A Review of Robot Crowd Navigation Algorithms

Shuijing Liu

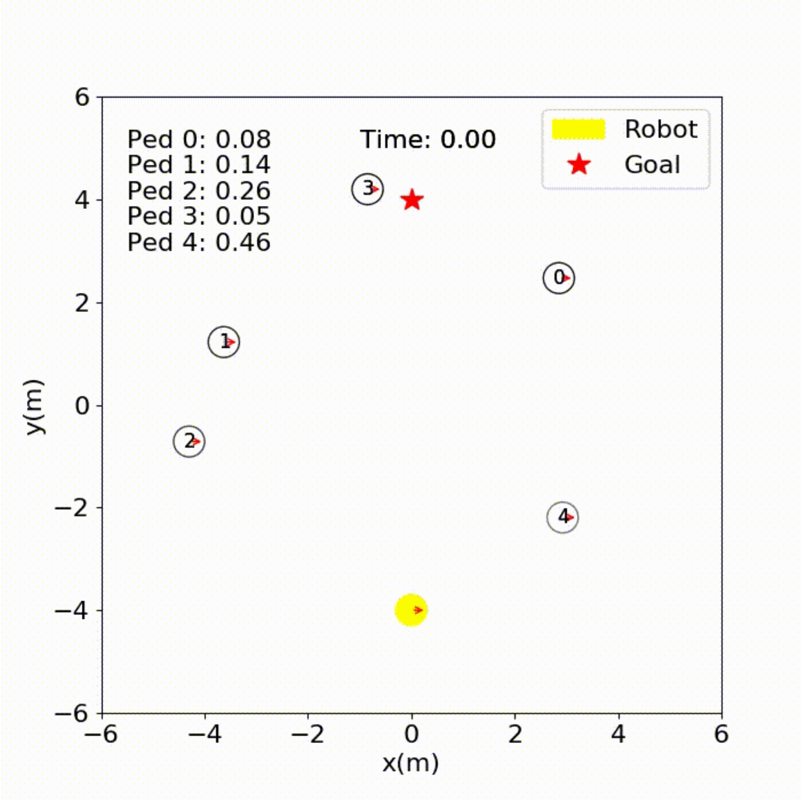
11/22/2019

Problem to solve

- In a crowded environment where other moving agents are present, the robot must navigate to a goal position **as quickly as possible without colliding with other agents**.

Problem formulation

- There is a set of non-communicating n agents in a 2D plane environment in $\mathbb{R}^2$
- Assume that each agent is circle-shaped and moves in this 2D plane
- State space/Observation space:
  - Observable state $s_o = \{p_x, p_y, v_x, v_y, r\}$
  - Hidden state $s_h = \{g_x, g_y, v_{pref}, \theta\}$
- Task: at each time step, each agent independently selects a new velocity (v'_x, v'_y) such that it will not collide with other agents AND this new velocity is as close to its v_{pref} as possible

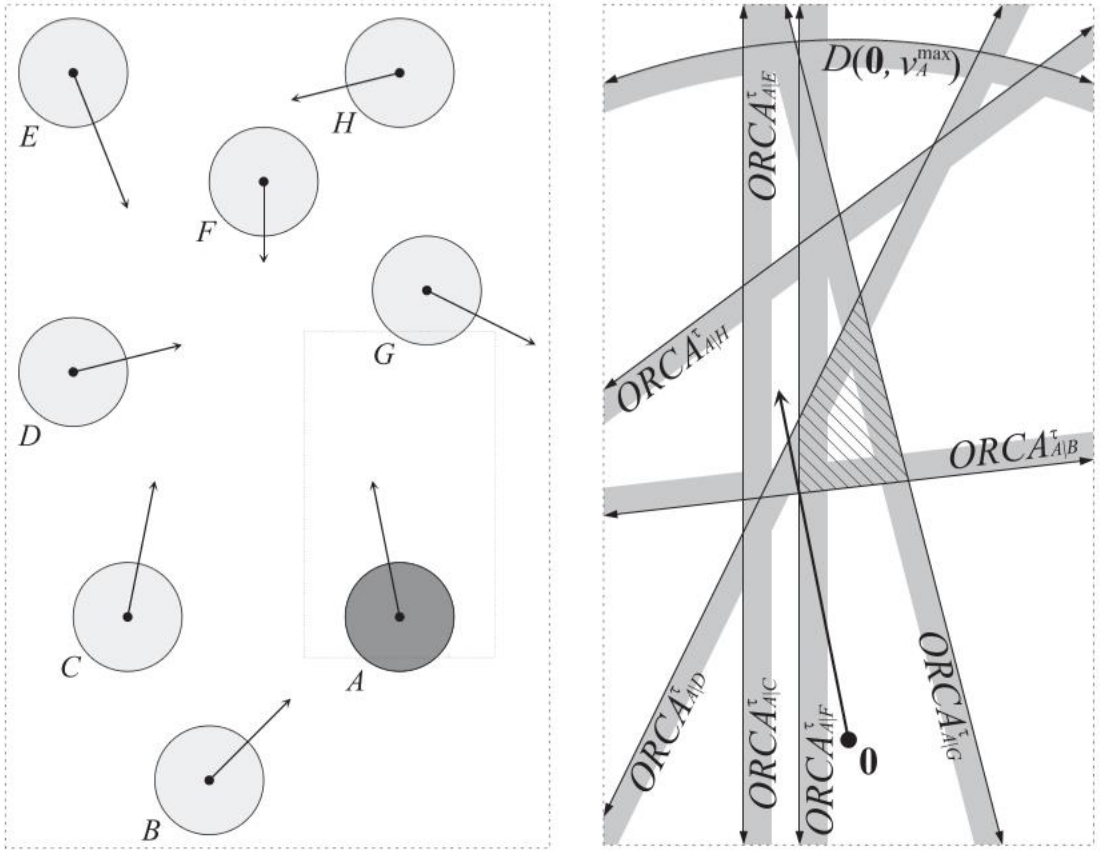


Early attempts

ORCA

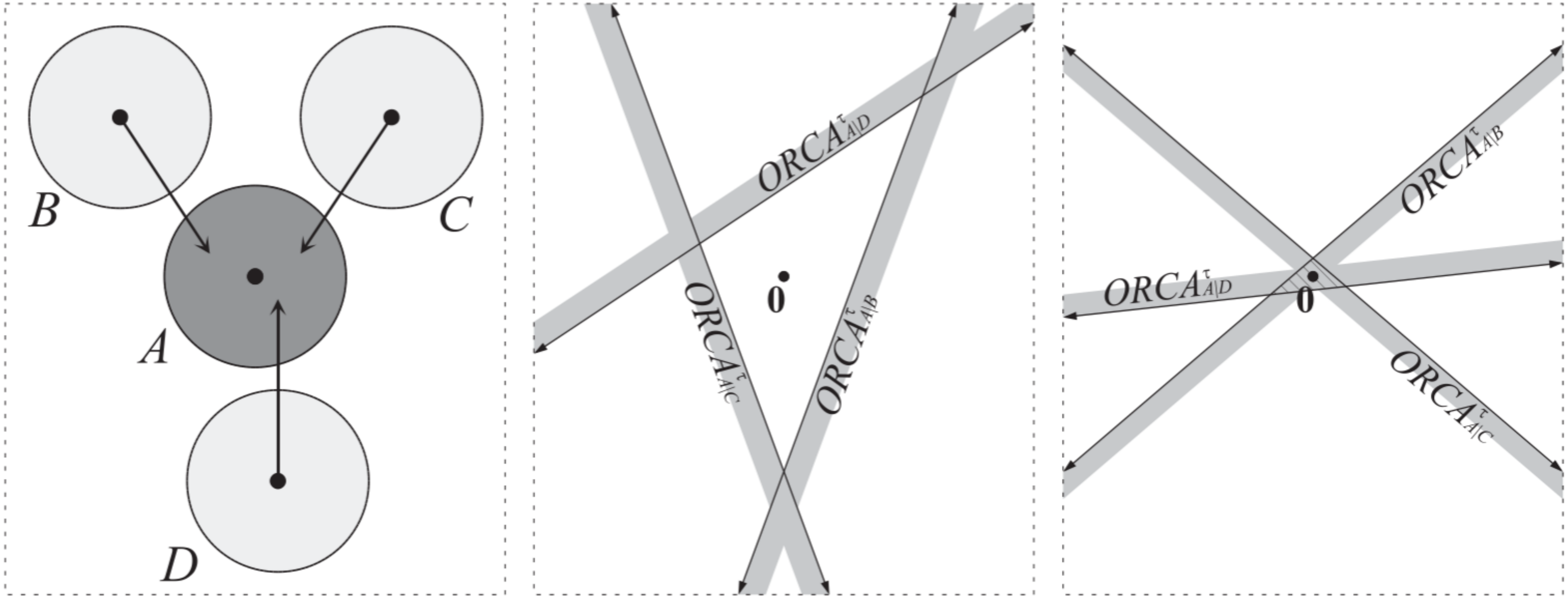
- For each pair of agent i and agent j, calculate the set of velocities such that
  - Agent i and j are collision free for at least τ time steps
  - The calculated velocities are close to their preferred velocities -- v_i^{opt} and v_j^{opt}

  which are denoted as $ORCA_{A|B}^{\tau}$ and $ORCA_{B|A}^{\tau}$
- Once we have all ORCAs of each pair of agents, we can find the permitted region of velocities for any agent



Van Den Berg, J., Guy, S. J., Lin, M., & Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research* (pp. 3-19). Springer, Berlin, Heidelberg.

ORCA



$$\tau = 2$$
$$v_A^{opt} = v_B^{opt} = v_B^{opt} = v_*$$

$$\tau = 2$$
$$v_A^{opt} = v_B^{opt} = v_B^{opt} = 0$$

Deep reinforcement learning based methods

CADRL

- The problem setting is the same as ORCA
- Objective function:

$$
\begin{aligned}
\underset{\pi(\mathbf{s}, \tilde{\mathbf{s}}^o)}{\operatorname{argmin}} \quad & \mathbb{E}\left[t_g | \mathbf{s}_0, \tilde{\mathbf{s}}_0^o, \pi, \tilde{\pi}\right] \\
s.t. \quad & ||\mathbf{p}_t - \tilde{\mathbf{p}}_t||_2 \geq r + \tilde{r} \qquad \forall t \longrightarrow \text{Collision avoidance constraint} \\
& \mathbf{p}_{t_g} = \mathbf{p}_g \longrightarrow \text{Goal constraint} \\
& \mathbf{p}_t = \mathbf{p}_{t-1} + \Delta t \cdot \pi(\mathbf{s}_{0:t}, \tilde{\mathbf{s}}_{0:t}^o) \\
& \tilde{\mathbf{p}}_t = \tilde{\mathbf{p}}_{t-1} + \Delta t \cdot \tilde{\pi}(\tilde{\mathbf{s}}_{0:t}, \mathbf{s}_{0:t}^o)
\end{aligned}
$$

The agents' kinematics

- Can be generalized to multiagent $(n > 2)$ scenarios by replacing the other agent's state $\tilde{s}_t^o$ with all other agents' states $\tilde{S}_t^o = [\tilde{s}_{1,t}^o, \dots, \tilde{s}_{n,t}^o]$
- It assumes that each agent would follow the same policy (reciprocity)

Chen, Y. F., Liu, M., Everett, M., & How, J. P. (2017, May). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In 2017 IEEE international conference on robotics and automation (ICRA) (pp. 285-292). IEEE.

CADRL

- Reward function:

$$R(\mathbf{s}^{jn}, \mathbf{a}) = \begin{cases} -0.25 & \text{if} \quad d_{min} < 0 \\ -0.1 - d_{min}/2 & \text{else if} \quad d_{min} < 0.2 \\ 1 & \text{else if} \quad \mathbf{p} = \mathbf{p}_g \\ 0 & \text{o.w.} \end{cases}$$

- (Optimal) value function:

$$V^*(\mathbf{s}_0^{jn}) = \sum_{t=0}^{T} \gamma^{t \cdot v_{pref}} R(\mathbf{s}_t^{jn}, \pi^*(\mathbf{s}_t^{jn}))$$

- (Optimal) policy:

$$\pi^*(\mathbf{s}_0^{jn}) = \underset{\mathbf{a}}{\operatorname{argmax}} R(\mathbf{s}_0, \mathbf{a}) + \gamma^{\Delta t \cdot v_{pref}} \int_{\mathbf{s}_1^{jn}} P(\mathbf{s}_0^{jn}, \mathbf{s}_1^{jn} | \mathbf{a}) V^*(\mathbf{s}_1^{jn}) d\mathbf{s}_1^{jn}$$

CADRL

- Algorithm:

Algorithm 1: CADRL (Coll. Avoidance with Deep RL)

Input: value network $\mathbf{V}(\cdot;\mathbf{w})$
Output: trajectory $\mathbf{s}_{0:t_f}$
while *not reached goal* **do**
  update t, receive new measurements $\mathbf{s}_t$, $\tilde{\mathbf{s}}_t^o$
  $\hat{\tilde{\mathbf{v}}}_t \leftarrow \text{filter}(\tilde{\mathbf{v}}_{0:t})$
  $\hat{\tilde{\mathbf{s}}}_{t+1}^o \leftarrow \text{propagate}(\tilde{\mathbf{s}}_t^o, \Delta t \cdot \hat{\tilde{\mathbf{v}}}_t)$
  $A \leftarrow \text{sampleActions}()$
  $\mathbf{a}_t \leftarrow \text{argmax}_{\mathbf{a}_t \in A} R(\mathbf{s}_t^{jn}, \mathbf{a}_t) + \bar{\gamma} V(\hat{\mathbf{s}}_{t+1}, \hat{\tilde{\mathbf{s}}}_{t+1}^o)$
   where $\bar{\gamma} \leftarrow \gamma^{\Delta t \cdot v_{pref}}$, $\hat{\mathbf{s}}_{t+1} \leftarrow \text{propagate}(\mathbf{s}_t, \Delta t \cdot \mathbf{a}_t)$
return $\mathbf{s}_{0:t_f}$

Use current value network/policy to sample a trajectory with length = t_f

Algorithm 2: Deep V-learning

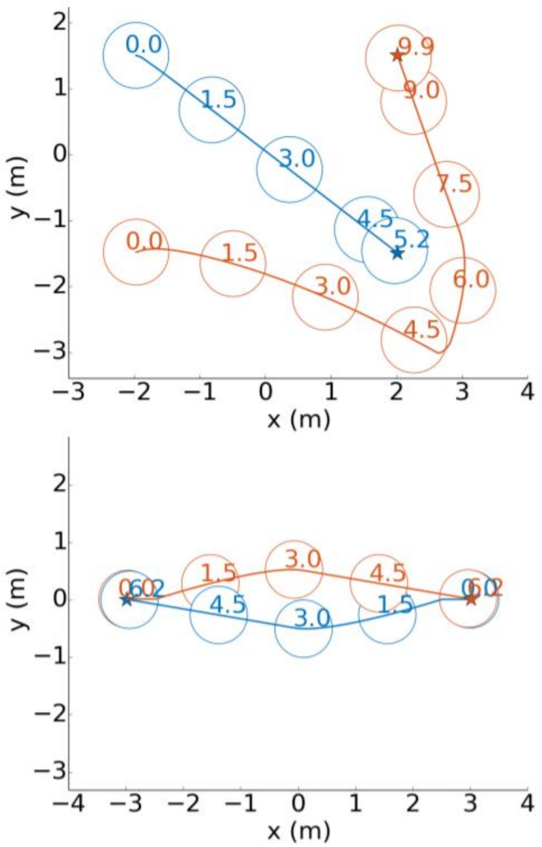
Input: trajectory training set D
Output: value network $V(\cdot;\mathbf{w})$
$V(\cdot;\mathbf{w}) \leftarrow \text{train\_nn}(D)$ //step 1: initialization
duplicate value net $V' \leftarrow V$ //step 2: RL
initialize experience set $E \leftarrow D$
for *episode*$=1,\ldots,N_{eps}$ **do**
  for *m times* **do**
    $\mathbf{s}_0, \tilde{\mathbf{s}}_0 \leftarrow \text{randomTestcase}()$
    $\boxed{\mathbf{s}_{0:t_f} \leftarrow \text{CADRL}(V),\ \tilde{\mathbf{s}}_{0:\tilde{t}_f} \leftarrow \text{CADRL}(V)}$
    $\mathbf{y}_{0:T},\ \tilde{\mathbf{y}}_{0:\tilde{t}_f} \leftarrow \text{findValues}(V', \mathbf{s}_{0:t_f}, \tilde{\mathbf{s}}_{0:\tilde{t}_f})$
    $E \leftarrow \text{assimilate}\Big(E, (\mathbf{y}, \mathbf{s}^{jn})_{0:t_f}, (\tilde{\mathbf{y}}, \tilde{\mathbf{s}}^{jn})_{0:\tilde{t}_f}\Big)$
  $e \leftarrow \text{randSubset}(E)$
  $\mathbf{w} \leftarrow \text{backprop}(e)$
  for *every C episodes* **do**
    $\text{Evaluate}(V)$, $V' \leftarrow V$
return V

1. Initialize the value network by ORCA

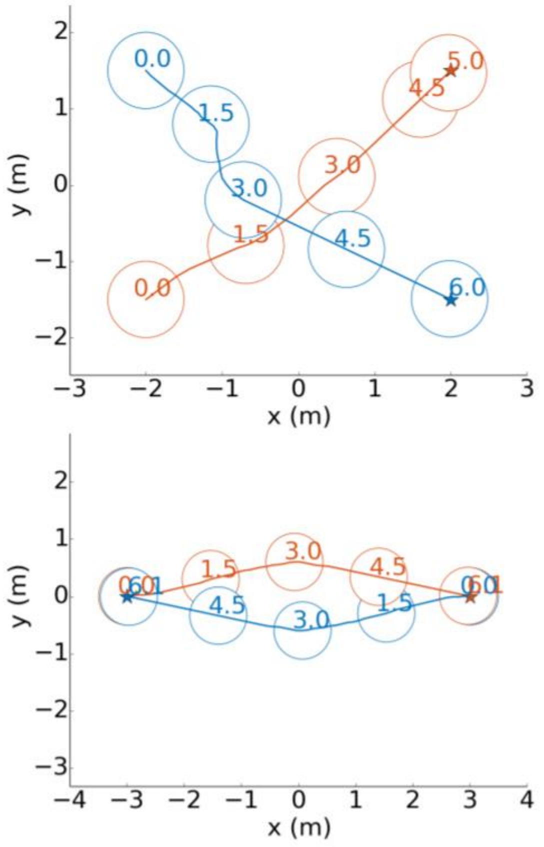
2. Train the value network using DQN with experience replay

CADRL

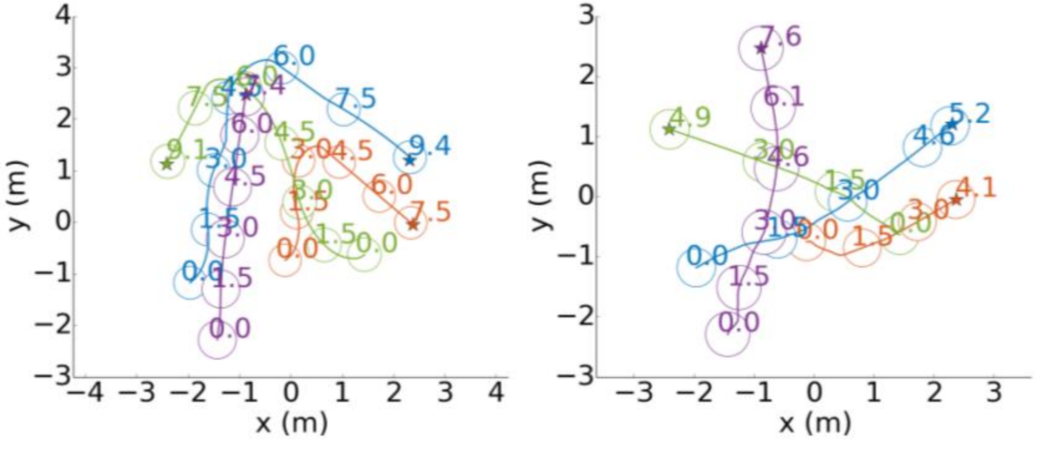
- Results



ORCA



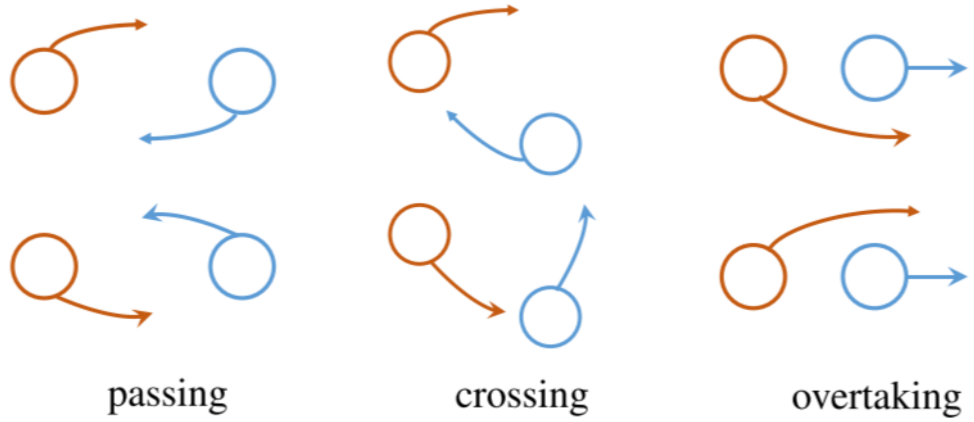
CADRL after 1000 training episodes



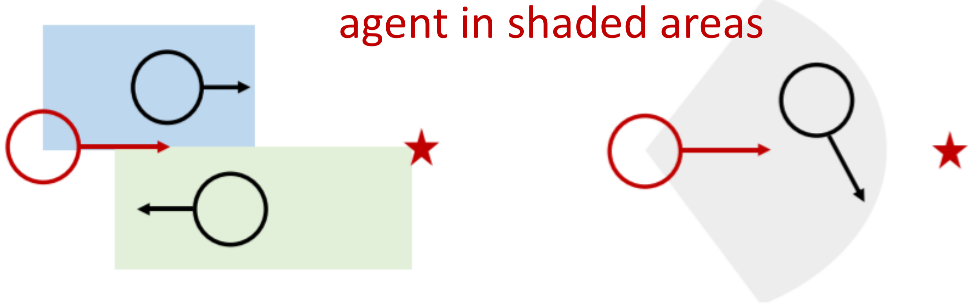
(a) ORCA (b) CADRL

Socially Aware CADRL

- Problems with CADRL:
  - The cooperative behaviors are not consistent with human interpretation
  - The cooperative behaviors cannot be controlled & depend on initialization of the value network
- Socially-aware CADRL introduces left-handed rules (left top) and right-handed rules (left bottom), and enforces these social norms by adding a R_{norm} to the reward function (right)



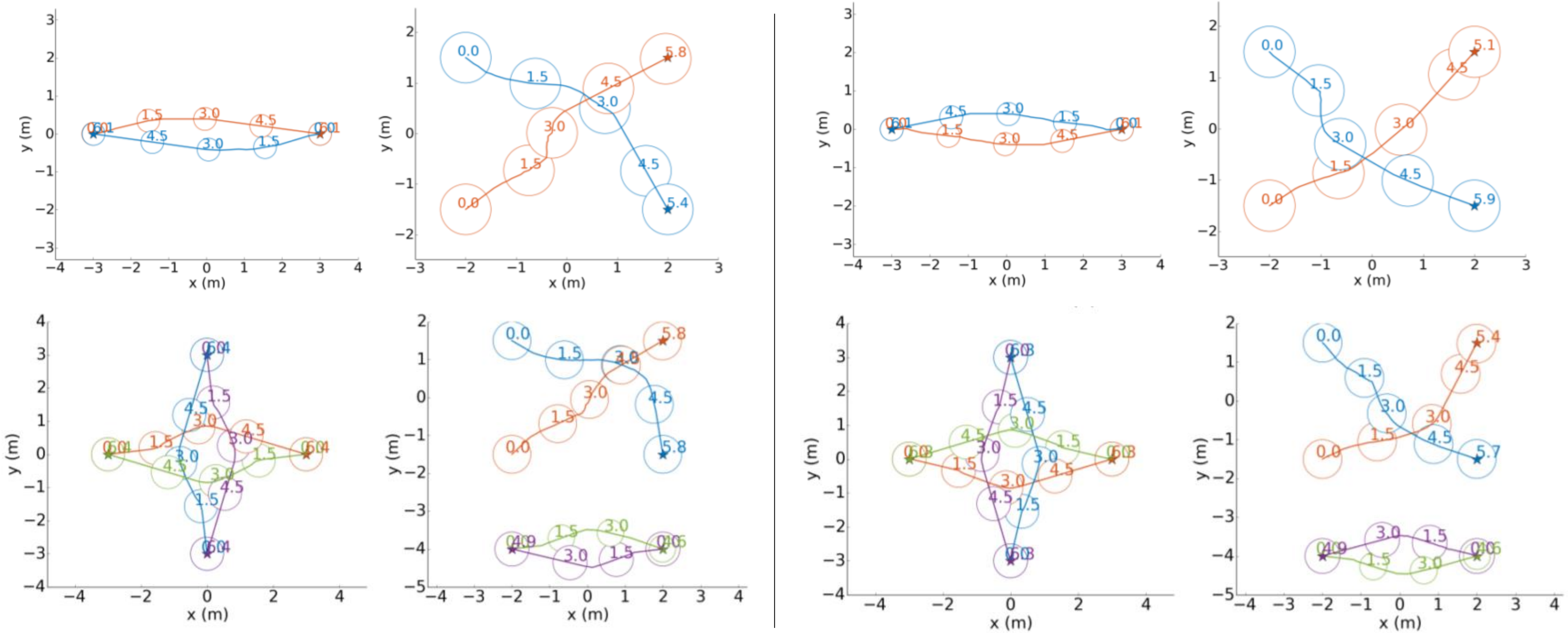
In right-handed rule, the red agent will be penalized if there's another agent in shaded areas



Chen, Y. F., Everett, M., Liu, M., & How, J. P. (2017, September). Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1343-1350). IEEE.

Socially Aware CADRL

- Results



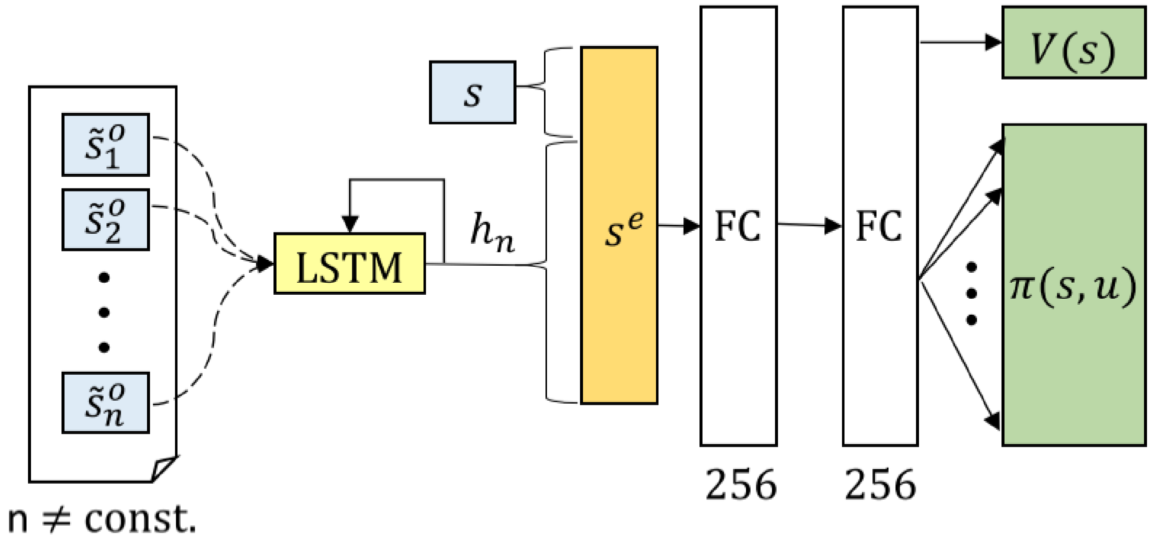
Left-handed rule

Right-handed rule

Socially Aware CADRL

LSTM CADRL

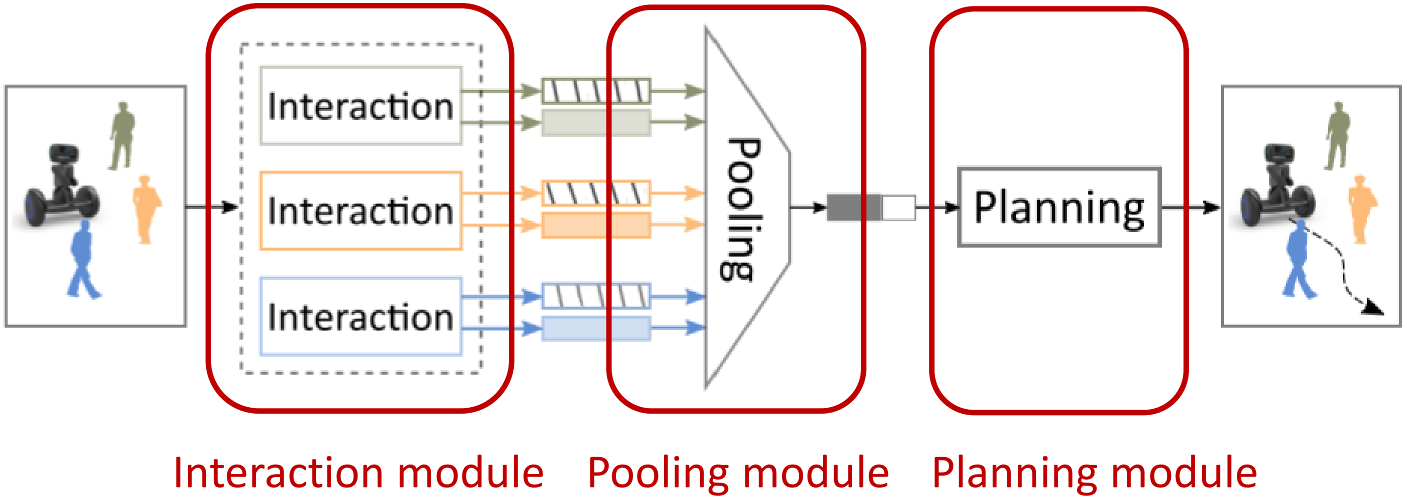
- Both CADRL and SA-CADRL assumes a fixed number of agents because the value network requires a fixed-size input
- LSTM CADRL utilizes LSTM to compress a variable number of $\tilde{s}^o$ vectors as a sequence



Everett, M., Chen, Y. F., & How, J. P. (2018, October). Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3052-3059). IEEE.

Crowd-aware robot navigation with attention

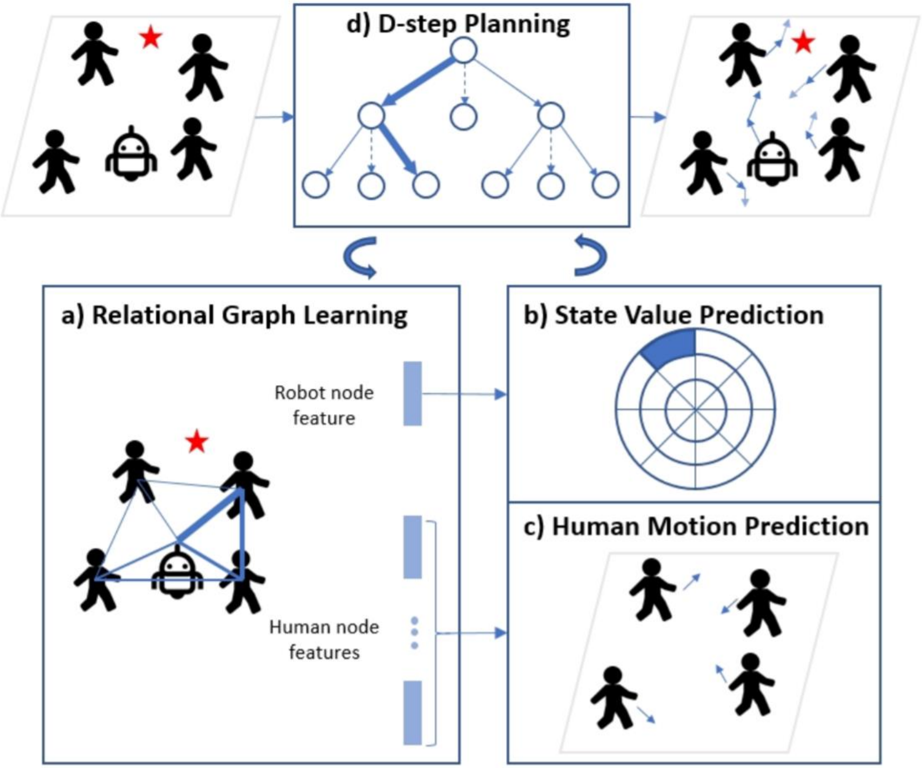
- Previous methods made simplified assumptions about the interaction between agents, and only focused on one-way interactions from humans to the robot
- This paper models the relative importance and the collective impact of neighboring agents for robot crowd navigation



Chen, C., Liu, Y., Kreiss, S., & Alahi, A. (2019, May). Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 6015-6022). IEEE.

Relational graph learning for crowd navigation

- Compared with the last paper, the Relational graph learning method
  - Explicitly model pairwise interactions among all agents in graph learning
  - Instead of assuming linear state transitions, it uses model-based RL to predict next states
  - Uses Monte-Carlo Tree Search (MCTS) with depth = d to plan the robot's trajectory



Chen, C., Hu, S., Nikdel, P., Mori, G., & Savva, M. (2019). Relational Graph Learning for Crowd Navigation. *arXiv preprint arXiv:1909.13165*.

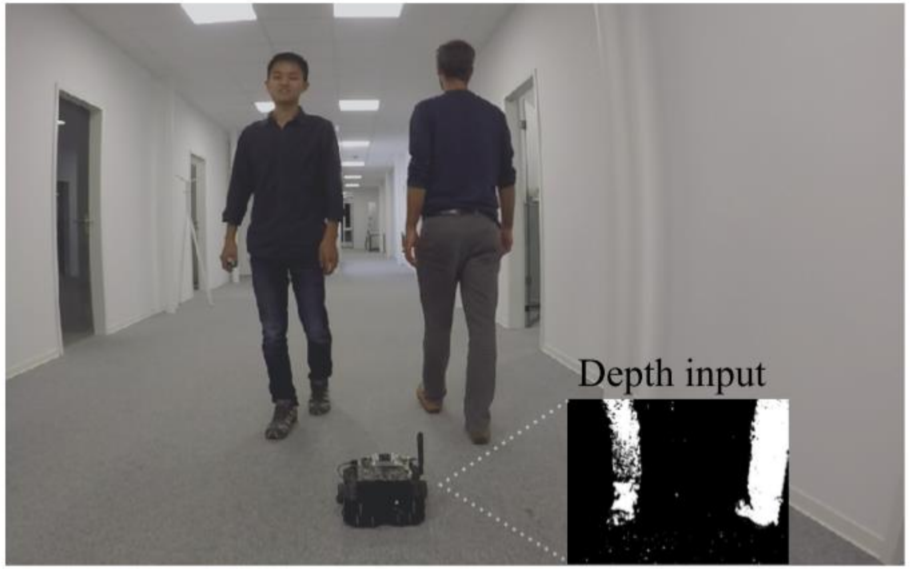
Extensions to partial observability

Problems with the algorithms so far

- All previous papers (from ORCA to Relational graph learning) have assumed full state observability
  - i.e. each agent knows the observable state ($s_o = \{p_x, p_y, v_x, v_y, r\}$) of all other agents
- These state information is measured by several Real-sense cameras and LiDAR, which is expensive
- Can we relax this full state observability assumption to achieve lower-cost crowd navigation?

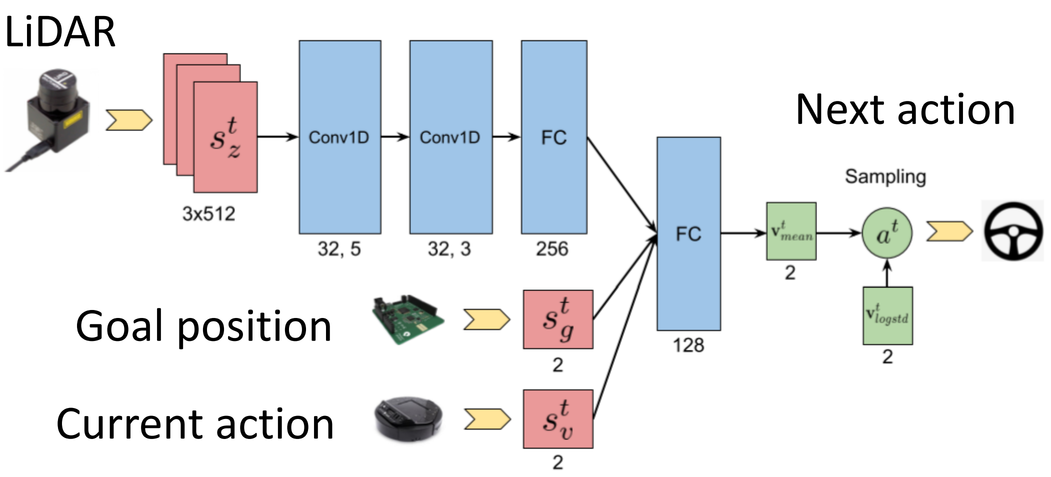
Using depth image

- Input of the robot: only a depth sensor with limited field of view
  - Sensing range is 3.5m, vertical sensing angle = $\pm 35°$
- Use generative adversarial imitation learning (GAIL) to train the policy



Tai, L., Zhang, J., Liu, M., & Burgard, W. (2018, May). Socially compliant navigation through raw depth inputs with generative adversarial imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1111-1117). IEEE.

Using LiDAR



Fan, T., Cheng, X., Pan, J., Monacha, D., & Yang, R. (2018). Crowdmove: Autonomous mapless navigation in crowded scenarios. *arXiv preprint arXiv:1807.07870*.

That’s it! Thank you for listening!