Effects of a Social Force Model reward in Robot Navigation based on Deep Reinforcement Learning

Óscar Gil*, and Alberto Sanfeliu

Institut de Robòtica i Informàtica Industrial, CSIC-UPC
{ogil,sanfeliu}@iri.upc.edu

Abstract. In this paper is proposed an inclusion of the Social Force Model (SFM) into a concrete Deep Reinforcement Learning (RL) framework for robot navigation. These types of techniques have demonstrated to be useful to deal with different types of environments to achieve a goal. In Deep RL, a description of the world to describe the states and a reward adapted to the environment are crucial elements to get the desire behaviour and achieve a high performance. For this reason, this work adds a dense reward function based on SFM and uses the forces in the states like an additional description. Furthermore, obstacles are added to improve the behaviour of works that only consider moving agents. This SFM inclusion can offer a better description of the obstacles for the navigation. Several simulations have been done to check the effects of these modifications in the average performance.

Keywords: Robot Navigation, Deep Reinforcement Learning, Social Force Model, dense reward function

1 Introduction

In Robotics research one of the most important task is improve the ability of a robot to travel to a goal or destination. This ability depends on the challenging that the environment can be, the information about the environment that the robot can obtain and the algorithms which the robot uses to navigate. These factors limit the applicability in most cases.

The situations that can be found in a navigation task are wide and very complex. Different traditional methods have been developed based on potential fields, forces, or other features to solve the task. Actually, one of the approaches to get this purpose is to use Deep Reinforcement Learning techniques.

These techniques have been used to teach the robot to take the correct actions during navigation but it is difficult to achieve a good performance in complex

environments[25]. In the particular case of facing complex obstacles SFM can be useful to describe the environment better and avoid them.

In the remainder of the paper, preliminaries and related work is presented in Sections 2 and 3. In Section 4 the problem formulation is exposed. Section 5 explains a new type of reward based on Social Force Model and Section 6 presents the simulations and the results. Finally, conclusions are discussed in Section 7.

2   Preliminaries

2.1   Social Force Model

For simulate pedestrians motion it is important to consider the physical forces caused by the environment like friction forces to walk, gravity forces or forces in a collision but it is equally important to consider the influence of other pedestrians and obstacles in social rules to walk. For example, when a person who is walking see an obstacle change the trajectory before be near the obstacle to avoid a possible collision. In this process the obstacle does not apply forces to the person but the person reacts as if a force exists.

This type of "virtual forces" that do not exist but can be a way to model the social interactions are known as the Social Force Model[14]. This model simulates the social interactions when a person walks to a determinate goal. There are 2 types of forces in this model applied to an agent n(robot or pedestrian). One is the attractive force to the goal, $\boldsymbol{q}_n^{goal}$,

$$\boldsymbol{f}_n(\boldsymbol{q}_n^{goal}) = k\left(\boldsymbol{v}_n^0(\boldsymbol{q}_n^{goal}) - \boldsymbol{v}_n\right) \ , \tag{1}$$

where k is a constant, $\boldsymbol{v}_n^0(\boldsymbol{q}_n^{goal})$ is the preferred velocity to the goal and $\boldsymbol{v}_n$ is the current agent velocity. The second type of force is the repulsive force of an obstacle or pedestrian z to an agent n,

$$\boldsymbol{f}_{z,n}^{int} = a_z e^{(d_z - d_{z,n})/b_z} \hat{\boldsymbol{d}}_{z,n} \ , \tag{2}$$

where a_z, b_z, d_z are parameters of the forces, $d_{z,n}$ is the distance between the agent n and the obstacle or pedestrian z and $\hat{\boldsymbol{d}}_{z,n}$ is the unitary vector in the direction defined by n and z, pointing to n. The resultant force applied to the agent is:

$$\boldsymbol{F}_n = \boldsymbol{f}_n(\boldsymbol{q}_n^{goal}) + \sum_{z=1}^{Z} \boldsymbol{f}_{z,n}^{int} \ . \tag{3}$$

Several works have been developed using the SFM in order to get socially acceptable trajectories for robots in diverse situations like side-by-side or approaching tasks[7, 8, 20] adding elements to incorporate anisotropy or other goals.

In this work the forces are calculated only for the robot and the resultant force is calculated only with the repulsive forces. The attractive force it's only used in a new type of reward.

2.2   Optimal Reciprocal Collision Avoidance (ORCA)

Reciprocal Velocity Obstacle (RVO) and Optimal Reciprocal Collision Avoidance (ORCA) are reactive methods to avoid collisions between moving agents[2, 26]. In ORCA, the current velocities and positions of all agents in an environment are used to calculate the new velocities that all agents should have to avoid all the possible collisions.

The advantage of this method is that the ratio of collisions is zero by definition because ORCA chooses the velocities to avoid collisions.

On the other hand, if an agent is not controlled by ORCA, it cannot be guaranteed that the collision with this agent is prevented. For this reason, ORCA is not very useful to control an agent like a robot that moves with independent agents like people or animals. Additionally, ORCA has problems when an agent has to avoid a concave obstacle. This problem limits the type of obstacles that can be used in the simulations.

In this paper the RVO library with ORCA is used to simulate the environment with agents and obstacles. The robot is controlled with ORCA in the Imitation Learning steps to learn the basic movements to the goal.

3   Related Work

In this section, different works related to this paper are briefly described.

3.1   The reward shaping problem

It is a very common problem in Reinforcement Learning tasks decide what type of reward choose to improve the training process. This is known as reward design or reward shaping.

Reward shaping is used to decrease the temporal credit assignment problem. This occurs when the agent only receives reward in a goal that must achieve and cannot use this information to know what are the best individual actions to reach the goal.

Several forms of reward shaping can be used to handle the temporal credit assignment[10, 19, 28]. It is difficult to get a proper reward that not get stuck in local minima. In this paper it is proposed a different reward transforming a sparse reward into a dense reward based in the SFM.

3.2   Navigation based on Deep Reinforcement Learning

Supervised Learning techniques have been applied to predict the movements of agents in a temporal horizon. For example, sequence models that use Long Short Term Memory (LSTM) recurrent neural networks like Social LSTM[1] and other[12, 13] are capable to encode the Human-Robot interactions and Human-Human interactions to improve the predictions. Other techniques are based in generative models like Social-GAN[11] or SoPhie[21] that use pooling modules and attention modules.

This type of techniques offers very useful information in navigation tasks but not take into account all the elements in navigation tasks like obstacles, actions, kinematics or goals.

For cover a more general point of view in the robot navigation task, Reinforcement Learning techniques are a more suitable way to achieve a good performance. Normally, these techniques use neural networks due to the very high number of states to make the computation possible and, for that reason, in these cases are Deep Reinforcement Learning techniques.

In particular, here it is used the Temporal Difference (TD) algorithm[22] using a neural network to calculate the value function. The aim of this method is to use the next prediction to actualize the current prediction before reach the final state taking into account that the states are correlated. This method gives very good results in areas like finance[27] and games[24].

Several works[4–6, 17] for navigation have been developed using these methods to learn policies. Sometimes, these methods combine visual information to improve predictions like PoliNet[15]. Another method[9] combines Probabilistic Roadmaps and a RL based local planner to guide the robot for long-range indoor navigation.

There are other works that use Imitation Learning approaches to learn policies[16, 18, 23]. In the Crowd-Robot Interaction (CRI) approach[3], in which is mainly based the navigation in this paper, Imitation Learning is combined with the Deep Reinforcement Learning task to obtain a policy (SARL) for robot navigation. For Imitation Learning, the policy used to control all the agents is ORCA. Deep V-learning algorithm is used in the Imitation Learning and Reinforcement Learning phases with a temporal-difference method, a fixed target network and standard experience replay. In contrast with this work, in the CRI approach obstacles have not been used in the training. It causes some problems in the robot movement when obstacles are added to test the policy.

4   Problem Formulation

In this work, it is considered the same navigation task as in the CRI approach. A robot navigates to a goal through an environment with n agents. The difference is that there are obstacles considered too. The objective is to learn a policy to decide the velocities for navigate to a goal avoiding the agents and obstacles with a velocity close to a preferred value and taking socially acceptable trajectories.

The state of the robot is defined with $\mathbf{s}$. The states of the agents are represented in $\mathbf{w} = [w_1, w_2...w_n]$. In this list of agents, obstacles are included too like agents with velocity equal to zero. All these states are rotated to the robot centred coordinate system with the x axe pointing to the goal. In this coordinate system the states, which proceed from the CRI work, are this,

$$\mathbf{s} = [d_g, v_{pref}, \theta, r, v_x, v_y] \ ,$$
$$\mathbf{w}_i = [p_x, p_y, v_x, v_y, r_i, d_i, r + r_i] \ , \tag{4}$$

where $d_i = ||\mathbf{p} - \mathbf{p}_i||_2$ is the robot's distance to the agent i, $d_g = ||\mathbf{p} - \mathbf{p}_g||_2$ is the robot's distance to the goal, $\mathbf{p} = [p_x, p_y]$ is the agent or obstacle position,

$\mathbf{v} = [v_x, v_y]$ is the current velocity, v_{pref} is the preferred velocity of the robot, θ is an angle different to zero for non-holonomic kinematic cases, r_i is the agent or obstacle radius and r is the robot radius.

In this work it is used a second type of states that include a SFM description of the agents and obstacles introducing the repulsive forces $\boldsymbol{f}_i = [f_x, f_y]$ (described in Section 2.1) of each agent and obstacle applied to the robot. These forces are included in the agent states and obstacle states. In the robot state it is included the resultant force of all the agents and obstacles, $\boldsymbol{F} = [F_x, F_y]$. These states are,

$$
\begin{aligned}
\mathbf{s} &= [d_g, v_{pref}, \theta, r, v_x, v_y, F_x, F_y] \ , \\
\mathbf{w}_i &= [p_x, p_y, v_x, v_y, r_i, d_i, r + r_i, f_x, f_y] \ .
\end{aligned}
\tag{5}
$$

The states for all the agents and obstacles are always known and used to calculate the value function using a neural network and the Temporal Difference algorithm of the CRI work[3]. With this value function the greedy policy is calculated, which gives in this case the robot's velocity(the optimal action) each timestep.

5   Social Force Model Reward

In this work two types of rewards are used to guide the robot to the correct behaviour. The first reward is the reward used in the CRI approach[3]:

$$
R = \begin{cases}
-0.25 & \text{if } d_t < 0 \\
0.25(-0.1 + d_t/2) & \text{else if } d_t < 0.2 \\
1 & \text{else if } \mathbf{p} = \mathbf{p}_g \\
0 & \text{otherwise}
\end{cases} \ ,
\tag{6}
$$

where $d_t = d_i - r - r_i$, represents the distance between agents minus the radius of the 2 agents.

The second reward is a dense reward based on the SFM described in Section 2.1. This Social Force Model Reward takes into account the attractive force to the goal and the repulsive forces of the agents and obstacles,

$$
R = \begin{cases}
-0.25 & \text{if } d_t < 0 \\
Ae^{-Bd_t} & \text{else if } d_t < 0.2 \\
1 & \text{else if } \mathbf{p} = \mathbf{p}_g \\
k - ||k(\boldsymbol{v} - \boldsymbol{v}_{pref})||_2/2 - 0.0001d_g & \text{otherwise}
\end{cases} \ ,
\tag{7}
$$

where $A = -0.03$, $B = 10$ and $k = 0.001$. These values have been chosen looking for rewards that are not very different from the first reward. The first term of the reward is applied in collisions, the second term is applied when an agent or obstacle is very near, the third term is applied if the robot get the goal and the last term is equivalent to an attractive force that gives more reward if the velocity is pointing to the goal and the robot is near to the goal. A variation of this reward it is used in a version with a little temporal discount, $-0.02(t - 10)$, if $t \geqslant 10$ in the third term. This discount is chosen to be not very high and taking into account the average time needed to reach the goal in the best cases.
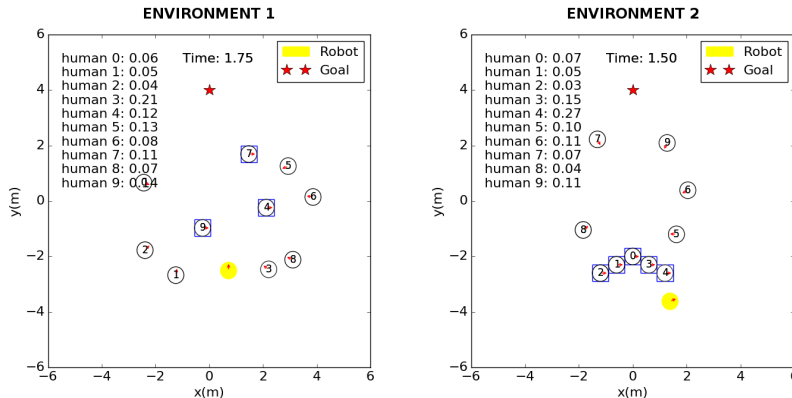
Fig. 1: **First and second environment.** In the first environment obstacles are separated. In the second case, obstacles are joined into a concave barrier.

6   Experiments

In this section are described the environment used in the simulations, the metrics to compare different versions, the results and the implementation details. No real experiments have been conducted.

6.1   Simulation Environment

The environment in the simulations is created using gym and the RVO library and plotted using matplotlib library like in the CRI work. It contains a robot in an initial position $(0, -4)$ and the robot's goal in $(0, 4)$. The agents and the robot are represented like circles using the same radius.

The obstacles are squares whose side is equal to the diameter of the agents and have an agent inside. This agent inside the obstacle has velocity zero and gives the state's information of the obstacle, w_i. This is to achieve that the robot be aware of the obstacle when is controlled by the SARL policy. The square obstacle is generated to achieve that the agents controlled by ORCA distinguish between agents and obstacles.

There are 5 types of environments that always contain 10 elements generated randomly as agents or obstacles. In the first environment, shown in Fig. 1, the agents are generated in a circumference with their goals in the opposite side. The probability of generate an agent is 0.6 and 0.4 for an obstacle. The second environment, shown in Fig. 1, has always 5 obstacles joined into a fixed concave barrier and 5 agents in a circumference with their goals in the opposite side. The other environments in Fig. 2 use combinations of the concave barrier and barriers of 2 or 3 obstacles. In environment 3 the concave barrier can be in different positions between the robot and the goal.
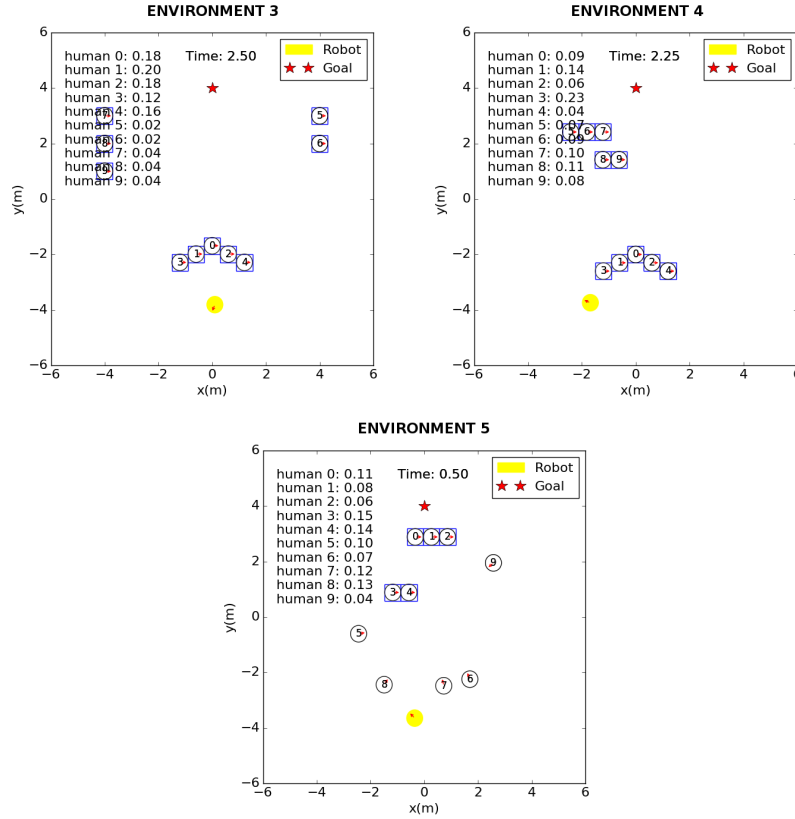
Fig. 2: **Other environments.** Environments from the third to the fifth.

6.2   Metrics

In this section are explained the metrics used to measure the performance of the navigation. These metrics are the same than in the CRI work.

- **Success Rate**: This value measures the ratio of times that the robot reaches the goal.
- **Collision Rate**: This value measures the ratio of times that the robot collides with agents or obstacles.
- **Navigation Time**: Time used for the robot to reach the goal. The maximum time is 25 s.
- **Total Reward**: The average cumulative reward of all the episodes.

The most meaningful metrics are the success rate and collision rate because brief on the most important features of the navigation task.

Version	Success Rate	Collision Rate	Navigation Time	Total Reward
SARL(2 times)	**0.96/0.97**	0.02/0.01	**10.85**/11.77	**0.3059/0.2850**
SARL-SFM	0.91	0.03	11.83	0.2417
SARL-SFM2	0.86	0.02	11.92	0.2363
SARL-SFM3	0.93	0.01	11.27	0.2741
SARL-SFM4	0.90	**0.00**	11.61	0.2646
SARL-SFM5	**0.96**	0.03	11.69	0.2799
SARL-SFM6	**0.96**	0.03	11.25	0.2713

Table 1: **Test results.** Average results obtained in the test evaluation after 500 episodes. All these cases are obtained using only the two first environments.

Version	Success Rate	Collision Rate	Navigation Time	Total Reward
SARL	**0.901**	**0.001**	**10.54**	**0.3046**
SARL-SFM6	0.893	0.004	10.88	0.2763

Table 2: **Second test results.** Average results obtained in the test evaluation after 1000 episodes using all the environments in training.

6.3   Quantitative Evaluation

In Table 1 the average results of a test using 500 episodes are exposed for the different versions:

- **SARL**: The CRI approach without local map.
- **SARL-SFM**: With SFM reward and temporal discount in the reward.
- **SARL-SFM2**: With SFM reward.
- **SARL-SFM3**: With SFM reward but without concave barrier in training.
- **SARL-SFM4**: With SFM reward but $k = 0.003$ in the reward.
- **SARL-SFM5**: With forces included in the states and without SFM reward.
- **SARL-SFM6**: With forces included in the states and with the SFM reward.

In the Fig. 3 is shown the success rate for each type of training during the validation. In Table 2 are shown the results of a second type of training with only the SARL and SARL-SFM6 cases using all the environments.

6.4   Qualitative Evaluation

Comparing to the cases analysed in the CRI paper, in which there were only 5 moving agents without obstacles, this paper has proposed a more demanding environment with 10 elements between agents and obstacles. Normally, with the new environments in training, the robot does not get stuck because of the obstacles and can surpass the concave barrier.
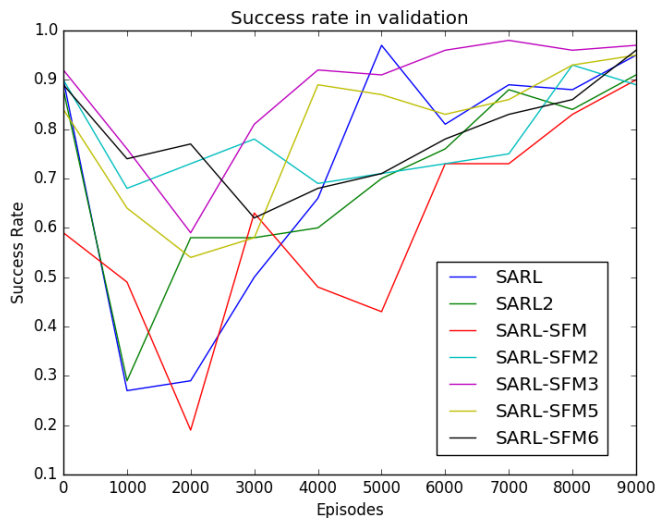
Fig. 3: **Validation graphic.** This graphic shows the validation results for the success rate each 1000 episodes of the first type of training.

In spite of the fact that the robot faces better the obstacles, there are some cases where the robot is very slow avoiding the concave barrier. If there are not agents moving in the environment the robot gets stuck with the concave barrier.

In Table 1, results obtained are, in general, worse than in the CRI paper because of the more challenging environment and the best success rate is obtained without the SFM reward. Comparing the SARL-SFM2 to SARL-SFM3, although the success rate is better in SARL-SFM3, in SARL-SFM2 the robot get stuck fewer in the concave barrier episodes. For this reason, the concave barrier can be useful in the training phase.

In the SARL-SFM4, k is 3 times bigger to increment the rewards and the influence of the attractive effect to the goal and there is not temporal discount. This causes a worse performance than the SARL-SFM case.

Although the test results are better in the SARL case, the training progression is not equal for all the cases. This can be observed in Fig. 3. For example, in the SARL-SFM3 the validation applied after 7000 episodes of training gives a success rate of 0.98. In SARL-SFM5, this success rate after 4000 episodes is 0.89, while in the SARL case was only 0.66. In the SARL-SFM case the success rate is always smaller than the SARL success rate.

The 2 last cases, SARL-SFM5 and SARL-SFM6, combine the SFM reward with the new states and offers better results that cases with only SFM reward. The convergence is faster in the first part of the training than the SARL case and the performance in test is more or less the same. These results suggest that

the training progression is faster introducing the forces in the states and the oscillations in training are reduced.

In Table 2 all the environments are used in the training and the performance of the robot facing obstacles improves only a little in the SARL and SARL-SFM6 cases. There are more or less a 10 % of cases where the robot get stuck and these cases normally occur in environments that only contain obstacles.

6.5   Implementation Details

For the implementation, the robot is visible for the agents and the different versions that are compared do not use the local map of the CRI work. The neural network is the same as the network used in the CRI approach. It is only changed the input layer in the first multilayer perceptron (MLP) and the input layer in the last MLP to use the new states proposed in this paper. In the states, the parameters of the repulsive forces described in Section 2.1 are $a_z = 1$, $d_z = 0$ and $b_z = 1$. This choice offers inputs neither too big nor too small to the neural network.

In the optimization task it is used the Stochastic Gradient Descent method with momentum 0.9. The batch size is 100.

For imitation learning, 3000 episodes are collected using ORCA for the robot and the policy is trained with 50 epochs and learning rate 0.01. In this phase, the environment is the one without the concave barrier in all cases.

For reinforcement learning, the learning rate is 0.001, the discount factor is 0.9, the number of episodes is 10.000 and the policy used is the ϵ-greedy policy of the CRI work. Each 1000 episodes there is a validation phase with 100 episodes. At the end of the training there is a test phase with 500 episodes. In Table 1, the environment used for the reinforcement learning phase of the training, the validation phase and the test phase is randomly chosen each episode between one of the two first environments with probabilities 0.7 for the first environment and 0.3 for the second one. In Table 2, the environments used in reinforcement learning are a total of 4, from the second to the fifth, with probabilities 0.25, the number of training episodes is 15.000 and there are 1000 test episodes.

The work assumes holonomic kinematics and the same action space that is used in CRI work. The implementation has been developed in PyTorch and the simulations have been launched in a Tesla K40c GPU.

7   Conclusion

In this work, 3 modifications have been done in order to check the changes in the performance of the SARL policy, designed in the CRI approach.

Firstly, the environment has been expanded through obstacles to improve the robot's behaviour when faces them. This type of modification has slightly improved the robot behaviour but there are still some problems when the robot faces only obstacles and a better description of the environment or a different neural network would be required.

Secondly, the reward signal has been modified taking account the SFM. This type of modification has caused overall worse results in the navigation performance reducing the success rate and increasing the collision rate. In spite of this problem, the convergence improves in the first 5000 episodes. This better initial convergence could be caused because the algorithm has additional information in the first part of the training.

Finally, the last modification has been the inclusion of the repulsive forces in the agent states and the resultant force in the robot state. This modification has not produced notable changes but it appears that the convergence is faster in the first 5000 episodes. It also makes the test results very much independent of using SFM reward, palliating its negative effect.

To sum up, this approach offers a way to slightly accelerate the convergence in the TD learning for navigation tasks. For better results in the performance of the policy are suggested as future work, forms of reward shaping, other RL algorithms like Policy Gradients, more prediction information of the environment like in Social GAN work[11] and mix the RL algorithms with a global planner[9].

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 961–971 (June 2016)
2. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: 2008 IEEE International Conference on Robotics and Automation. pp. 1928–1935. IEEE (2008)
3. Chen, C., Liu, Y., Kreiss, S., Alahi, A.: Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In: 2019 International Conference on Robotics and Automation. pp. 6015–6022 (May 2019)
4. Chen, Y.F., Everett, M., Liu, M., How, J.P.: Socially aware motion planning with deep reinforcement learning. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1343–1350. IEEE (2017)
5. Chen, Y.F., Liu, M., Everett, M., How, J.P.: Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 285–292 (2017)
6. Everett, M., Chen, Y.F., How, J.P.: 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 (2018)
7. Ferrer, G., Sanfeliu, A.: Proactive kinodynamic planning using the extended social force model and human motion prediction in urban environments. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1730–1735. IEEE (2014)
8. Ferrer, G., Zulueta, A.G., Cotarelo, F.H., Sanfeliu, A.: Robot social-aware navigation framework to accompany people walking side-by-side. Autonomous robots 41(4), 775–793 (2017)
9. Francis, A., Faust, A., Chiang, H.L., Hsu, J., Kew, J.C., Fiser, M., Lee, T.E.: Long-range indoor navigation with PRM-RL. CoRR abs/1902.09458 (2019)
10. Grzes, M., Kudenko, D.: Reward shaping and mixed resolution function approximation. In: Developments in Intelligent Agent Technologies and Multi-Agent Systems: Concepts and Applications, pp. 95–115. IGI Global (2011)

11. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social GAN: Socially acceptable trajectories with generative adversarial networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2255–2264 (June 2018)
12. Haddad, S., Wu, M., Wei, H., Lam, S.K.: Situation-aware pedestrian trajectory prediction with spatio-temporal attention model. CoRR abs/1902.05437 (2019)
13. Hasan, I., Setti, F., Tsesmelis, T., Del Bue, A., Galasso, F., Cristani, M.: MX-LSTM: mixing tracklets and vislets to jointly forecast trajectories and head poses. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6067–6076 (2018)
14. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Physical review E 51(5), 4282 (1995)
15. Hirose, N., Xia, F., Martin-Martin, R., Sadeghian, A., Savarese, S.: Deep visual MPC-Policy learning for navigation. arXiv preprint arXiv:1903.02749 (2019)
16. Liu, Y., Xu, A., Chen, Z.: Map-based deep imitation learning for obstacle avoidance. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 8644–8649. IEEE (2018)
17. Long, P., Fanl, T., Liao, X., Liu, W., Zhang, H., Pan, J.: Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: 2018 IEEE International Conference on Robotics and Automation. pp. 6252–6259 (2018)
18. Long, P., Liu, W., Pan, J.: Deep-learned collision avoidance policy for distributed multiagent navigation. IEEE Robotics and Automation Letters 2(2), 656–663 (2017)
19. Ng, A.Y., Harada, D., Russell, S.J.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the Sixteenth International Conference on Machine Learning. pp. 278–287. ICML '99, Morgan Kaufmann Publishers Inc. (1999)
20. Repiso, E., Garrell, A., Sanfeliu, A.: Adaptive side-by-side social robot navigation to approach and interact with people. International Journal of Social Robotics pp. 1–22 (2019)
21. Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., Savarese, S.: Sophie: An attentive GAN for predicting paths compliant to social and physical constraints. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1349–1358 (2019)
22. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, second edn. (2018)
23. Tail, L., Zhang, J., Liu, M., Burgard, W.: Socially compliant navigation through raw depth inputs with generative adversarial imitation learning. In: 2018 IEEE International Conference on Robotics and Automation. pp. 1111–1117 (2018)
24. Tesauro, G.: Temporal difference learning and TD-Gammon. Communications of the ACM 38(3), 58–68 (1995)
25. Trautman, P., Ma, J., Murray, R.M., Krause, A.: Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. The International Journal of Robotics Research 34(3), 335–356 (2015)
26. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics research, pp. 3–19. Springer (2011)
27. Van Roy, B.: Temporal-difference learning and applications in finance. Cambridge, MA: MIT Press (2001)
28. Zou, H., Ren, T., Yan, D., Su, H., Zhu, J.: Reward shaping via meta-learning. CoRR abs/1901.09330 (2019)