# Learning Social Heuristics for Human-Aware Path Planning

Andrea Eirale[1], Matteo Leonetti[2], and Marcello Chiaberge[1]

*Abstract*— **Social robotic navigation has been at the center of numerous studies in recent years. Most of the research has focused on driving the robotic agent along obstacle-free trajectories, respecting social distances from humans, and predicting their movements to optimize navigation. However, in order to really be socially accepted, the robots must be able to attain certain social norms that cannot arise from conventional navigation, but require a dedicated learning process. We propose Heuristic Planning with Learned Social Value (HPLSV), a method to learn a value function encapsulating the cost of social navigation, and use it as an additional heuristic in heuristic-search path planning. In this preliminary work, we apply the methodology to the common social scenario of joining a queue of people, with the intention of generalizing to further human activities.**

## I. Introduction

In the last few years, the proliferation of robotic platforms in society has inspired a significant amount of research in the field of service robotics. Robots can provide support to humans in a large number of specific tasks, such as helping the elderly [1], [2], [3], giving classes to students [4], [5], giving guided tours [6], [7], [8] and many more [9], [10].

In all these cases, how the robot interacts with people and how human beings perceive it is often fundamental for the successful completion of the service task [11]. Thus, traditional autonomous navigation, focused on avoiding obstacles while reaching the goal, is often unsuitable. In addition to conventional navigation metrics, like the distance to the goal or the presence of static obstacles, robots can exploit navigation behaviors that consider social factors. Introducing social information can lead to navigation policies able to access a trade-off between performance and social acceptability. The robot should not only respect others' personal spaces, but also comply with specific social norms. These norms include planning aspects, such as navigating through cluttered and crowded environments, and behavioural aspects, namely the adaptation to social signals which can be inferred from human actions [12].

We propose a novel method to augment classical path planning with socially acceptable behaviours learned through Reinforcement Learning (RL), at no additional planning cost. We train a social agent able to recognize specific social contexts; as a proof of concept, for this paper, we consider the problem of following a queue. The learned value function encapsulates the complexity of the social behaviour and depends on all necessary external factors, such as the position and activity of other people in the environment, so that the path planner does not have to take them into account. The learned value function is combined with the heuristic used by A*, which, with no change to the planning algorithm, produces socially acceptable trajectories.

[1]Department of Electronics and Telecommunications (DET), Politecnico di Torino, Torino, Italy [andrea.eirale@polito.it, marcello.chiaberge@polito.it]

[2]Department of Informatics, King's College London, London, UK matteo.leonetti@kcl.ac.uk

## II. Related Work

The problem of designing a behavioral policy able to consider social factors has been studied for more than twenty years. Following the RHINO [13] and MINERVA [14] deployments as tour guides in museums, where people are treated like dynamic, non-responsive obstacles, researchers focused on autonomous navigation systems able to distinguish humans from inanimate objects.

An important paradigm in this area involves planning around estimates of future human motion, which allows the agent to plan an optimal trajectory avoiding collisions with people and obstacles, and maintaining at any time a social distance from humans. Human motion prediction is usually achieved with deep neural networks, like generative adversarial networks [15], convolutional neural networks [16], [17] and attention transformers [18].

Another central body of work, which extends human motion prediction, consists in intention-aware navigation. Exploiting behaviour prediction models, it is possible not only to predict future movements of humans, but also to estimate their final goals. Bai er al. [19] model the uncertainty of human intent in the Partially Observable Markov Decision Process (POMDP) framework. Other works focused on navigation in dense human crowds avoiding blockage due to human activities [20], and cooperating with people through interacting Gaussian processes [21]. Mavrogiannis et al. employ geometric [22] and topological invariant [23] representations to model the coupling among trajectories of multiple navigating agents.

A different class of works have proposed Deep Reinforcement Learning for prediction in crowd navigation domains. Chen et al. [24] apply CADRL, a deep reinforcement learning framework for socially aware multiagent collision avoidance. Everett et al. [25] exploit an actor-critic variant to relax prior assumptions and learn policies and agent motion models at the same time. Furthermore, Tai et al. [26] train a generative adversarial imitation learning model on a dataset generated using the social force model [27]. Finally, Chen et al. [28] use attention-based reinforcement learning to produce interaction-aware collision avoidance behaviors.

All the work cited above focuses on people movement: their trajectory, their intention and their destination. However, a number of social norms are independent of movement,

but dependent on people's activity. For instance, it is considered rude to walk between two people talking to each other, even if they are just standing still. Our work is an initial step in the direction of integrating arbitrary scene features (not necessarily movement) into classical path planning.

Within queue following, the particular social navigation scenario we consider in this paper, Nakauchi et al. [29] designed a dedicated pipeline. In their work, they put great attention on how a line of people can be defined and then perceived by the autonomous agent. The navigation system generates a series of goals (depending on the number of people in the queue) until the back of the line is reached, and does not provide full path planning.

## III. Methodology

The main insight of our approach is that the social component, which depends on people in the environment and their activities, and the navigation component, which only depends on position and orientation of the robot, can be decoupled. The social component is trained off-line with RL, and then re-integrated in the planner's objective function. We start by introducing the notation regarding task modeling and heuristic-search planning.

### A. Notation

We model the navigation task as a Markov Decision Process (MDP) described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$ [30]. An agent starts its interaction with the environment in an initial state $s_0$, drawn from a distribution $p(s_0)$ and then at every time step $t$ selects an action $a_t \in \mathcal{A}$ from a state $s_t \in \mathcal{S}$ which results into a new state $s_{t+1}$, receiving a reward $r_t = R(s_t, a_t)$. A reinforcement learning process aims to optimize a parametric policy $\pi_\theta$, which defines the agent behavior. In the context of navigation learning, we model the task as an episodic MDP, with maximum time steps $T$. Hence, the agent is trained to maximize the cumulative expected reward $\mathbb{E}_{\tau\sim\pi} \sum_{t=0}^{T} \gamma^t r_t$ over each episode, where $\gamma \in [0, 1)$ is the discount factor.

We use heuristic-search planners [31], such as A*, which estimate the value of a given state $s$ as the sum of the cumulative cost up to $s$ and a heuristic value from $s$, minimizing the objective:

$$f(s) = g_n(s) + h_n(s), \tag{1}$$

where $n$ indicates that both functions are related to the navigation objective. We will modify this objective by adding a social component.

### B. Planning Objective

In addition to the usual navigation cost and heuristic (cf. Eq. 1), we want the planner to take also a *social* cost into account when computing the optimal path. A social cost would be non-zero, for instance, when a position would result in the robot cutting a queue, or driving between two people in a conversation. In this work, we assume the existence of such a social cost function $c_s(s)$, which, in general, is also a function of the goal. In principle, it is also possible to learn this cost function from experience, but for this preliminary work we assume is as known.

The social cost makes the problem effectively multi-objective: the robot has to both minimize travel distance and social cost. Since heuristic-search planners require a heuristic function in addition to the cost function, and the social cost is known, the rest of this methodology is devoted to the definition of a social heuristic.

We modify the objective function of Equation 1 to take both objectives into account, representing with the subscript $n$ the navigation component of the objective and with the subscript $s$ the social component of the objective:

$$f(s) = g_n(s) + h_n(s) + w(g_s(s) + h_s(s)), \tag{2}$$

where $w \geq 0$ is a parameter weighting the social cost over the navigation cost.

The function $g_s$ is obtained by accumulation during planning of the cost $c_s$, just like for its navigation counterpart. The function $h_s$ is defined in the following section.

### C. Social Heuristic

We propose to obtain the social heuristic function $h_s$ through RL.

**Environment** The agent is trained in a fully observable gridmap-like environment, with a given number of people whose position and orientation in the grid is known.

**State Space** The state representation embeds the necessary information about the goal and people, and it is ego-centric, so that it does not depend on the particular coordinates used in training. Different definitions are possible, depending on the social behaviour one intends to capture. For our queue following scenario we used:

- The distance $d_t^g$ and the angle $\Delta\theta_t^g$ of the goal from the agent.
- For every person in the environment, the distance $d_t^i$ and the angle $\Delta\theta_t^i$ of the $i$th person with respect to the agent.

The state space is, therefore, defined as:

$$S(t) = [d_t^g, \Delta\theta_t^g, d_t^1, \Delta\theta_t^1, ..., d_t^i, \Delta\theta_t^i]. \tag{3}$$

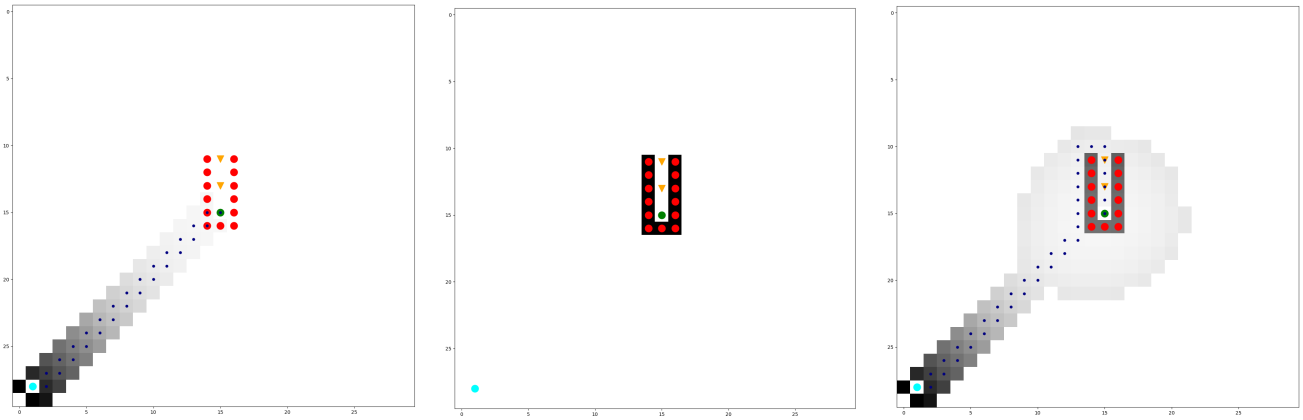
**Action Space** The agent can choose to go forward, backward, turn left, or right.

**Reward** We define the navigation reward $r_n$ as:

$$r_n = \begin{cases} k_g & \text{if goal is reached} \\ d_{t-1}^g - d_t^g - k_r & \text{otherwise,} \end{cases} \tag{4}$$

where $d_t^g$ is the distance of the goal from the agent, $d_{t-1}^g$ is the distance of the goal at the previous time step, $k_g$ is a large reward received when the goal is reached, and $k_r$ is a small value implemented to encourage convergence in as few steps as possible. The social reward is the opposite of the weighted social cost: $r_s = -wc_s$.

We train an RL agent to maximize $r_T = r_n + r_s$ with standard RL, and, in doing so, it learns a value function $Q_T$. At the same time, the agent learns a second value function $Q_s$, trained to estimate the expected cumulative unweighted social component of the reward. This social value function is



(a) *Traditional* $A^*$ (b) *Social Cost Function* (c) *Integration of* $A^*$ *with Social Cost Function*

Fig. 1: Results obtained from the demo discrete environment: cyan and green points are respectively the start and goal points, orange arrows represent people, small blue dots are the trajectory chosen by the planner, and red dots are the virtual obstacles. Black cells represent cells associated with a high cost, while the cost decrease as they turn to light grey. $(a)$ represents the path chosen by a traditional $A^*$ search algorithm, and (b) shows the costs added by the Social Cost Function $c_s$ extracted from $Q_s$. In contrast, (c) represents the path chosen by the $A^*$ algorithm integrated with the Social Cost Function $c_s$.

trained on reward $r_s$ only. Note that, during learning, actions are only selected so as to maximize $Q_T$. That is, the function $Q_s$ is learned entirely as a side effect of maximizing $Q_T$, and does not affect the behaviour policy. This is crucial because $Q_s$ does not encode the navigation goal. For instance, in following a queue, maximizing $Q_T$ leads the agent to wait at the end of the queue, while maximizing $Q_s$ would only lead the agent to stay away from people in order to avoid the potential cost of cutting the queue. So, while executing and learning the full task, the agent also stores a long-term estimate of the social cost for later use as a planning heuristic.

### D. Deployment

In the deployment phase, the value function $Q_T$ is discarded, since we are only interested in the social components of the objective function $g_s$ and $h_s$ as in Equation 2. These components are extracted from the social value function $Q_s$. In particular, the heuristic social component is computed as:

$$h_s(s) = \begin{cases} c_s(s) & \text{if } c_s(s) > k_{thresh} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$c_s(s) = 1 - Q_s(s, a_{s^- \to s}, \theta_s) \tag{6}$$

Where $s_t$ is the agent observation at time $t$, $a_{s^- \to s}$ is the action that brings the agent from the current state $s^-$ to the next state $s$, $\theta_s$ are the trained social wights, and $Q_s(s_t, a_{s^- \to s}, \theta_s)$ is the value predicted by the action-value function $Q_s$ when the action $a_{s^- \to s}$ is taken starting from the state $s^-$. Finally, $k_{thresh}$ is a confidence threshold: if the cost $c_s(s)$ exceeds this value, the agent is making a strong assumption on which direction (not) to take. This configuration allows the $A^*$ algorithm to manage the general path planning, while the social component adds its contribution only when the social scenario is recognized.

Moreover, the cumulative social cost $g_s(s)$ is defined as:

$$g_s(s) = g_s(s^-) + c_s(s) \tag{7}$$

## IV. Experiments and Results

To validate our methodology, we trained a HPLSV agent exploiting a customized version of the TF2RL library [32]. As a proof of concept, we consider the social scenario of a queue of people, where the agent has to enter the queue without cutting it. At first, the agent is trained in a demo gridmap environment, where people are perfectly aligned with the goal, and the distance between the goal and the first person, and between each person, has the same value. Then, a series of close-to-real environments are obtained from a Gazebo simulation, and are used to retrain the agent. In the next paragraphs, the results obtained in each of these environments are presented.

### A. Discrete environment

The first environment presents a 30x30 gridmap. The goal is placed in the center of this gridmap, and two other entities, representing people in line, are placed just above the goal. One cell separates the goal from the first person, and another cell is left between the two people. The three entities, the goal and the two people, are aligned on the x axis. We define virtual obstacles as a continuous box surrounding people and goal, which leaves just a passage to the goal at the end of the queue. If the agent hits the virtual obstacle, it is equivalent to cutting the queue, and it receives the corresponding social cost. During initial training episodes, the agent always starts from the same position, at the bottom left of the gridmap. Then, in order to enhance exploration, the starting point is changed after each episode, until the end of the training.

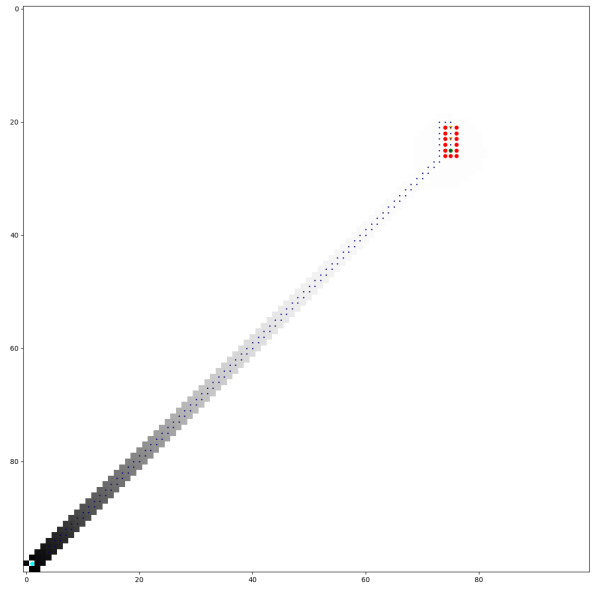


Fig. 2: Results obtained on a much bigger environment compared to the one used for training. The social cost function does not interfere with the traditional planner until the social scenario is reached.

For the considered scenario of a queue of people, the social reward function $r_s$ is defined as:

$$r_s = \begin{cases} -k_s & \text{if the virtual obstacle is crossed} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Where $k_s$ is a small value received every time the agent cross a virtual obstacle.

To test and validate the model, it is integrated with a classic $A^*$ planner, as explained at Section III-D. Figure 1a shows the path chosen by the $A^*$ planner without any other external contribution, with the related costs. In this case the planner just drives the agent towards the goal, cutting the line. Figure 1b shows the costs added by the social action-value function $Q_s$ extracted from the trained model. Finally, Figure 1c shows the path and the costs obtained by the $A^*$ planner integrated with our trained model. In contrast with the previous case, the planner is now able to recognize the social scenario and extract an optimal path to the goal which passes through the end of the queue, avoiding to cut it.

The ego-centric representation of the heuristic makes it independent of coordiantes, as shown in Figure 2, where a much larger 100x100 gridmap testing environment is employed. The social component of the objective does not activate (the expected social cost of actions is 0) until the agent gets near the goal. This allows the traditional planner to compute an optimal trajectory towards the goal without any other interference. Then, when the queue is reached, the costs introduced by the social contributions increase, avoiding to cut the line.

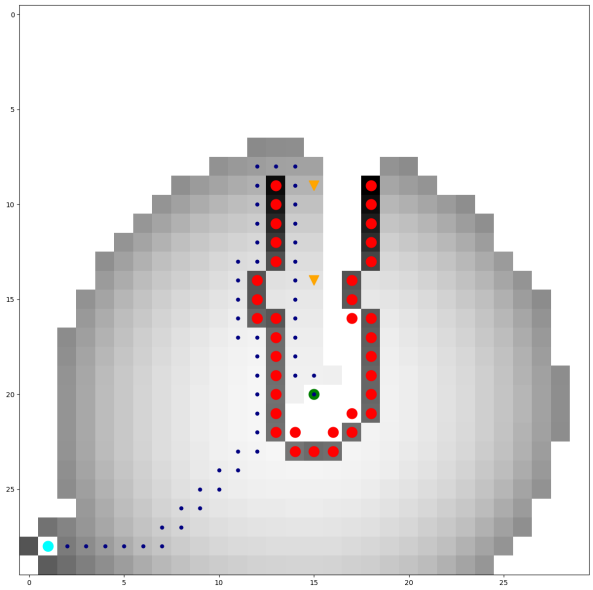


Fig. 3: Results obtained from the continuous environmet.

### B. *Continuous environments*

A series of different environments are obtained from a Gazebo simulation, to train and test the agent in more realistic conditions. People and goals are manually set in a continuous environment, in order to recreate a realistic queue scenario. Each of these environments are then discreteized into a 30x30 gridmap with a resolution of 0.2 meters. The virtual obstacles are defined as a box similar to demo version, at a distance of 0.5 meters from the goal and from each person. The new obtained discrete environments are then used to retrain the previous DRL agent. During the training both, the starting point of the agent and the environment, are randomly changed after every episode. Similarly to before, the model is integrated with the $A^*$ planner, and the results can be observed in Figure 3.

## V. Conclusions

We introduced Heuristic Planning with Learned Social Value (HPLSV), a novel RL-based human-aware path planning method. For these preliminary results, we developed a proof of concept on queue following, but intend to extend the methodology to more, if not all, human activities.

The greater limitation of our approach resides in the choice of a proper representation for the working environment. In this work we assumed to know the exact position and orientation of all and only the people in the queue. In the real world, a scene may have many people, most of which have nothing to do with the navigation task. Recognising and interpreting human activity is part of the social navigation challenge, and we do not expect the heuristic function to be able to solve this problem end-to-end, directly for robot sensory inputs. Another strong assumption is the a priori knowledge of the social cost function, represented in training with virtual obstacles. In general, it would be interesting to learn it from real human-robot interactions.

## REFERENCES


[1] E. Martinez-Martin and A. P. del Pobil, "Personal robot assistants for elderly care: an overview," *Personal assistants: Emerging computational technologies*, pp. 77–91, 2018.

[2] A. Vercelli, I. Rainero, L. Ciferri, M. Boido, and F. Pirri, "Robots in elderly care," *DigitCult-Scientific Journal on Digital Cultures*, vol. 2, no. 2, pp. 37–50, 2018.

[3] A. Eirale, M. Martini, L. Tagliavini, D. Gandini, M. Chiaberge, and G. Quaglia, "Marvin: An innovative omni-directional robotic assistant for domestic environments," *Sensors*, vol. 22, no. 14, p. 5261, 2022.

[4] S. Lee, H. Noh, J. Lee, K. Lee, G. G. Lee, S. Sagong, and M. Kim, "On the effectiveness of robot-assisted language learning," *ReCALL*, vol. 23, no. 1, pp. 25–58, 2011.

[5] L. P. E. Toh, A. Causo, P.-W. Tzuo, I.-M. Chen, and S. H. Yeo, "A review on the use of robots in education and young children," *Journal of Educational Technology & Society*, vol. 19, no. 2, pp. 148–163, 2016.

[6] M. Shiomi, T. Kanda, H. Ishiguro, and N. Hagita, "Interactive humanoid robots for a science museum," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 305–312.

[7] A. Al-Wazzan, R. Al-Farhan, F. Al-Ali, and M. El-Abd, "Tour-guide robot," in *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)*. IEEE, 2016, pp. 1–5.

[8] Y. Sasaki and J. Nitta, "Long-term demonstration experiment of autonomous mobile robot in a science museum," in *2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*. IEEE, 2017, pp. 304–310.

[9] S. Pieska, M. Luimula, J. Jauhiainen, and V. Spiz, "Social service robots in wellness and restaurant applications," *Journal of Communication and Computer*, vol. 10, no. 1, pp. 116–123, 2013.

[10] M. Kyrarini, F. Lygerakis, A. Rajavenkatanarayanan, C. Sevastopoulos, H. R. Nambiappan, K. K. Chaitanya, A. R. Babu, J. Mathew, and F. Makedon, "A survey of robots in healthcare," *Technologies*, vol. 9, no. 1, p. 8, 2021.

[11] A. Honour, S. B. Banisetty, and D. Feil-Seifer, "Perceived social intelligence as evaluation of socially navigation," in *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 519–523.

[12] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh, "Core challenges of social robot navigation: A survey," *ACM Transactions on Human-Robot Interaction*, vol. 12, no. 3, pp. 1–39, 2023.

[13] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The museum tour-guide robot rhino," in *Autonome Mobile Systeme 1998: 14. Fachgespräch Karlsruhe, 30. November–1. Dezember 1998*. Springer, 1999, pp. 245–254.

[14] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, *et al.*, "Probabilistic algorithms and the interactive museum tour-guide robot minerva," *The international journal of robotics research*, vol. 19, no. 11, pp. 972–999, 2000.

[15] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[16] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 14 424–14 432.

[17] D. Zhao and J. Oh, "Noticing motion patterns: A temporal cnn with a novel convolution operator for human trajectory prediction," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 628–634, 2020.

[18] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 4601–4607.

[19] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 ieee international conference on robotics and automation (icra)*. IEEE, 2015, pp. 454–460.

[20] C. Park, J. Ondřej, M. Gilbert, K. Freeman, and C. O'Sullivan, "Hi robot: Human intention-aware robot planning for safe and efficient navigation in crowds," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3320–3326.

[21] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[22] C. I. Mavrogiannis and R. A. Knepper, "Multi-agent path topology in support of socially competent navigation planning," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 338–356, 2019.

[23] ——, "Multi-agent trajectory prediction and generation with topological invariants enforced by hamiltonian dynamics," in *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*. Springer, 2020, pp. 744–761.

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

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

[26] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1111–1117.

[27] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[28] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[29] Y. Nakauchi and R. Simmons, "A social robot that stands in line," *Autonomous Robots*, vol. 12, pp. 313–324, 2002.

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[31] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Procs. of the ICAPS workshop on planning under uncertainty for autonomous systems)*, 2005.

[32] K. Ota, "Tf2rl," https://github.com/keiohta/tf2rl/, 2020.