focus on modeling human-robot cooperations and neglect complex interactions between pedestrians. In addition, these methods are implemented using assumptions of perfect sensing about the states of pedestrians, which makes the model less robust to the perception uncertainty. This work presents a novel algorithm to learn an efficient navigation policy that exhibits socially normative navigation behaviors. We propose to employ convolutional social pooling to jointly capture human-robot cooperations and inter-human interactions in an actor-critic reinforcement learning framework. In addition, we propose to focus on partial observability in socially normative navigation. Our model is capable to learn the representation of unobservable states with recurrent neural networks and further improves the stability of the algorithm. Experimental results show that the proposed learning algorithm enables robots to learn socially normative navigation behaviors and achieves a better performance than state-of-the-art methods.

Keywords: Socially normative navigation · Reinforcement learning

1 Introduction

In recent years, mobile robots have been developed to provide mobile services in human populated environments, such as shopping malls and subway stations. The task requires the robot not only to be socially normative with respect to person's space but also to navigate efficiently in crowded environments.

This work is partially supported by the National Natural Science Foundation of China under Grants 61673378 and 61421004.

T. Gedeon et al. (Eds.): ICONIP 2019, LNCS 11953, pp. 418–429, 2019.
https://doi.org/10.1007/978-3-030-36708-4_34

Different from traditional navigation methods, in "socially normative navigation", pedestrians are considered as more than dynamic obstacles, but rather as rational agents which maintain social relations and comply with social norms.

A usual method [1,2] on socially normative navigation treats pedestrians as dynamic obstacles and uses hand-crafted rules for collision avoidance. However, it is infeasible to program the complex behaviors manually and these methods do not consider human behaviors. To this end, some works [3,4] try to find a collision-free path by forecasting the future states of pedestrians according to pedestrians' motion. Nevertheless, in dense crowds, the set of potential paths may occupy most of the space, which causes the freezing robot problem [5]. A lot of works [6,7] are presented to capture the interdependencies of the trajectories and learn human navigation behaviors. However, these methods suffer from high computational cost. Thus, recent works have applied reinforcement learning (RL) to solve the above issues.

Although several works [8–11] successfully apply RL to the task of socially normative navigation and present good performance, there are some issues to address. First, existing methods [8–10] focus on modeling the effect of pedestrian motion on robot, but neglect complex interactions between pedestrians. Jointly capturing human-robot cooperations and inter-human interactions is essential for the robot to navigate safely and efficiently in complex environments. To this end, Chen et al. [11] propose to learn the collective importance of neighboring humans with a self-attention mechanism. However, the method relies on predicting the future states of pedestrians to find an action form the state-value function, which becomes infeasible in complex environments where the behaviors of pedestrians are difficult to predict. What's more, the methods [8–11] rely on perfect sensors to obtain the states of pedestrians. However, the assumption does not hold in real world. Many sources can lead to pedestrian tracking failures, such as sensor limitations, occlusions and perception uncertainty, which results in partial observability. What's more, pedestrians' intended goal position, preferred speed is unobservable. Existing methods [8–11] are unable to infer unobservable states since inferring the unobservable states often relies on history information, which makes the algorithm less robust to the partial observability.

Inspired by the existing methods [8–11], in this work, we present a novel framework to address the above issues of socially normative navigation using RL. The contributions of this work are as follows: (1) We propose to use convolutional social pooling [12] to encode the states of pedestrians and robot. Our model can handle an arbitrary number of pedestrians while jointly capturing human-robot cooperations and inter-human relations. (2) We propose to focus on partial observability in socially normative navigation and employ a recurrent neural network (RNN) architecture called gated recurrent unit (GRU) [13] to infer the unobservable states and further improve the robustness of algorithm. (3) The simulation results show that our proposed model achieves a better performance than state-of-the-art methods.

The paper is organized as follows: In Sect. 2, we introduce the problem formulation. In Sect. 3, we present the details of our approach. In Sect. 4, we detail our experiments and discuss the experiment results. Finally, the conclusions are drawn in Sect. 5.

2 Problem Formulation

2.1 Problem Formulation

The task of socially normative navigation is to steer the robot from current position to a desired goal in pedestrian-rich environments, which can be formulated as a sequential decision making problem in a RL framework [8–11]. Let s_t, a_t denote robot's state and action at timestep t. Denote b_t^i the observed state of a nearby pedestrian i. For each agent (including robot and pedestrian) i, the position and velocity can be described by $p^i = [p_x^i, p_y^i]$ and $v^i = [v_x^i, v_y^i]$ in 2D. The joint observation states s_t^{jn} can be divided into two parts: $s_t^{jn} = [s_t, b_t]$, where $s = [p^0, p_g^0, v^0, v_{pref}^0, r^0]$ denotes robot's current position, goal position, velocity, preferred speed and radius; $b = [b^1, ..., b^n]$ refers to the observed state of n pedestrians in the field-of-view, where $b^i = [p^i, v^i, r^i]$. The action a_t is robot's velocity.

The goal of RL is to learn a policy $\pi_\theta : s_t^{jn} \mapsto a_t$ which maximizes the expected discounted reward:

$$J = \mathbb{E}_\tau \left[\sum_{t=1}^{\infty} \gamma^{t-1} R(s_t^{jn}, a_t) \right] \tag{1}$$

where $\tau = (s_1^{jn}, a_1, ...)$ denotes the whole trajectory and $a_t \sim \pi_\theta(\cdot|s_t^{jn})$. $R(s_t^{jn}, a_t)$ is the reward received. $\gamma \in (0, 1)$ is a discount factor.

As mentioned in [10], previous algorithms [8,9,11] employ a state-value function $V(s_t^{jn})$ to estimate the expected reward at state s_t^{jn}. However, the optimal policy $\pi^*(s_t^{jn})$ can only be extracted indirectly from value function $V^*(s_t^{jn})$:

$$\pi^*(s_t^{jn}) = \arg\max_{a_t} R(s_t^{jn}, a_t) + \gamma \int_{s_{t+1}^{jn}} P(s_{t+1}^{jn}|s_t^{jn}, a_t) V^*(s_{t+1}^{jn}) ds_{t+1}^{jn} \tag{2}$$

A major challenge in finding the optimal policy is that the state-transition probability $P(s_{t+1}^{jn}|s_t^{jn}, a_t)$ is unknown. In order to avoid computing integrals in (2), previous algorithms estimate the next state s_{t+1}^{jn} by assuming that pedestrians continue their current velocities for a sufficiently large duration. However, the assumption of constant velocity neglects the effects of inter-human relations, which is not valid in dense crowds [10].

2.2 Policy-Based Learning

To overcome the shortcomings of the assumption of constant velocity, we consider a recently proposed actor-critic framework called Proximal Policy Optimization (PPO) [14,15] instead of using a state-value function. What's more, our model contains a RNN architecture, which will be detailed in Sect. 3. PPO has two improvements which make it convenient to facilitate the use of RNN. First, PPO algorithm relies only on first order gradients. Second, PPO uses a truncated

version of generalized advantage estimation with K-steps returns to support variable length episodes:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + ... + (\gamma\lambda)^{K-t+1}\delta_{K-1} \tag{3}$$

where

$$\delta_t = r_t + \gamma V(s_{t+1}^{jn}) - V(s_t^{jn}) \tag{4}$$

where K varies from episode to episode. t specifies the timestep index in $[0, K]$.

Given a current policy $\pi_{\theta_{old}}$, let the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t^{jn})}{\pi_{\theta_{old}}(a_t|s_t^{jn})}$, PPO optimizes the policy by maximizing the following surrogate loss function:

$$L_t^{clip}(\theta) = \hat{\mathbb{E}}_t\big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\big] \tag{5}$$

where the first term inside the min is an approximation of the expected advantages, and the second term removes the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$ by clipping the probability ratio.

To approximate policy and value function with a neural network architecture, the objective function is formulated to combine the policy surrogate loss and a value function error term, which is maximized:

$$L_t^{clip+vs+s}(\theta) = \hat{\mathbb{E}}_t\big[L_t^{clip}(\theta) - c_1 L_t^{vf}(\theta) + c_2 S[\pi_\theta](s_t^{jn})\big] \tag{6}$$

where S denotes an entropy bonus added to ensure sufficient exploration, and L_t^{vf} is a squared-error loss $(V_\theta(s_t^{jn}) - V_t^{targ})^2$, and c_1, c_2 are hyper-parameters.

3 Approach

In this section, first, we introduce the basic concepts in our RL framework. Second, we describe the architecture of our model. Finally, we describe the training framework and detail the training setup.

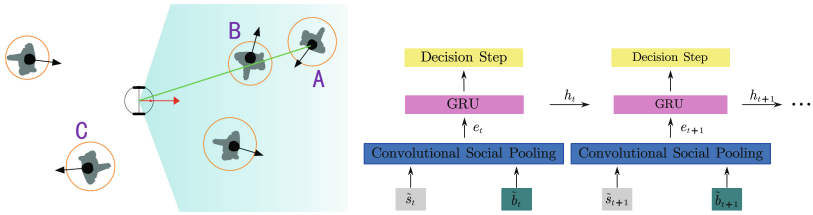
3.1 Basic Concepts

We introduce the basic concepts in our RL framework, including state space, action space and reward function.

(1) State Space: Different from the coordinate frame in [8–11], this work uses a robot-centric coordinate frame with the origin at current robot position, and the x-axis points in the direction of linear velocity of the robot. The states of the robot and per pedestrian i through coordinate transformation are parameterized:

$$\tilde{s} = [\tilde{p}_{gx}^0, \tilde{p}_{gy}^0, v_{pref}^0, \tilde{v}^0, r^0] \tag{7}$$

$$\tilde{b}^i = [\tilde{p}_x^i, \tilde{p}_y^i, \tilde{v}_x^i, \tilde{v}_y^i, r^i] \tag{8}$$



(a) Sensor limitations and occlusions in our simulation

(b) GRU unrolled for 2 timesteps

Fig. 1. (a): Sensor limitations and occlusions in our simulation. It is assumed that only the pedestrians in front of the robot can be detected, which simulates the restricted field-of-view of sensors. The pedestrian A is regarded as occluded and can not be detected if the line connecting pedestrian A and robot intersects the circle which is centered at pedestrian B with its radius. (b): GRU unrolled for 2 timesteps. At each time, GRU accepts the final encodings e_t and stores the important information in h_t for next action. So the decision step can utilize information about past.

where $[\tilde{p}^0_{gx}, \tilde{p}^0_{gy}]$ is the goal position of robot in robot-centric coordinate frame, $\tilde{v}^0$ is the translational velocity of the robot.

The previous works [8–11] are based on the assumption of perfect sensing about the states of pedestrians, which is not valid in in reality. In our simulation, we consider the sensor limitations and occlusions which widely exist in real world, as illustrated in Fig. 1(a).

(2) Action Space: The robot's action consists of a translational velocity and change in heading angle. Similarly to [10], the action space is discretized into 11 permissible discrete velocity vectors: with a translational velocity of v_{pref}, there are 5 headings evenly spaced between $\pm\pi/6$, for translational velocity of $0.5v_{pref}$ and 0 the heading choices are $[-\pi/6, 0, \pi/6]$. Although discretizing the action space may lose some information about the structure of the action domain, it makes the algorithm convenient to combine with other obstacle avoidance algorithms in the future work, such as [16].

(3) Reward Function: Inspired by [8], a reward function is designed to guide the robot to achieve the goal without collisions, which awards the robot for reaching its goal and penalizes the robot for colliding with pedestrians or intruding pedestrians' intimate space.

$$R_t(s_t^{jn}, a_t) = \begin{cases} -0.25 & if \quad d_t^{cp} \leqslant 0 \\ -0.1 + 0.5 \cdot d_t^{cp} & if \quad d_t^{cp} < 0.2 \\ 1 & if \quad p^0 = p_g^0 \\ 0 & otherwise \end{cases} \tag{9}$$

where d_t^{cp} is the minimum distance between robot and the closest pedestrian.

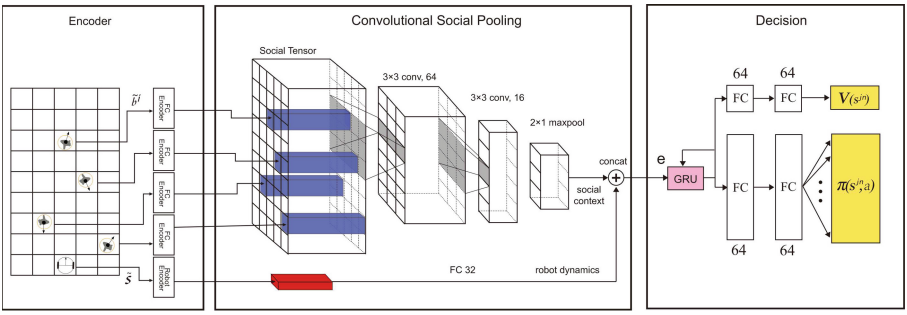


Fig. 2. The total architecture of our model. The pedestrian encoder (FC encoder) is a fully connected layer with shared weights. The convolutional social pooling layers model the human-robot cooperations and inter-human interactions. In decision step, the final encodings are fed into the GRU layer, as unrolled for 2 timesteps in Fig. 1(b).

3.2 Network Architecture

The total architecture of our model is shown in Fig. 2. We introduce our network from the following three aspects:

(1) Convolutional Social Pooling Module: The number of surrounding pedestrians can vary dramatically in different scenes, which certainly brings a great challenge to many learning-based planning methods that require a fixed-size input [10]. Everett et al. [10] propose to feed the states of surrounding pedestrians into LSTMs and take the LSTM's final hidden state as a fixed-length, encoded state of the pedestrians. Although the LSTM encoder can handle an arbitrary number of agent inputs, it fails to capture the inter-human interactions in the scene. Chen et al. [11] propose to aggregate the states into a fixed-length embedding vector by a self-attention mechanism.

Inspired by [12], we propose to extend convolutional social pooling for robustly learning inter-human interactions. We set up our social tensor by defining a grid based on the sensor bandwidth. As shown in Fig. 2, a 9×5 spatial grid is defined in front of the robot, where the grids are separated by a distance of 1 m which approximately equals the diameter (size) of a pedestrian. The state of each pedestrian $\tilde{b}^i$ is fed into a fully connected layer with a hidden layer of 64 units and Leaky ReLU nonlinearity to capture the dynamics of pedestrian. The social tensor is formed by populating this grid with the feature vector according to the location of pedestrians. By defining the social tensor, the model can encode the states of a variable number of pedestrians into a fixed-length vector. Aiming to obtain local features within the spatial grid of the social tensor, two 3×3 convolutional layers are applied to the social tensor. The output of the convolutional layer is fed into a 2×1 max-pooling layer to add local translational invariance. The output of max-pooling layer is denoted as social context encoding which captures inter-human interactions. In addition, the state of the robot $\tilde{s}$ is passed through a fully connected layer with a hidden layer of 32 units and Leaky ReLU nonlinearity to capture the dynamics encoding of robot.

Algorithm 1. Framework of learning

Initialize network weights θ by supervised learning
for iteration = 1,2,3... **do**
  for robot = 1,2,3,..., N **do**
    run policy $\pi_{\theta_{old}}$ in environment for an episode, collecting $\{s_t^{jn}, a_t, r_t\}$
    compute advantage estimates $\hat{A}_1$...
  Update θ through time with learning rate l_r by Adam w.r.t $L^{clip+vs+s}$ for F epochs
  $\theta_{old} \leftarrow \theta$

To jointly model human-robot cooperations and inter-human interactions, the two encodings are concatenated to form the final encoding for the decision step.

(2) Recurrent Module: Due to the existence of sensor limitations, occlusions and perception uncertainty, it seems difficult to perfectly obtain the states of surrounding pedestrians. What's more, the pedestrians' intended goal position, preferred speed is unobservable. The reasons stated above model the task as a partially-observable sequential decision making problem. It is challenging for the models in [8–11] to infer the unobservable states since inferring the unobservable states often relies on history.

Inspired by recent works [14,17,18], we add a GRU layer before the final fully connected layers, as shown in Fig. 2. The GRU unrolled is shown in Fig. 1(b). The GRU layer stores important information in its hidden states for making decision on the next action, which helps to infer unobservable state and capture long-term dependency on history.

(3) Decision Module: In decision step, the final encodings are fed into the GRU. The final hidden state is passed through two parallel fully connected layers with two hidden layers of 64 units and tanh nonlinearities. The outputs are policy $\pi(s_t^{jn})$ represented as discrete probability distribution across actions and a state-value function $V(s_t^{jn})$.

3.3 Training Details

(1) Training Scheme: The learning scheme is described in Algorithm 1 (adapted from [14]). Inspired by [8–11], the network is first initialized by supervised learning on a set of state-action-value pairs generated by GA3C-CADRL [10]. The loss of supervised learning phase consists of square-error loss on the value output and cross-entropy loss on the policy output. The initialization step enables the robot to reach the goal in the scenarios with few pedestrians while obtaining positive reward, which improves the convergence ability of the algorithm. The second training step improves the solution with PPO [14,15]. In each iteration, each of N robots simultaneously follows the same policy $\pi_{\theta_{old}}$ to complete an episode. Then the advantages $\hat{A}_1$... are estimated using Eq. (3) with state-value function $V(s_t^{jn})$. The collected episodes are used to construct the surrogate loss and the updates begin at the beginning of the episode and proceed forward

through time to the end of the episode for training GRU. The loss is optimized with the Adam [19] optimizer for F epochs.

(2) Training Scenarios: To improve the generalization ability of the model, experiences are generated from simulations of randomly-generated scenarios. In each episode, the robot is generated with fixed initial position, but randomly selected orientation within a 10.0×10.0 square domain. Following current learned policy, the robot tries to navigate to a randomly-sampled goal. The simulated pedestrians are controlled by a random assortment of policies such as ORCA [20] and Social Force [21] to reach the randomly-generated goals. The radius of pedestrians $r \in [0.2, 0.5]m$. The number of the pedestrians in scenarios varies from 2 to 8. In addition, to avoid the simple cases where the robot easily reaches the goal without encountering with pedestrians, the goals are sampled randomly at a distance of more than 5 m to the initial position. The complex training scenarios lead the robot to explore the high-dimensional observation space and improve the robustness of the model.

4 Experiment

This section is organized as follows: First, the details of computation are provided. Second, qualitative experiments are carried out in simulation. Third, experiment metrics and quantitative experiment results are discussed. We refer to our whole model as SNNRL-GRU. To demonstrate the benefit of GRU layer, a copy of our model without GRU (SNNRL) is trained for comparison.

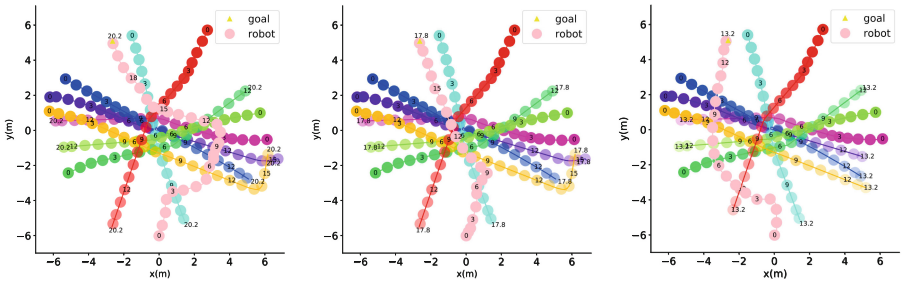
4.1 Computational Details

We implement the model with Tensorflow [22] and train it on a computer with an NVIDIA GTX 1080 graphics card. The offline training takes about 28 h to complete $2.2 \cdot 10^7$ timesteps for the policy to converge. A query of trained model only takes 1.8 ms on an i7-7700K CPU.

4.2 Qualitative Experiments

Qualitative experiments are carried out to evaluate the performance of the algorithm. The trajectories obtained by different algorithms are compared in the same crossing scenario, as illustrated in Fig. 3. In qualitative experiments, aiming to generate the same pedestrian trajectories for comparison, we set the robot invisible to the pedestrians. That means the pedestrians will not cooperate with the robot.

The performance of each algorithm can be roughly evaluated by navigation time and clearance between robot and pedestrians. As shown in Fig. 3, CADRL [8] only considers one neighbor pedestrian, which leads the robot to take longer path for passing on the right side of crowds. The robot controlled by GA3C-CADRL [10] slows down in 6.0 s–9.0 s, hesitating about which side to pass the crowds, which makes the robot maintain smaller clearance to the pedestrians. While, our SNNRL-GRU recognizes the inter-human interactions and finds an efficient and safe path to navigate to the goal with the shortest time.



(a) CADRL (b) GA3C-CADRL (c) SNNRL-GRU

Fig. 3. The trajectories obtained by different algorithms. The goal of robot is indicated by a yellow triangle. The trajectory of robot is visualized with a pink circle. The circles of other colors represent the trajectories of pedestrians and lighten as time increases. The numbers indicate the time at agent's position. In addiction, the trajectories are recorded until the robot reaches the goal. (a): trajectories of using policy CADRL with navigation time 20.2 s. (b): trajectories of using GA3C-CADRL with navigation time 17.8 s (c): trajectories of using our SNNRL-GRU with navigation time 13.2 s. (Color figure online)

4.3 Experiment Metrics

In order to evaluate quantitatively the performance of our algorithm and compare the different algorithms, the following evaluation metrics are defined:

(1) Success rates: The percentage of cases where the robot reaches their goals within a certain time limit without colliding with pedestrians.
(2) Extra time to goal: The difference between the travel time that robot spends to reach the goal and the lower bound of the travel time (going straight toward the goal at preferred speed [8]).
(3) Proxemic intrusions (PI): According to the distance between robot and the closest pedestrian, a percentage of time spent in pedestrian's intimate zone for a complete trajectory is defined by:

$$PI = \frac{1}{M} \sum_{t=1}^{M} \mathbb{1}(\|p_t^0 - p_t^{cp}\|_2 - r^0 - r^{cp} < 0.2) \tag{10}$$

where M represents the total timesteps in a complete trajectory. p_t^{cp}, r^{cp} denote the position of the closest pedestrian to the robot and its radius at timestep t. $\mathbb{1}(\cdot)$ is the indicator function.

4.4 Quantitative Experiments

We implement three state-of-the-art methods, ORCA [20], CADRL [8] and GA3C-CADRL [10] as baseline method to present a comparison with our algorithm. The training process of the two learning-based algorithms (CADRL and

Table 1. Results of different algorithms in different scenes.

Num[a]	Method	Success (%)	Collision (%)	Stuck (%)	Extra time to goal (s)[b] [Avg / 75th / 90th pctl]	PI(%)
2	ORCA	75.0	25.0	0.0	0.684 / 0.789 / 1.383	12.476
	CADRL	97.6	2.2	0.2	1.031 / 1.549 / 2.182	3.509
	GA3C-CADRL	98.4	1.6	0.0	0.915 / 1.415 / 1.981	5.387
	SNNRL-GRU	97.8	2.2	0.0	1.259 / 1.746 / 2.259	1.450
	SNNRL	97.4	2.4	0.2	1.331 / 1.944 / 2.403	1.631
5	ORCA	61.2	38.8	0.0	0.878 / 1.079 / 1.846	11.593
	CADRL	84.4	11.2	4.4	2.168 / 2.937 / 4.827	3.599
	GA3C-CADRL	90.2	7.6	2.2	2.042 / 2.984 / 4.133	4.274
	SNNRL-GRU	96.0	3.8	0.2	2.373 / 3.375 / 4.435	1.537
	SNNRL	93.4	6.2	0.4	2.614 / 3.611 / 4.834	1.416
8	ORCA	49.8	50.2	0.0	1.135 / 1.420 / 2.027	12.352
	CADRL	76.4	12.4	11.2	3.234 / 4.618 / 6.357	4.173
	GA3C-CADRL	86.4	8.2	5.4	3.338 / 4.622 / 6.561	3.713
	SNNRL-GRU	95.0	4.8	0.2	3.473 / 4.745 / 6.835	1.807
	SNNRL	91.0	8.6	0.4	3.621 / 5.038 / 7.184	1.914

[a] Num represents the number of pedestrians in different scenes.
[b] pctl is the abbreviation of percentile which is used to measure the dispersion degree of experiment results.

GA3C-CADRL) is conducted under the same simulation setup, except that CADRL is trained in a two-agent environment since it does not support multi-agent training.

For the sake of comparing the performances of different algorithms, we define 3 different test scenes with different numbers of pedestrians, as illustrated in Table 1. Each pedestrian is controlled by a randomly selected policy such as ORCA, Social Force and zero velocity. The experiments of each algorithm are conducted under the same 500 test cases.

The test results obtained by different algorithms are listed in Table 1. As can be seen, compared to ORCA [20], three learning-based algorithms take more time to reach the goal as the algorithms try to adapt their pathes to people. However, ORCA achieves a very low success rate due to the short-sighted and conservative behaviors. In $n = 2$ pedestrians, our algorithm gets a similar performance with other learning-based algorithms. GA3C-CADRL is slightly better. However, with the increase of number of pedestrians, the advantages of our proposed algorithm become more obvious. The success rate of CARDL drops to 84.4% when $n = 5$. As described in [8], its minimax implementation is limited in that it only considers one neighbor at a time. The lower success rate shows the importance of considering all the pedestrians simultaneously. When $n = 8$, our model significantly outperforms the other algorithms. The success rate of GA3C-CADRL drops to 86.4%, while our SNNRL-GRU remains 95%. What's more, our algorithm maintains less proxemic intrusions (PI) across all the test

cases. The results demonstrate that our proposed algorithm is more applicable to learn a policy that exhibits socially normative navigation behaviors. By comparing the experiment results of two copies of our model, it can be concluded that the GRU layer makes the model more robust.

5 Conclusions

In this paper, we propose a novel algorithm to learn a policy that exhibits socially normative navigation behaviors. Our model uses a convolutional social pooling layer that robustly models human-robot cooperations and complex interactions between pedestrians. Moreover, we focus on partial observability in socially normative navigation. We employ a recurrent policy that infers unobservable states from the history information and further improves the robustness of algorithm. The experiment results show that our approach outperforms the state-of-the-art methods in complex scenarios.

References

1. Kirby, R., Simmons, R., Forlizzi, J.: COMPANION: a constraint-optimizing method for person-acceptable navigation. In: IEEE International Symposium on Robot and Human Interactive Communication, Toyama, pp. 607–612 (2009)
2. Phillips, M., Likhachev, M.: SIPP: safe interval path planning for dynamic environments. In: IEEE International Conference on Robotics and Automation, Shanghai, pp. 5628–5635 (2011)
3. Unhelkar, V.V., Pérez-D'Arpino, C., Stirling, L., Shah, J.A.: Human-robot co-navigation using anticipatory indicators of human walking motion. In: IEEE International Conference on Robotics and Automation, Seattle, pp. 6183–6190 (2015)
4. Aoude, G.S., et al.: Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. Auton. Robots **35**(1), 51–76 (2013)
5. Trautman, P., Krause, A.: Unfreezing the robot: navigation in dense, interacting crowds. In: IEEE International Conference on Intelligent Robots and Systems, Taipei, pp. 797–803 (2010)
6. Kuderer, M., Kretzschmar, H., Sprunk, C., Burgard, W.: Feature based prediction of trajectories for socially compliant navigation. In: Robotics: Science and Systems (2012)
7. Kretzschmar, H., Spies, M., Sprunk, C., Burgard, W.: Socially compliant mobile robot navigation via inverse reinforcement learning. Int. J. Robot. Res. **35**(4), 1289–1307 (2016)
8. Chen, Y.F., Liu, M., Everett, M., How, J.P.: Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: IEEE International Conference on Robotics and Automation, Singapore, pp. 285–292 (2017)
9. Chen, Y.F., Everett, M., Liu, M., How, J.P.: Socially aware motion planning with deep reinforcement learning. In: IEEE International Conference on Intelligent Robots and Systems, Vancouver, BC, pp. 1343–1350 (2017)
10. Everett, M., Chen Y.F., How, J.P.: Motion planning among dynamic, decision-making agents with deep reinforcement learning. In: IEEE International Conference on Intelligent Robots and Systems, Madrid, pp. 3052–3059 (2018)

11. Chen, C., Liu, Y., Kreiss, S., Alahi, A.: Crowd-robot interaction: crowd-aware robot navigation with attention-based deep reinforcement learning. In: IEEE International Conference on Robotics and Automation, Montreal, pp. 6015–6022 (2019)
12. Deo, N., Trivedi, M.M.: Convolutional social pooling for vehicle trajectory prediction. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1468–1476 (2018)
13. Cho, K., Van Merrienboer, B., Gulcehre, C.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint (2014). arXiv:1406.1078
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint (2017). arXiv:1707.06347
15. Heess, N., et al.: Emergence of locomotion behaviours in rich environments. arXiv preprint (2017). arXiv:1707.02286
16. Lu, D.V., Hershberger D., Smart, W.D.: Layered costmaps for context-sensitive navigation. In: IEEE International Conference on Intelligent Robots and Systems, Chicago, IL, pp. 709–715 (2014)
17. Heess, N., Hunt, J.J., Lillicrap, T.P., Silver, D.: Memory-based control with recurrent neural networks. arXiv preprint (2015). arXiv:1512.04455
18. Mnih, V., Badia, A.P., Lillicrap, T.P., et al.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937 (2016)
19. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint (2014). arXiv:1412.6980
20. van den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Pradalier, C., Siegwart, R., Hirzinger, G. (eds.) Robotics Research, vol. 70, pp. 3–19. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-19457-3_1
21. Helbing, D., Molnr, P.: Social force model for pedestrian dynamics. Phys. Rev. E. **51**(5), 4282–4286 (1995)
22. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning. OSDI **16**, 265–283 (2016)