Collision-free Navigation of Human-centered Robots via Markov Games

Guo Ye[1*] Qinjie Lin[1*] Tzung-Han Juang[1] Han Liu[1]

***Abstract*— We exploit Markov games as a framework for collision-free navigation of human-centered robots. Unlike the classical methods which formulate robot navigation as a single-agent Markov decision process with a static environment, our framework of Markov games adopts a multi-agent formulation with one primary agent representing the robot and the remaining auxiliary agents form a dynamic or even competing environment. Such a framework allows us to develop a path-following type adversarial training strategy to learn a robust decentralized collision avoidance policy. Through thorough experiments on both simulated and real-world mobile robots, we show that the learnt policy outperforms the state-of-the-art algorithms in both sample complexity and runtime robustness.**

***Index Terms*— Collision-free navigation, human-centered robotics, deep reinforcement learning, multi-agent system, adversarial training**

I. INTRODUCTION

This paper studies the collision-free navigation problem of human-centered robots [1], [2]. Such robots need to interact, assist and cooperate with humans. One motivating example is grocery robots as shown in Figure 1, from which we see such robots must operate in a dynamic environment due to humans activities. Collision-free navigation [3]–[9] is a fundamental required capability of human-centered robots.

The dynamic environment of human-centered robots prevents us from using traditional trajectory-based methods [10], [11] or reinforcement learning based methods which assume a static map [7], [12]. More specifically, these methods formulate the robot navigation problem as a single-agent Markov decision process, then conduct either forward path planning (random tree search algorithms) [13], [14] or backward policy search (e.g., policy gradient algorithms) [15]–[18] for collision-free navigation. They all assume the robot operating in a static environment, which is unrealistic for human-centered robots.

To handle this challenge, we exploit Markov games [19]–[22] as a modeling framework for collision-free navigation. Unlike traditional trajectory-based methods or reinforcement learning based methods, our framework of Markov games adopts a multi-agent formulation with one primary agent representing the robot and the remaining auxiliary agents form a dynamic or even competing environment. This framework allows us to develop an adversarial training strategy to learn a robust decentralized collision avoidance policy.

Under the multi-agent formulation [23]–[27], the primary agent representing robot is called a protagonist. Our method spawn several auxiliary agents who attack the protagonist in an adversarial way. These adversary agents get reward by resulting in protagonist not reaching the goal. A major contribution of this paper is the development of a novel path-following policy learning strategy for solving the Markov games. More specifically, all agents are modeled by a Markov process but we allow the auxiliary agents to have a global view of the system state while the primary agent (the robot) only have a local view. The Markov processes of the auxiliary agents are centrally configurable and are parameterized by a set of aggressiveness parameters (e.g., moving velocity, perception accuracy, etc.) that affect their capabilities being adversarial. By varying the aggressiveness parameters from tight to more relaxed (i.e., the auxiliary agents become more and more adversarial), the obtained parametric family of Markov decision processes form a regularization path that can be used to robustly train the primary agent's navigation policy using a path-following algorithm (More details are provided in Section III). The same idea has also been exploited in developing the interior point method [28] in nonlinear optimization and parametric simplex method in linear optimization [29].

Fig. 1: A grocery robot navigating in a retail scenario. Such robots must operate in a dynamic environment due to humans activities. Picture source: https://vosizneias.com/.

We conduct thorough numerical simulations to demonstrate the efficacy of the proposed method. We show that the learnt policy is simultaneously efficient in sample complexity and adaptive to complex human-centered environment. We also conduct a sensitivity analysis to demonstrate the robustness of the proposed algorithm. In addition, we deploy the learnt policy on a real mobile robot equipped with only one low cost 2D LIDAR and show that the trained agent is directly deployable to complex environment.

*Contributed equally.

[1]Department of Computer Science, Northwestern University, Evanston, IL, USA 60201 guoye2018, qinjielin2018, tzung-hanjuang2018@u.northwestern.edu; hanliu@northwestern.edu

II. RELATED WORK

This section summarizes some related work. Relevant literature includes Multi-agent collision-free navigation, Markov games formulation of mobile robots and adversarial learning.

A. Multi-Agent Collision-free Navigation

There are two collision-free navigation paradigms in multi-agent environments: the centralized approach vs decentralized approach. The centralized approach [30], [31] assumes each agent has perfect knowledge of the other agents, which is unrealistic for human-centered robots since the primary agent (the robot) obvious can not know the perfect state of every auxiliary agent (humans). In contrast, the decentralized approach assumes the primary agent only has partial observation of the rest. Thus is more relevant to our setting. Related works on decentralized collision-free navigation methods include [32] and [17]. However, [32] only considers auxiliary agents with stochastic or prefixed policies while [17] assumes the primary and auxiliary agents share the same policy. In contrast, our method models the auxiliary agents using an adversarial setting which improves both sample complexity and policy robustness of the primary agent.

B. Makov Games Formulation of Multi-agent Systems

Markov games are powerful at modeling multi-agent systems [25], [26]. However, solving their equilibrium is nontrivial. To achieve tractable solution, one approach is to constrain the problem into a two-agent zero-sum game to learn a stationary policy for both protagonist and adversary [33]. Another approach is to exploit centralized actor-critic type methods [34]. A third approach is to constrain all the agents can only communicate with their neighbors in a network setting [35]. None of these methods is readily applicable to our setting where the protagonist (the robot) is decentralized while the remaining agents are centralized.

C. Adversarial Learning

Motivated by the success of generative adversarial networks [36], recent works show that training an agent in an adversarial setting could lead to improved performance. For example, in game environments, Trapit et al. [37] and Adam et al. [38] applied 2 competeting agents (one protagonist and one opponent) in the training process and obtained superior performance compared to the vanilla reinforcement learning methods. For driving games, [39] showed that the protagonist works better in the presence of a risk-seeking opponent. All these works exploit two-player zero-sum games. In contrast, we employ multi-agent general-sum Markov games which are more challenging to solve and motivate the development of a new path-following adversarial training strategy (More details are provided in Section III).

III. METHODS

In this section, we formulate the collision-free navigation problem as a Markov game and describe a path-following type strategy for solving it. Though our algorithmic framework is fully generic and applicable to any human-centered robot, we describe the algorithm using a simple 2D `Stage` simulation environment [40] with the hope of making the main idea more concrete to accessible to the audience. All the agents in this simulator exploits a 2D LIDAR with laser scan range 3.5m as sensing device. The maximum linear speed of the mobile robot in this simulator is 1m/s.

A. Collision-free Navigation Formulated as Markov Games

A human-centered robot needs to navigate in an environment with the interference of one or more moving humans. This scenario can be modeled as a multi-agent system with the robot as the primary agent (protagonist) and the humans as auxiliary agents (adversaries). Each agent is modeled by a Markov decision process. We allow the adversaries to access the global state information of all other agents while the protagonist to access only local information of its nearby agents. Each agent aims at maximizing its accumulative reward under its own reward mechanism. These co-evolving and competitive agents thus form a Markov game [41]:

$$\mathcal{M} = (S, O_1 \dots O_N, A_1 \dots A_N, T, R_1 \dots R_N).$$

Here N is the total number of agents with Agent 1 as the protagonist. S denotes the states of all agents. $O_1 \dots O_N, A_1 \dots A_N$ are the sets observations and actions for each agent. A_i is action of Agent i sampled from a stochastic policy π_i and the next state is generated by the state transition function $T : S \times A_1 \times \dots \times A_N \to S$. At each step, every agent gets a reward according to the state and corresponding action $r_i : S \times A_i \times \dots \times A_N \to \mathbb{R}$ along with an observation of the system state $o_i : S \to O$. While the adversarial agents $2, \dots, N$ have access to the global information S, the protagonist only have access to partial information o_i (More details are provided in Section IV). The objective for the i-th agent is to learn a policy that maximizes the cumulative discounted rewards

$$\mathcal{R}(i) := \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t r_i^{(t)})\Big], \tag{1}$$

where $\gamma \in (0, 1)$ is the discount factor and $r_i^{(t)}$ is the the reward received at the t-th step.

B. Path-following Policy Learning Strategy

The cumulative discounted reward of the protagonist $R(1)$ and those of the adversaries $R(2), \mathcal{R}(3) \dots, \mathcal{R}(N)$ are coupled since these adversary agents get reward by resulting in protagonist not reaching the goal. This forms a general-sum Markov game and it is nontrivial to solve its equilibrium. To proceed, we propose a path-following policy learning strategy. The main idea is to parameterize all the auxiliary agents by a set of aggressiveness parameters that affect their capabilities being adversarial. By varying the aggressiveness parameters from tight to more relaxed, the auxiliary agents become more and more adversarial. In another word, the environment of the protagonist shifts from being stochastic to more adversarial. The obtained parametric family of Markov decision processes form a regularization path that can be

used to robustly train the primary agent's navigation policy using a path-following strategy.

More specifically, we consider the following set of aggressiveness parametrized of the auxiliary agents: (i) *Linear velocity*: The mobile robot contains linear and angular speed v_ℓ, v_ω, the latter does not have notable influence to adversaries' ability. Hence, we only consider the linear velocity in the range $v_\ell \in [0, 1.5]$. (ii) *Adversarial size*: We model the shape of each auxiliary agent by a square with width $s \in [0.1, 0.4]$. The larger size brings difficulty for the protagonist to conduct collision avoidance. (iii) *Perception accuracy*: Each adversarial agent can access the positions of other agents with an additive Gaussian noise with spheirical covariance matrix $N(0, 1/\rho^2 I)$. Here we set $\rho^2 \in [0.2, 1000]$. We parameterize the above agressiveness parameters using a one-dimensional parameter $\tau \in [0, 1]$ to be monotone increasing functions $v_\ell(\tau), s(\tau)$ and $\sigma(\tau)$ such that $v_\ell(0) = 0, v_\ell(1) = 1.5$; $s(0) = 0.1, s(1) = 0.4$; $\rho^2(0) = 0.2, \rho^2(1) = 1000$. Once the functions $v_\ell(\cdot), s(\cdot), \rho^2(\cdot)$ are fixed, the adversarial capabilities of the adversarial agents are indexed by the parameter τ. Our path-follwing strategy is to train the policy of the protagonist along the regularizaiton path by varying τ from 0 to 1. For simplicity, we set the functions $v_\ell(\cdot), s(\cdot), \rho^2(\cdot)$ to be linear in this paper. More sophisticated parameterization paradigms are also possible.

To train the protagonist's poicy along the path, we conduct reinforcement learning of the protagonist in an environment with adversaries of minimum difficulty $\tau = 0$. We then increase the aggressiveness level of the adversary agents to a higher level according to some pre-defined updating rule and use the learnt policy from the previous step for initialization. The process keeps going until reaching the hardest setting corresponding to $\tau = 1$ as Algorithm 1 shows.

Algorithm 1: Path-following Policy Learning

Require: Transition model T of the Markov game.

Initialize $\tau = 0$; The policy $\pi_1, \ldots, \pi_N$ of each agent; Maximum number of episodes E.

for episode =*1, …, E* **do**

  for *m* steps **do**

    Each agent selects action $a_i \sim \pi_i(o_i)$.

    Step forward $s', r \leftarrow T(s, a_1, \ldots, a_N)$.

    Each agent collects trajectory (s, s', a, r).

    Update π_i using collected data

  Update τ using certain updating rule.

Return π_1^*

The above path-follwoing algorithm requires an updating rule of the parameter τ. We consider 3 updating methods: (i) *discrete*, (ii) *sigmoid* and (iii) *linear*. Let E be the number of maximum episodes and i be the episode index. The discrete rule generates a path with only 3 values of $\tau = 0, 0.5, 1$, the sigmoid rule generates a path by varying τ along a sigmoid-shaped curve $1/(1 + e^{-(i-E/2)})$. The linear rule generates a path by varying τ along a linear line from 0 to 1.

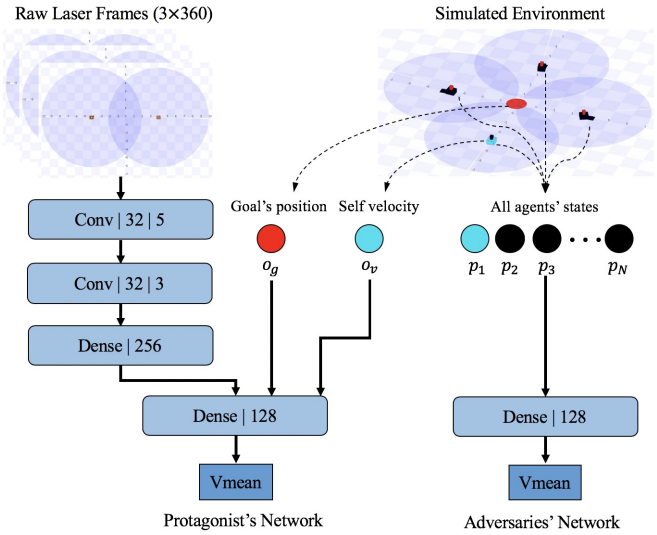


Fig. 2: Network Architecture of Protagonist and Adversary. Conv represents convolution layer followed by its dimension and kernel size. Dense represents fully-connected layer with its dimension.

C. Protagonist Setting

To train the protagonist, we use the same PPO algorithm [42] as described in [17] with some minor adaptions (e.g., the input dimension of the LIDAR has different dimensions). For the purpose of completeness, we briefly describe the setups in this section to ease the audience.

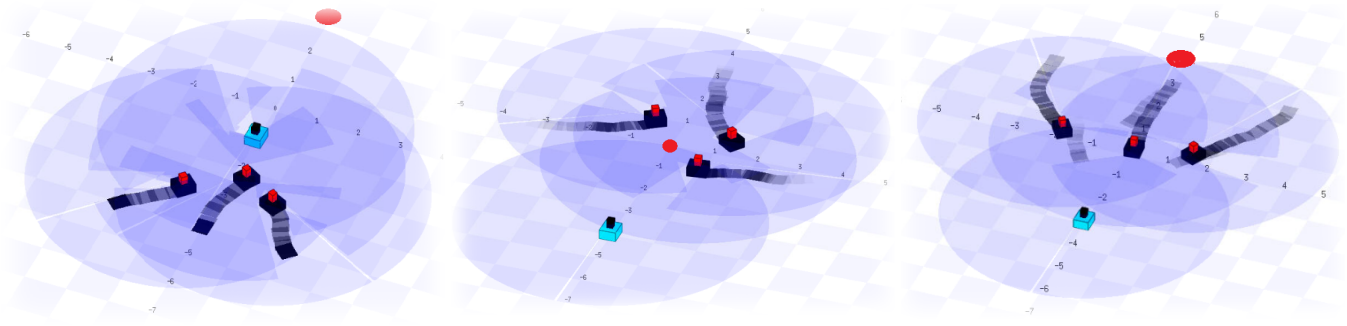
1) Observation Space and Action Space: The observations of the protagonist contain laser readings $o_\ell \in \mathbb{R}^{3\times 360}$ (Representing data from the 3 most recent frames of the LIDAR scan, each of which is a 360-dimensional vector), current velocity $o_v \in \mathbb{R}^2$ including both linear and angular velocities and the target location's relative polar coordinates $o_g \in \mathbb{R}^2$ with respect to the agent's local coordinate system. Note that each value of the laser reading o_ℓ lies in between 0 to 3.5 which is the range of laser beam.

2) Reward Setting: To navigate the agent, the reward function is the sum of 4 components:

$$r^{(t)}(s^{(t)}, a^{(t)}) = r_d^{(t)} + r_g^{(t)} + r_c^{(t)} + r_w^{(t)}.$$

Here $r_d^{(t)} = w_d \cdot (\|\mathbf{p}_1^{(t-1)} - \mathbf{p}_g\|_2 - \|\mathbf{p}_1^{(t)} - \mathbf{p}_g\|_2)$ guides the agent to move towards the goal. $\mathbf{p}_1 = [\mathbf{p}_1^x, \mathbf{p}_1^y] \in \mathbb{R}^2$ and $\mathbf{p}_g^{(t)} = [\mathbf{p}_g^x, \mathbf{p}_g^y] \in \mathbb{R}^2$ are the current and goal positions of the protagonist. We set the weight $w_d = 2.5$. Reward for reaching the goal $r_g^{(t)} = 15$ if $\|\mathbf{p}_1^{(t)} - \mathbf{p}_g\|_2 < 0.5$. We set $r_c^{(t)} = -15$ if a collision happens and similarly $r_w^{(t)} = w_\omega \cdot |\omega^{(t)}|$ if $\omega^{(t)} > 1.05$ with $w_\omega = -0.1$ is used to force the protagonist moving more smoothly.

3) Network Architecture: The network architecture is shown in Figure 2. The LIDAR data is fed to two convolution layers and one 256-dimensional fully-connected (fc) layer. The processed laser information is then concatenated with the other two observations o_g and o_v. The concatenated vector is fed into a 128-dimensional fully-connected layer to generate the mean of the velocity v_{mean}. We then construct a Gaussian policy with v_{mean} as the mean parameter and a



(a) Chasing (b) Blocking (c) Crossing

Fig. 3: The 3 Adversarial Paradigms: Chasing, Blocking and Crossing. The blue cube represents the protagonist, the black ones represents adversaries. The red circle denoting protagonist's goal location. The trajectories of adversaries present different attacking strategies.

variance parameter v_{std} trained in the same way as in [17]. In addition to the policy network, the PPO algorithm also involves a value network which is nearly the same except the last fully-connected layer outputs a value for judging the policy's performance. More details can be found in [17].

D. Adversarial Setting

Unlike most multi-agent navigation methods which are either fully centralized with all agents share the same policy [43] or fully decentralized with all agents only have limited partial observations [17], [23], our setting have all adversaries centralized but with different policies and the protagonist decentralized. This allows us to fully harness the benefit of adversarial learning to obtain a more robust policy for the protagonist. Since all adversaries are centralized, each one has an observation containing the position information of all the agents: $o_a^{(t)} = [\mathbf{p}_1^{(t)} \dots \mathbf{p}_N^{(t)}, \mathbf{p}_g^{(t)}]$, where $\mathbf{p}_i^{(t)} = [\mathbf{p}_i^x, \mathbf{p}_i^y] \in \mathbb{R}^2$ represents the current position of the i-th agent. $\mathbf{p}_g^{(t)}$ is the goal position of the protagonist. The observation $o^{(t)}$ is fed into a neural network with the same architecture as in Figure 2 but each adversary agent independently trains the network parameters. Similarly, a Gaussian policy is constructed for each adversary with the output action $a^{(t)} = [v_l, v_w] \in \mathbb{R}^2$ containing linear and angular velocities. To further promote the diversity of the adversarial agents, we consider 3 adversarial paradigms shown in Figure 3. These paradigms have different attacking strategies characterized by different reward mechanisms. To train the protagonist, we simultaneously launch multiple instances of the adversarial agents under different paradigms. The policy networks of the adversaries from the same paradigm share the same parameters. The protagonist is simultaneously trained in these instances using the path-following strategy. We describe the 3 adversarial paradigms below.

1) Chasing: In this paradigm, 3 adversaries are chasing the protagonist and are rewarded when catching it. The reward function of adversaries is:

$$r^{(t)}(s^{(t)}, a^{(t)}) = r_d^{(t)} + r_c^{(t)},$$

where $r_d^{(t)} = w_d \cdot (\|\mathbf{p}_i^{(t-1)} - \mathbf{p}_1\|_2 - \|\mathbf{p}_i^{(t)} - \mathbf{p}_1\|_2)$ with $w_d = 2.5$ and $r_c^{(t)} = 15$ if collision with protagonist happens. It is easy to see that $r_d^{(t)}$ encourages an adversary to chase the protagonist and $r_c^{(t)}$ encourages the collision. One thing to note is that if two adversaries collide, there is no punishment but will terminate the current episode and all the adversaries will be relaunched at new random positions.

2) Blocking: This paradigm has 3 adversarial agents trying to prevent the protagonist from reaching the goal position with the following reward function:

$$r^{(t)}(s^{(t)}, a^{(t)}) = (r_d^{(t)} - r_{d'}^{(t)}) + r_c^{(t)},$$

where $r_d^{(t)}, r_c^{(t)}$ are the same as in the chasing section and $r_{d'}^{(t)} = w_d \cdot (\|\mathbf{p}_1^{(t-1)} - \mathbf{p}_g\|_2 - \|\mathbf{p}_1^{(t)} - \mathbf{p}_g\|_2)$ with $w_d = 2.5$ encourages the protagonist to move towards the goal $\mathbf{p}_g$. It is easy to see that the component $r_{d'}^{(t)}$ encourages the adversaries to block the protagonist away from the goal.

3) Crossing: This paradigm is exactly the same as the chasing paradigm except the goal position is purposely put behind the 3 adversaries. The reason for designing this paradigm is that crossing is the one of the most common scenarios encountered by a human-centered robot. Many traditional approaches assume the humans follow social norms or reciprocal policy and should actively avoid colliding robot. However, in real applications, there could be some curious huamans (e.g., a kid in the grocery store) who tend to get close to the robot and that is the scenario we aim to model.

IV. EXPERIMENTS

We present experimental results to demonstrate the efficacy of the proposed method. In particular, we show that our method outperforms the existing state-of-the-art collision-free navigation methods in both benign and adversarial environments. We also deploy our learnt policy to a physical robot and demonstrate that the trained agent is directly applicable to real-world physical environment.

A. Computational Details

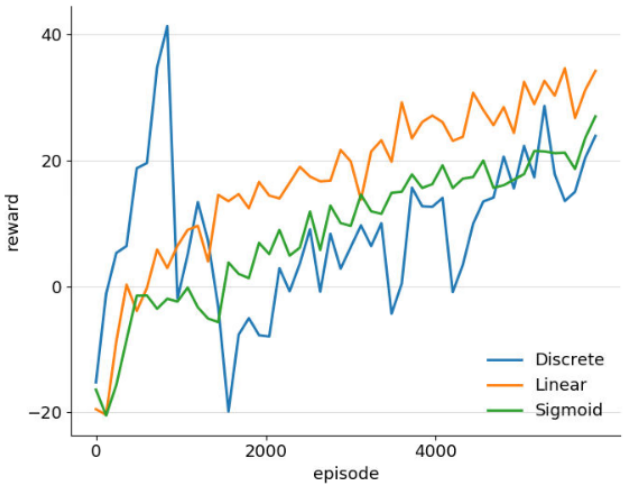
We conduct the multi-agent robots simulation in `Stage` [40] with the algorithm implemented in PyTorch [44]. The network architectures are the same as shown in Figure 2. The maximum linear speed of the mobile robot during training is 1m/s and the 2D LIDAR equipped on the robot senses 360 degree and 0-3.5m distance. We train collision-free policy on a server with 2 Xeon 8168 CPUs (48 cores), 1TB memory

and 4 Nvidia GTX 2080 GPUs. The collision-free policy training takes 4 hours (about 2000 episodes) to converge to a robust solution in all adversarial paradigms.

B. Experimental Design

1) Evaluation Metrics: We consider the following metrics adopted from [17]: (i) *Success rate*: The proportion of trials that the protagonist successfully reaches the target position without any collision. (ii) *Extra time*: The difference between the average travel time of the protagonist over all testing cases and the lower bound of the travel time measured by robot moving straightly toward goals without adversarial interference. (iii) *Extra distance*: The difference between average travel distance over all testing cases and the travel distance measured by the protagonist moving straightly towards the goal without adversarial interference. (iv) *Average speed*: The average speed of the protagonist achieving goal.

2) Baseline Methods: We consider the following baseline methods: (i) ORCA [43], a centralized method that have a central sever controlling all agents. (ii) CADRL [32], an agent-level decentralized method with each agent accesses precise state information. The purpose of comparing with CADRL is to show that even with more crude input, our method achieves higher performance. (iii) DRLMACA [17], a fully decentralized method in which a same policy is trained for all agents in several multi-agent environments.

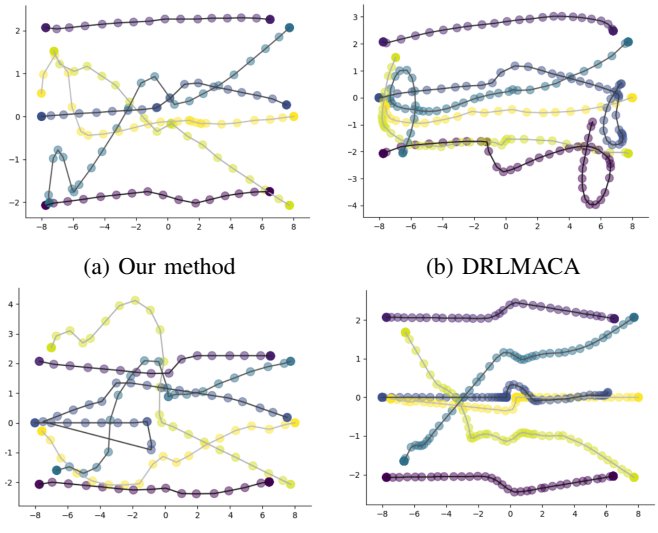


x

Fig. 4: Protagonists' reward curves for collision avoidance policy learning. The learning curves of *Discrete*, *Linear*, *Sigmoid* path-following policy learning strategy converge around the reward 20.

C. Policy Training

Figure 4 evaluates the influence of different updating rules of τ (noted as *Discrete*, *Sigmoid* and *Linear*) on policy training. We see two dramatic drop down of the blue curve (discrete) corresponding to where adversaries change their aggressiveness intensity. Whenever τ increase 0.5 after each training epoch, it brings dramatic hardness to the environment. However, after several episodes, the protagonist becomes accustomed to the new switch and converges by the end. Unlike the bumpy curve from the *discrete* case, the other two reward curves change more smoothly with the *Linear*'s reward slightly outperforms the *sigmoid*'s. The reason could be the hardness of *sigmoid* starts increasing more rapidly than *linear* in the halftime which is more challenging for the protagonist to adapt. At the end, no matter in which way, all three methods converge to the same level.



(a) Our method (b) DRLMACA

(c) CADRL (d) ORCA

Fig. 5: The trajectory plots of 6 robots that are equally divided into two groups trying to reach the opposite positions. The interval between two adjacent footprints is 1s. ORCA moves extremely slow inferred by dense points in a trajectory but almost following the optimal path. CADRL moves fast while traveling longest. Some redundant trajectories shaped in circles are observed using DRLMACA. Our method achieves the best balance between time and travelling distance.

D. Simulation Results

We compare our method with the baselines under both non-adversarial Figure 5 and adversarial environments Figure 3. All agents have identical size and speed same across different methods.

1) Non-adversarial Scenario: We place 6 robots in two groups facing each other. They're required to reach the positions their opposite agents located. We visualize the obtained trajectory plots in Figure 5. As expected, all methods achieve high success rate in this non-adversarial scenario. However, the CADRL and ORCA exploit global information which is not needed by our method. We observe that CADRL and ORCA chose completely opposite strategies for collision avoidance. CADRL is more aggressive with very high speed but takes much longer distance to reach the goal. In contrast, ORCA behaves extremely cautious and spend a large amount of time carefully calculating the shortest path. DRLMACA and our method achieve a better comprise of these two extremes with our method outperforms DRLMACA.

Also, from Figure 6, our method significantly outperforms the the remaining methods, with higher speed, shorter time and distance. We also see a gradient pattern within our three updating rules across all 4 histograms: the *discrete* performs

best, followed by *linear* and *sigmoid*. It indicates that sharp intensity change brings more robustness to mobile robot.

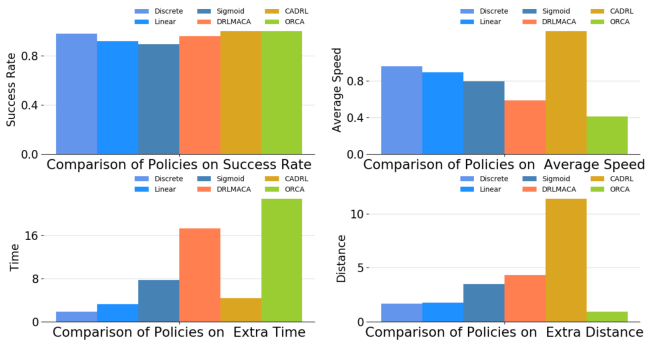


Fig. 6: Performance metrics in non-adversarial scenario. We see that all methods achieve high success rate, but our methods performs better on extra time and extra distance.

2) Adversarial Scenario: Having demonstrated the superior performance of our method in normal setting, we now compare the policies learnt using our method with the three baselines in the adversarial scenarios described in Figure 3. In these experiments, the protagonist is controlled by each policy being compared while the adversaries are controlled by the adversarial policies obtained from adversarial training described in Section IV-C.

We present the comparison results in Figure 7. It is easy to see that our method outperforms all other baseline methods in the adversarial environments and behaves especially competitive in the chasing paradigm. This is due to the fact that the other baseline methods rarely consider the scenario that the adversarial agents might move towards the protagnist from behind.

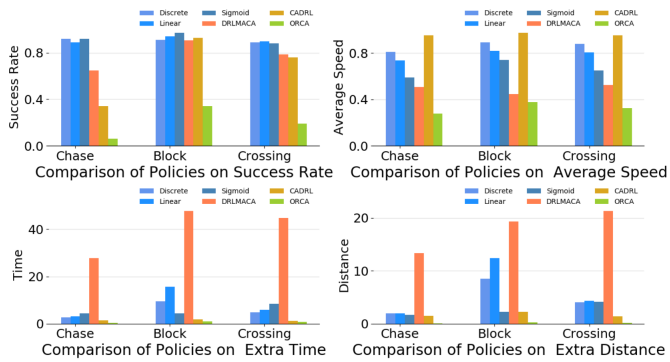


Fig. 7: Performance metrics in adversarial scenarios. Our method performs the best in all four metrics while DRLMACA has abnormal behaviours when confronting adversarial agents.

ORCA behaves badly as expected since its reciprocal assumption of t he adversarial agents is violated. In contrast, CADRL handles the adversarial environment pretty well since the protagonist has perfect knowledge of the adversaries' intents. Compared to the non-adversarial scenario, CADRL still achieves the highest speed but with a lower success rate. It's worth noting that DRLMACA has an abnormally high extra distance and travel time. It seems terribly frightened once the nearby agents deliberately move towards it rather than actively moving away.

E. Real-World Experiments

To test the transferability of our trained policy to physical robots in the real world, we deploy the policy trained using the *Discrete* updating rule on a mobile robot platform `Turtlebot3 waffle_pi`. This robot is equipped with LDS-01, a 2D LIDAR sensing 360 degrees with 0.12 to 3.5 meter range and a scan rate of 300 ± 10 rpm. The robot is controlled with the maximum speed 1m/s. The on-board computer of `waffle_pi` is `Raspberry Pi 3` with Robot Operating System (ROS) installed. Our control policy samples actions at a rate of 50Hz. In the real-world deployment, we first place the robot in an environment with dense pedestrians randomly moving around, then test the policy when several humans try to block it from different directions as shown in Figure 8. For the experiments, we randomly generate goal positions for the robot. Using our the learnt policy, the robot navigates very smoothly in many challenging scenarios, demonstrating the practical efficacy of our method.

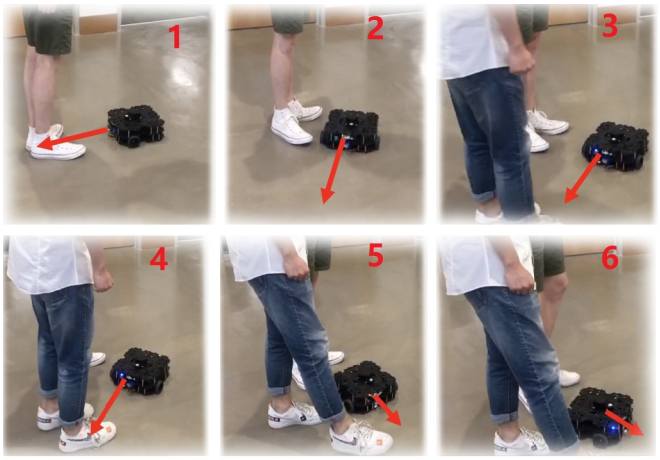


Fig. 8: A real-world deployment. We deploy our collision avoidance policy on `turtlebot3`. The red arrows represent the moving directions of the robot. The result shows that when two pedestrians try to maliciously block the robot, our trained collision avoidance policy still find a collision-free path for the robot.

V. CONCLUSIONS

We propose to learn robust collision-avoidance policies via Markov games. By introducing multiple auxiliary agents to model a competing environment. The experiments show that our path-following adversarial policy learning strategy significantly improves the sample complexity and adaptivity of the trained primary agent (i.e., the robot). Promising experimental results on a physical robot also demonstrate that the superior performance of the learnt policy is generalized to new scenarios. Future work will consider combining tree-based random search global planning algorithms with our learnt policy to navigate robots in crowded scenarios.

Acknowledgement We sincerely thank Jiayi Wang and Zhihan Zhou from the Northwestern University for providing very helpful discussions and graphics materials in this paper.

REFERENCES

[1] R. Riener, L. Lünenburger, and G. Colombo, "Human-centered robotics applied to gait training and assessment." *Journal of Rehabilitation Research & Development*, vol. 43, no. 5, 2006.

[2] C.-P. Lam, C.-T. Chou, K.-H. Chiang, and L.-C. Fu, "Human-centered robot navigationtowards a harmoniously human–robot coexisting environment," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 99–112, 2010.

[3] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[4] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[6] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[7] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[8] H. Bharadhwaj, Z. Wang, Y. Bengio, and L. Paull, "A data-efficient framework for training and sim-to-real transfer of navigation policies," *arXiv preprint arXiv:1810.04871*, 2018.

[9] H. Jun, H. Kim, and B. Lee, "Goal-driven navigation for non-holonomic multi-robot system by learning collision," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1758–1764.

[10] A. Francis, A. Faust, H.-T. L. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T.-W. E. Lee, "Long-range indoor navigation with prm-rl," *arXiv preprint arXiv:1902.09458*, 2019.

[11] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[12] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[13] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *International Conference on Robotics and Automation (ICRA)*, 2018.

[14] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2347–2354.

[15] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[16] 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.

[17] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[18] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.

[19] M. L. Littman, "Friend-or-foe q-learning in general-sum games," in *Conference on Artificial Intelligence (AAAI)*, 2001.

[20] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team markov games," in *Advances in neural information processing systems*, 2003, pp. 1603–1610.

[21] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *Journal of machine learning research*, vol. 4, no. Nov, pp. 1039–1069, 2003.

[22] P. Casgrain, B. Ning, and S. Jaimungal, "Deep q-learning for nash equilibria: Nash-dqn," *arXiv preprint arXiv:1904.10554*, 2019.

[23] 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.

[24] L. Bu, R. Babu, B. De Schutter *et al.*, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[25] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," *arXiv preprint arXiv:1707.09183*, 2017.

[26] M. Weinberg and J. S. Rosenschein, "Best-response multiagent learning in non-stationary environments," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*. IEEE Computer Society, 2004, pp. 506–513.

[27] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling others using oneself in multi-agent reinforcement learning," *arXiv preprint arXiv:1802.09640*, 2018.

[28] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 281–302, 2000.

[29] H. Pang, H. Liu, R. J. Vanderbei, and T. Zhao, "Parametric simplex method for sparse learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 188–197.

[30] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[31] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[32] 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.

[33] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2817–2826.

[34] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[35] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv preprint arXiv:1802.08757*, 2018.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[37] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," *arXiv preprint arXiv:1710.03748*, 2017.

[38] A. Gleave, M. Dennis, N. Kant, C. Wild, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," *arXiv preprint arXiv:1905.10615*, 2019.

[39] X. Pan, D. Seita, Y. Gao, and J. Canny, "Risk averse robust adversarial reinforcement learning," *arXiv preprint arXiv:1904.00511*, 2019.

[40] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[41] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

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

[43] 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.

[44] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.