Learning-Based Socially Aware Robot Crowd Navigation

Shuijing Liu

Department of Electrical and Computer Engineering

University of Illinois at Urbana Champaign

3/26/2020

Introduction

- With the rapid growth of machine intelligence, an increasing number of robots are entering everyday applications [1]



- When mobile robots share the living space with people, they need to navigate in places crowded with moving people and other agents, and they must achieve and balance the following requirements:
  - Safety
  - Speed
  - Behaviors are consistent with social norms

Problem scope

- We focus on solving the decentralized, non-communicating crowd navigation problem
  - In real-life navigation scenarios, it is difficult to establish explicit communication channels between the robot and pedestrians
  - We can only control the robot but not other agents in the same environment
  - We don’t have full observability of other agents, such as their intended goals
- The environment is not static or stationary
  - the robot’s actions and humans’ actions have mutual influence on each other
  - We need to train the robot so that it can make appropriate decisions in this dynamic environment

MDP formulation

- We formulate the robot's decision making in a crowd navigation environment as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:
  - State space $\mathcal{S}$: $s_t = [s_{robot}, s_{humans}]$, where
    - s_{robot} contains both the observable state (position, velocity, radius) and the hidden state (goal position, preferred speed, heading angle) of the robot
    - s_{humans} contains only the observable state of all humans in the environment
  - Action space $\mathcal{A}$: $a_t = [v_x, v_y]$ is the robot's horizontal and vertical speed
  - State transition probabilities $P(s_{t+1}|s_t, a_t)$
  - Reward function R (Appendix 1):
    - Award the robot if it reaches the goal
    - Penalize the robot if collision happens, or if it gets too close to a human
    - Equals to 0 otherwise
  - Discount factor γ

MDP formulation

- At each timestep t, the agent takes an action $a_t \in \mathcal{A}$ in its state $s_t \in \mathcal{S}$ according to its policy $\pi(a_t|s_t)$
- The goal of the agent is to maximize the total accumulated return:

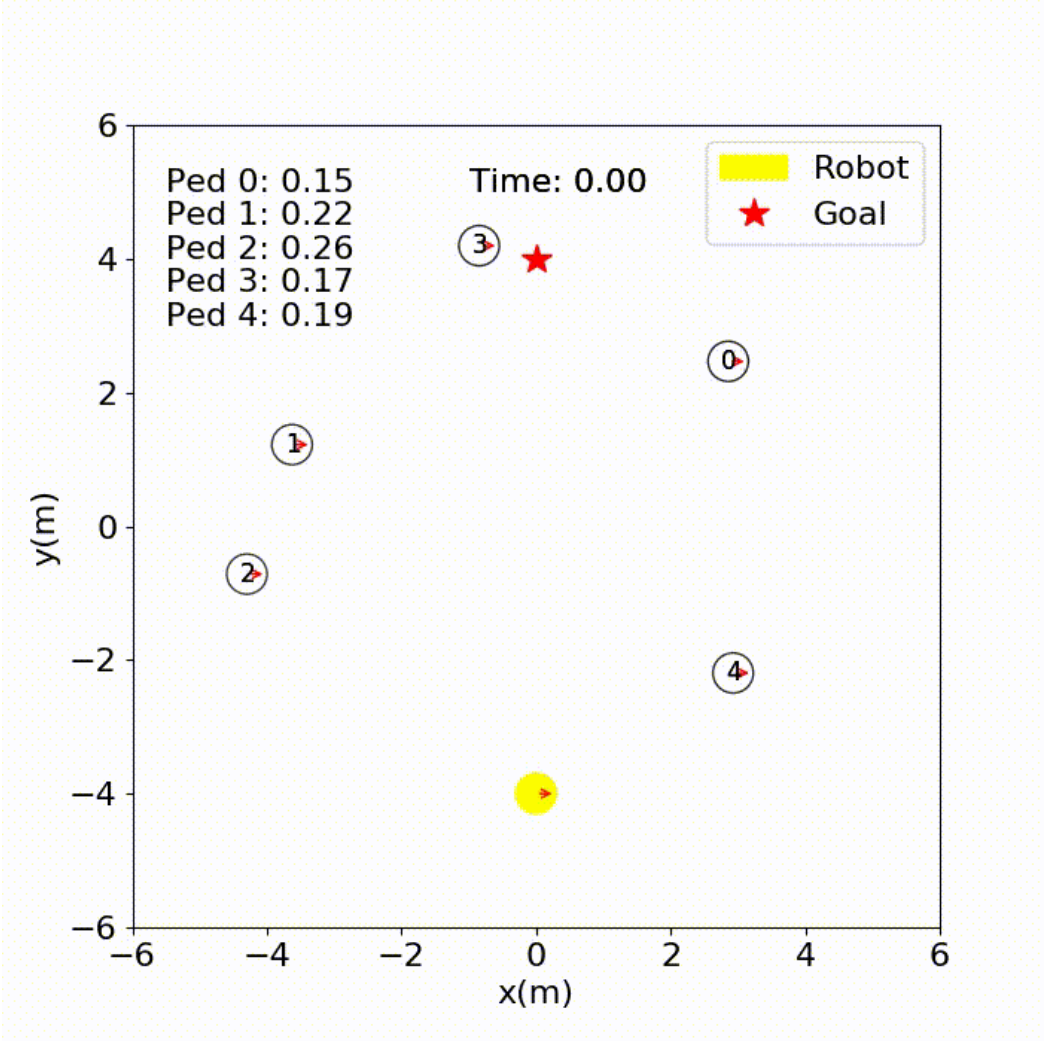
$$R_{total} = \sum_{t=0}^{T} \gamma^t r_t$$

- The value of state s under policy π is the expected return if the agent follows π from state s:

$$V^{\pi}(s) = \mathbb{E}[R_{total}|s_t = s]$$

Environment

- The environment contains one ego-robot (yellow) and several people (numbered from 0 to 4)
- We model all agents as circles in a 2D plane to save computation
- The goal of the robot is to navigate to a predefined destination as quickly as possible without colliding with other people
- We use a reaction-based navigation policy called ORCA to control the people



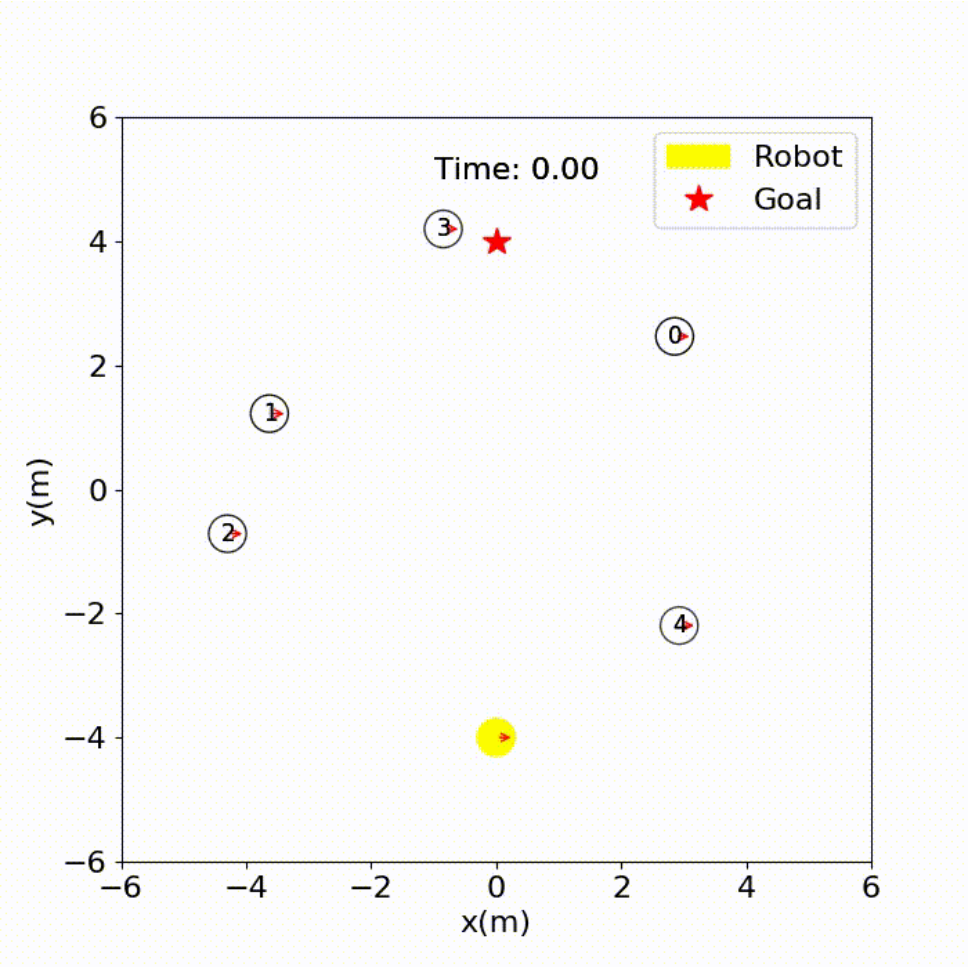
Previous works

- (Traditional method) Reciprocal n-body collision avoidance (ORCA) [2] is a reaction-based algorithm that
  - Models other agents as velocity obstacles
  - Then plans a new velocity for the ego-agent within the set of permitted velocity
- (Learning-based method) A series of algorithms started from CADRL [3]-[7] have used neural networks to approximate the value function
  - They first initialize the network using imitation learning with ORCA as the demonstration policy
  - Then they fine-tune the network using Deep Q-Learning, except that they fit a value network instead of Q network
  - They retrieve policy from the trained value network using one-step lookahead

(More details in Appendix 2)

Problems of previous works

- These methods assume that the robot knows the dynamics of all agents, which is unrealistic in real life
- Since the robot policy only depends on the current joint state, it sometimes exhibits shortsighted behaviors
  - For example, the robot always avoids collision in the last minute, which is dangerous and may cause discomfort to humans
- These methods need precise measurements of state parameters (agents' position, velocity, etc), which is expensive and slow

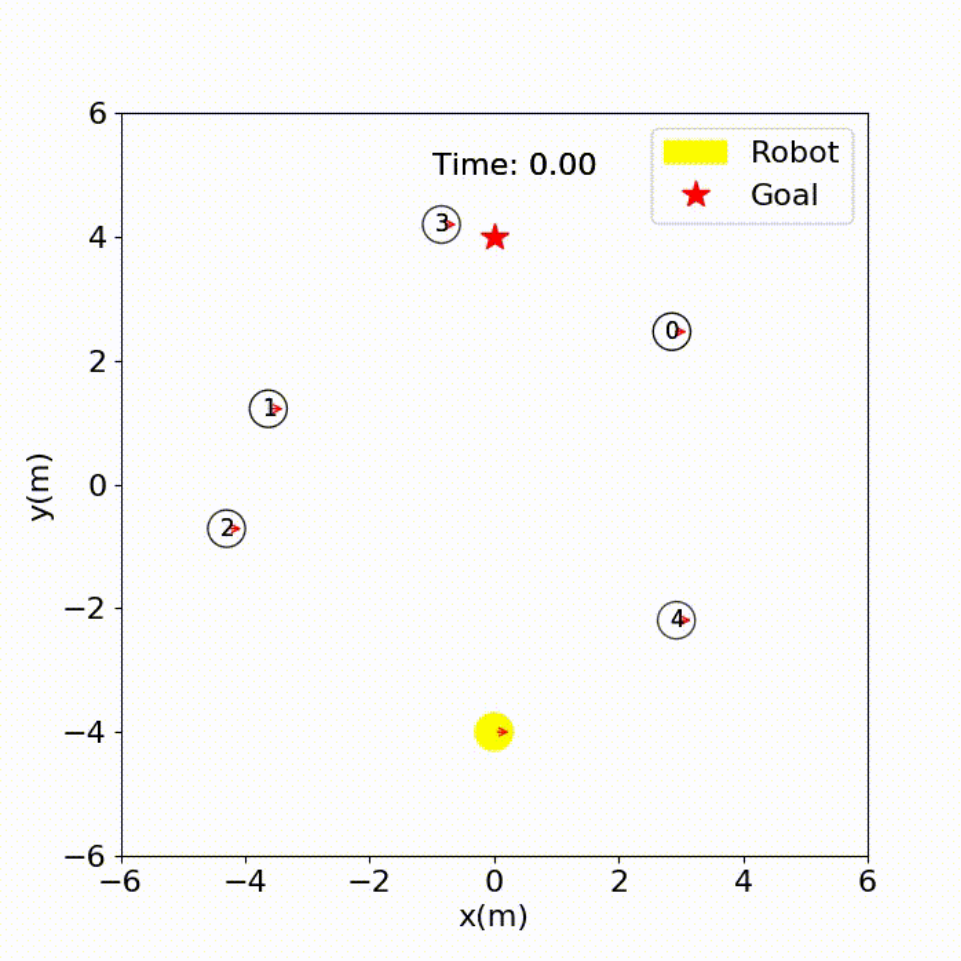


Our contributions

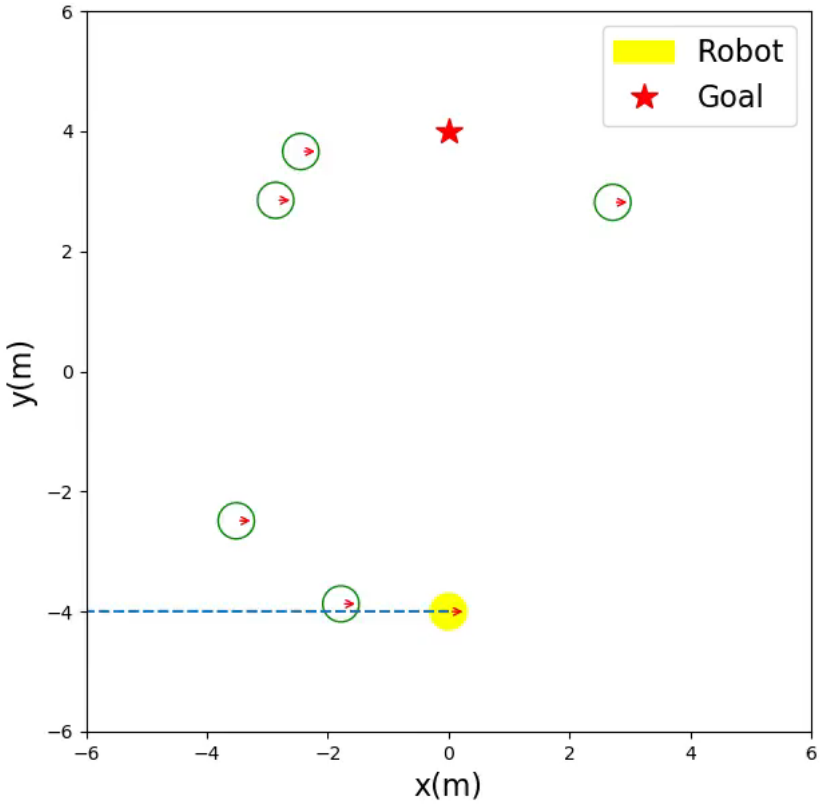
- To remove the assumption about agents' dynamics, we propose to use direct policy search instead of planning by value function
  - We use a model-free policy gradient algorithm, Proximal Policy Optimization (PPO) to learn a navigation policy [8]
- To encourage actions that benefit in the long term, we use Long Short-term Memory (LSTM) network to incorporate past states into the robot's decision maker [9]
  - LSTM makes the robot more proactive and be able to produce socially-aware behaviors

Simulation results

In our model, the robot is slower, but safer and more socially aware



CADRL



Ours

Future work

In next step, we are continuing this work in the following ways:

- Make our pipeline work under noisy state measurements or even with less state information to reduce the cost and improve real-time efficiency
- Add more heterogeneous human behaviors to our environment
- Transfer the simulated policy to a real TurtleBot2

References

[1] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, “Human-aware robot navigation: A survey,” Robotics and Autonomous Systems, vol. 61,no. 12, pp. 1726–1743, 2013.

[2] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, “Reciprocal n-body collision avoidance,” in Robotics research. Springer, 2011,pp. 3–19.

[3] Helbing, Dirk, and Peter Molnar. "Social force model for pedestrian dynamics." *Physical review E* 51.5 (1995): 4282.

[3] . F. Chen, M. Liu, M. Everett, and J. P. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017, pp. 285–292.

[4] Y. F. Chen, M. Everett, M. Liu, and J. P. How, “Socially aware motion planning with deep reinforcement learning,” in2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).IEEE, 2017, pp. 1343–1350.

[5] M. Everett, Y. F. Chen, and J. P. How, “Motion planning among dynamic, decision-making agents with deep reinforcement learning,”in2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, pp. 3052–3059.

[6] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, “Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning,” in2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 6015–6022.

[7] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, “Relational graph learning for crowd navigation,”arXiv preprint arXiv:1909.13165,2019.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O.Klimov,“Proximal policy optimization algorithms,” arXiv preprint arXiv:1707.06347, 2017.

[9] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997

Appendices

Reward function

- The reward function awards the robot for reaching its goal, and penalizes the robot for colliding with humans or getting too close to humans:

$$r(s_t, a_t) = \begin{cases} -0.25, & \text{if } d_{min} < 0 \\ -0.1 - 0.01 \times d_{min}, & \text{else if } 0 < d_{min} < 0.2 \\ 1, & \text{else if } p_x = g_y \text{ and } p_y = g_y \\ 0, & \text{otherwise.} \end{cases}$$

where d_{min} is the minimum separation distance between the robot and any human at time t

More literature review

- Communication-based methods
  - Centralized path planning
  - Distributed algorithms based on message-passing

  But establishing communication channels in crowd navigation is impossible
- Reaction-based methods
  - RVO/ORCA

  “Freezing robot problem”/shortsighted or unnatural behaviors
- Trajectory-based methods
  - Predict other agent’s intent and then plan a safe path

  But predictions are computationally expensive, and stochasticity of other agents makes the prediction harder

CADRL algorithm

- Assume $P(\cdot\,|s_t, a_t)$ is known, then if we have the optimal value function of each state:

$$V^*(s_t) = \mathbb{E}[\sum_{\tau=t}^{T} \gamma^t \cdot R\big(s_\tau, \pi^*(s_\tau)\big)|s_0 = s]$$

- We will be able to retrieve the optimal policy from the value function:

$$\pi^*(s_t) = argmax_a \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s_{\tau+1}|s_\tau, a)V^*(s')\right]$$

Our network architecture