# Map-Based Deep Reinforcement Learning for Collision Avoidance of Ackermann-Steering Vehicles

Guoliang You[1,*], Yujie Yang[1], Tingting Jiang[1], Xu Li[1], Xingchen Li[2]

***Abstract*— It is challenging to develop a collision avoidance algorithm for Ackermann-steering vehicles to find collision-free and kinematically-feasible paths in various environments with both static and moving obstacles. In this paper, we propose an end-to-end deep reinforcement learning method for collision avoidance of Ackermann-steering vehicles. We use the egocentric local grid map of a vehicle to represent its immediate environmental information, which specifies its shape and observable appearances of obstacles. Then we apply the distributed proximal policy optimization (DPPO) algorithm with curriculum learning to train a convolutional neural network that directly maps three frames of such grid maps and the position of the vehicle's target into a drivable trajectory. We train the neural network in simulation environments with static obstacles and multiple differential-drive robots, which are navigated by the modified version of the network by converting the outputs to robot's control commands, i.e., linear velocities and angular velocities. Then we deploy the trained model to a vehicle to perform collision avoidance in the real world. We evaluate the approach with multiple scenarios both in the simulation and the real world. Both qualitative and quantitative experiments show that the approach is effective with a high success rate.**

## I. Introduction

Collision avoidance for Ackermann-steering vehicles [1] is to efficiently find a collision-free and kinematically-feasible path to the target in various environments with both static and moving obstacles, which is one of the major challenges for multiple autonomous applications, like self-driving cars [2] and delivering vehicles [3]. Although numerous collision avoidance methods have been proposed, they suffer from several common limitations in practice [4]. For instance, assumptions of the method may not hold in every environment [5], intensive computational demands are imposed by some methods [6], difficult and time-consuming manual parameter tuning is required to deploy the method [7], and it is difficult for the method to learn from past experiences [8]. Moreover, multiple robots with different shapes and different kinematic constraints collision avoidance in a distributed and communication-free scenario is a much more challenging task. This also allows each robot to obtain better dynamic obstacle avoidance capabilities. Such as Optimal reciprocal collision avoidance (ORCA) [9] and Bicycle reciprocal collision avoidance (B-ORCA) [10] provides a sufficient condition for multiple robots with the same kinematic constraints to avoid collisions with each other, which cannot be applied under different kinematic constraints. Generalized reciprocal collision avoidance [11] applied to robots with different kinematic constraints (differential-drive, car-like, etc.) to achieve collision avoidance, and solves the problem that can only be applied to the same dynamic constraints.

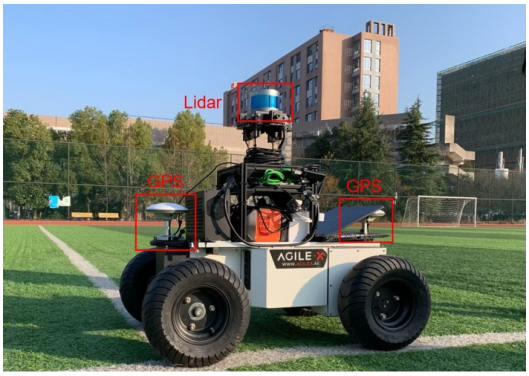



Fig. 1: The Ackermann-steering vehicle.

In order to overcome these limitations and improve robots obstacle avoidance, deep reinforcement learning (DRL) approaches for collision avoidance have been proposed as a possible solution with promising results [12]. However, training multiple differential robots in this way can achieve better obstacle avoidance capabilities, but it can only be applied to obstacle avoidance between robots with the same dynamic constraints, which limits the scope of application to a certain extent.

According to the difference between the networks' inputs, existing DRL methods can be roughly divided into three categories: agent-based, sensor-based, and map-based. In particular, an agent-based method [13] takes into account positions and movement data, like velocities or accelerations, of other robots and obstacles, which assumes the perfect sensing of the surroundings and is hard to be implemented in the real world. A sensor-based method uses the sensor data, like laser scan data [14], as the inputs of the network, which only works for a certain type of sensors. A map-based method considers an intermediate representation of the surrounding environment, like local grid maps [15], [12], which can be easily generated from multiple sensor data or sensor fusion results. Compared to other methods, the map-based approach is more robust to noisy sensor data, does not require robots' movement data, easy to be trained in a simulator, and considers sizes and shapes of related robots,

This work is partially supported by the 2030 National Key AI Program of China 2018AAA0100500, the National Natural Science Foundation of China (No. 61573386), and Guangdong Province Science and Technology Plan Projects (No. 2017B010110011).

[1]Robotics Lab, University of Science and Technology of China,Hefei 230026, China

[1]Lab for Intelligent Networking and Knowledge Engineering, University of Science and Technology of China,Hefei 230026, China

which make it to be more efficient, robust, and easier to be deployed to real robots.

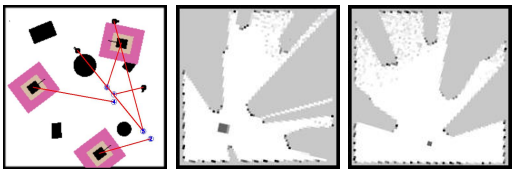
In this paper, inspired by B-ORCA, we propose a map-based DRL approach for collision avoidance of Ackermann-steering vehicles, so that Ackermann-steering vehicles has better dynamic obstacle avoidance capabilities. We use the egocentric local grid map of the vehicle to represent its immediate environmental information, which specifies its shape and observable appearances of obstacles. Then we apply the distributed proximal policy optimization (DPPO) algorithm to train a convolutional neural network that directly maps three frames of egocentric local grid maps and the position of the vehicle's local target into a drivable trajectory. In specific, the network outputs both the curvature of the trajectory and the expected acceleration of the vehicle, which would be further processed by the trajectory tracking system of the vehicle to obtain the corresponding control commands of the specified vehicle. Notice that, it is more robust for an Ackermann-steering vehicle to track trajectories by its fine-tuned trajectory tracking system, than directly execute low-level commands learned from the simulator. We first train the neural network in simulation environments with only static obstacles. Then we train the network in environments with other vehicles and multiple differential-drive mobile robots, which are navigated by the modified version of the network by converting the outputs to robot's control commands, i.e., linear velocities and angular velocities. At last, we deploy the trained model to a real vehicle to perform collision avoidance in its navigation without tedious parameter tuning.

Note that, it is not easy to learn a collision avoidance policy for dynamic obstacles from simulation environments. If these dynamic obstacles are navigated by a specified program, then the learned policy might be vulnerable to moving obstacles that behave differently. However, we show that by training the policy in environments with other vehicles and differential-drive robots, where all of them are navigated by the same network, the performance of collision avoidance for various moving obstacles can be greatly improved.

We also introduce two strategies to accelerate and stabilize the training process. First, we apply the reward shaping technique [16] by adding stepped penalties for obstacles in the warning or danger zone of the vehicle. Then we use a multi-stage parallel curriculum learning strategy [17] to speed up and optimize the training.

We evaluate the approach with multiple scenarios both in simulation and the real world. Experimental results show that the approach is effective with a high success rate. We also conduct ablation studies to show the positive effects of applying our improvements. The demonstration video can be found at https://youtu.be/_KwpzBEMqjs. Our main contributions are summarized as follows:

- We propose a map-based DRL approach for collision avoidance of Ackermann-steering vehicles, which maps local grid maps of the vehicle into a drivable trajectory. The experimental results show that the approach is effective and easy to be deployed with a high success rate.



(a) (b) (c)

Fig. 2: (a) A simulation environment, where the blue digital circles represent the target positions of corresponding vehicles or robots, black blocks in magenta blocks denote vehicles, and black spots denote robots. (b) The egocentric local grid map for the vehicle 0. (c) The corresponding map for the robot 1.

- We introduce a specified reward shaping strategy and a multi-stage parallel curriculum learning strategy to accelerate and stabilize the training process. The ablation studies explore the positive effects of both strategies.
- We train the collision avoidance policy in simulation environments with other vehicles and differential-drive robots. All of these vehicles and robots are navigated by the same network. The experimental results show that such training environments can greatly improve the performance of collision avoidance for dynamic obstacles.

The rest of this paper is organized as follows. The formulation of the collision avoidance problem and the details of the DRL approach are described in Section II. Section III presents experimental results, followed by conclusions in Section IV.

## II. Approach

We first provide a formulation of the collision avoidance problem. Then we introduce the DPPO algorithm with our improvements. At last, we specify details on deploying the trained model to a real vehicle.

### A. Problem Formulation

We specify the collision avoidance problem as a Partially Observable Markov Decision Process (POMDP) problem [18], i.e., the tuple $\langle S, A, P, R, \Omega, O \rangle$.

We introduce imitation learning in our experiments to form the Prompt imitation learning subtask. Among them, imitation learning can quickly learn the target task through demonstrations given by experts. The policy finally learned by imitation learning is$\pi_\theta\left(a_t \mid s_t\right)$, where $s$ is the state at time $t$ and $a$ is the behavior at time $t$. The demonstration given by the expert is that there are multiple sets of trajectory $\mathcal{D}=\{\tau_1, \tau_2, \ldots, \tau_m\}$, and each trajectory contains $\tau_i =< s_1^i, a_1^i, s_2^i, a_2^i, \ldots, s_{n_n i+1}^i >$. The model needs to be trained on expert data, in order to make the state-action trajectory distribution $p_{\pi_\theta}\left(s_t\right)$ generated by the model match the input trajectory distribution $p_{\text{data}}\ \left(s_t\right)$. Using imitation learning can quickly learn the target task and achieve better performance.

In specific, an observation $o \in \Omega$ received by the vehicle (resp. robot) consists of the triple $(M, g, \alpha)$, where $M$ denotes its egocentric local grid map, $g$ denotes the position of its target, and $\alpha$ denotes its heading angle. Note that, $M$

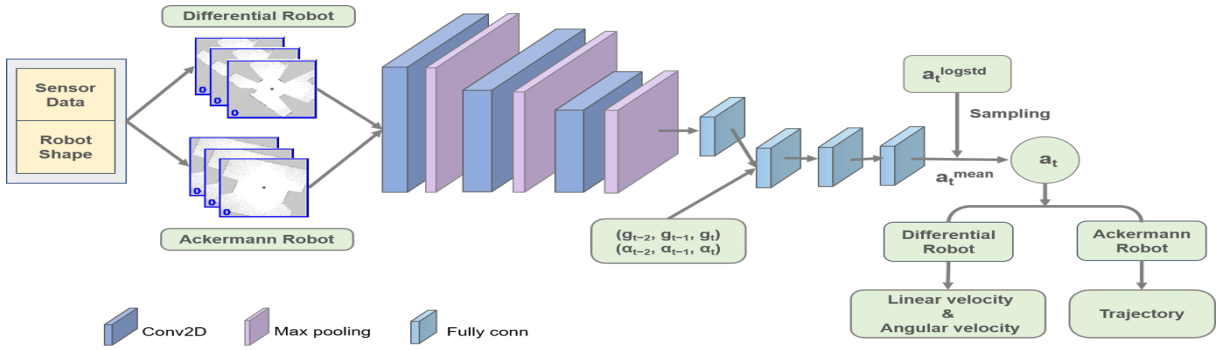
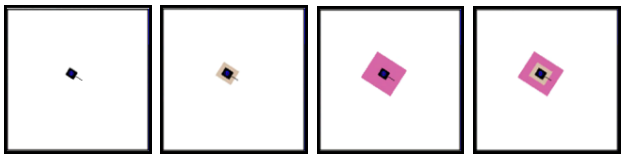



Fig. 3: The architecture of the policy network.



(a) (b) (c) (d)

Fig. 4: (a) A vehicle in the simulation environment. (b) The danger zone of the vehicle. (c) The warning zone of the vehicle. (d) The combination of both zones.

can easily generated from its local costmap[1]. Figure 2 shows corresponding egocentric local grid maps for a vehicle and a robot in a simulation environment.

We specify different action spaces for Ackermann-steering vehicles and differential-drive robots. An action $a \in A$ for a vehicle is a pair $(c, \sigma)$, where $c$ denotes the curvature of a trajectory and $\sigma$ denotes the expected acceleration for vehicle. Note that, $(c, \sigma)$ specifies a trajectory that should be followed by the vehicle, which would be further processed by the trajectory tracking system of the vehicle to obtain the corresponding control commands. In our implementation, we set $c \in [-1.43, 1.43]$ $(m^{-1})$ and $\sigma \in [-11.25, 11.25]$ $(m/s^2)$. On the other hand, an action for a robot is pair $(v, \omega)$, where $v$ and $\omega$ denote expected line and angular velocities of the robot respectively, which can be executed by the robot directly. In our implementation, we set $v \in [0, 0.6]$ $(m/s)$ and $\omega \in [-0.9, 0.9]$ $(s^{-1})$.

### B. Distributed Proximal Policy Optimization

*1) Reward Shaping:* We first introduce the reward function for a robot, then we apply reward shaping for a vehicle by adding stepped penalties w.r.t its warning and danger zones. In specific, the reward function for a differential-drive robot is defined as follows:

$$r_r = r^g + r^c + r^s, \tag{1}$$

$$r^g = \begin{cases} r_{arr} & \text{if } \|p_t - g\| < 0.6, \\ \varepsilon\left(\|p_{t-1} - g\| - \|p_t - g\|\right) & \text{otherwise,} \end{cases} \tag{2}$$

$$r^c = \begin{cases} r_{col} & \text{if collision,} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $r_{arr} > 0$, $p_t$ denotes the position of the robot at the current time step $t$, $\varepsilon$ is a hyper-parameter, and $r^g$ denotes the reward for arriving the target and the penalty for departing the target. $r_{col} < 0$ and $r^c$ denotes the penalty for the collision. As last, we apply a small negative penalty for each time step, i.e., $r^s < 0$, to encourage short paths.

As illustrated in Figure 4, we define a waring and a danger zone of an Ackermann-steering vehicle for reward shaping.

Then the reward function for an Ackermann-steering vehicle is defined as follows:

$$r_v = r^g + r^c + r^s + r^w + r^d, \tag{4}$$

$$r^w = \begin{cases} r_{warn} & \text{if obstacles in the warning zone,} \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

$$r^d = \begin{cases} r_{danger} & \text{if obstacles in the danger zone,} \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

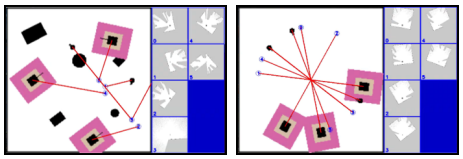
where $r_{warn}, r_{danger} < 0$, $r^w$ (resp. $r^w + r^d$) denotes the penalty when there were obstacles in the warning (resp. danger) zone of the vehicle.

In our implementation, we set $r_{arr} = 500$, $\varepsilon = 10$, $r_{col} = -500$, $r^s = -5$, $r_{warn} = -20$, and $r_{danger} = -10$.

In this paper, we apply the Distributed Proximal Policy Optimization (DPPO) algorithm [19] to train the stochastic policy $\pi_\theta(a \mid o)$ of collision avoidance for Ackermann-steering vehicles. DPPO is extended from PPO by collecting experiences in a distributed setting from a variety of environments where multiple vehicles and robots share the same policy $\pi_\theta$ while take different actions. We specify details of the network architecture and the training process in the following.

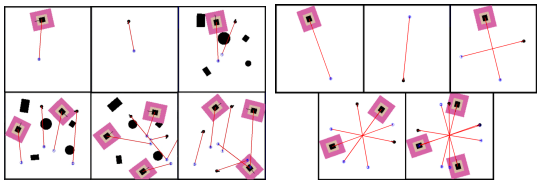
*2) Network Architecture:* The architecture of the convolutional network for the collision avoidance policy $\pi_\theta$ in DPPO is shown in Figure 3. The input of the network consists of three frames of observations, i.e., egocentric local grid maps,

[1]http://wiki.ros.org/costmap_2d.



(a) Random scenario (b) Circular scenario

Fig. 5: (a) Random scenario: environments with randomly located obstacles and multiple vehicles and robots. (b) Circular scenario: environments with randomly placed vehicles and robots on a circle.



(a) Random scenario (b) Circular scenario

Fig. 6: Training scenarios with different difficulties.

positions of targets, and heading angles. The network outputs the mean of the action, which is sampled from a Gaussian distribution. We use different clip functions to convert the result to the corresponding action for the Ackermann-steering vehicle or the differential-drive robot, respectively. Then both vehicles and robots can share the same network during the training, while converting the results to corresponding actions.

*3) Multi-Stage Parallel Curriculum Learning:* The policy network needs to be trained in simulation environments of two scenarios illustrated in Figure 5, where random scenario helps the vehicle to be able to avoid obstacles and circular scenario helps the vehicle to be able to interact with others. However, as shown in experiments, it is hard to learn a policy by directly applying DPPO on these environments.

We introduce a multi-stage parallel curriculum learning strategy to accelerate and stabilize the training process. In specific, we first train the network on environments in random scenario by parallelly training the network on a series of scenarios with different difficulties as illustrated in Figure 6(a). Once the network has achieved a good performance on these scenarios, we start the next stage and train the network on environments in circular scenario by parallelly training the network on a series of scenarios with different difficulties as illustrated in Figure 6(b).

### C. Deployment on Vehicle

As shown in Figure 1, the Ackermann-steering vehicle has a 3D LiDAR sensor to generate point clouds[2] of the surrounding environment. Then the costmap converted from the point clouds can construct the egocentric local grid map. Meanwhile, the vehicle has a RTK-GNSS[3] receiver which provides the target position, the vehicle's position and heading angle.

[2]http://wiki.ros.org/pcl.
[3]http://wiki.ros.org/rtklib.

Given an action $(c, \sigma)$, the vehicle tracks the corresponding trajectory with the pure pursuit method[4], whose parameters have been tuned for the specified vehicle. We also implement a safety system for the real vehicle to stop it if there were obstacles in its danger region.

| Parameter | Value |
|---|---|
| learning rate for policy network | $1.0 \times 10^{-3}$ |
| learning rate for value function | $3.0 \times 10^{-4}$ |
| training iterations for policy network | 80 |
| training iterations for value function | 80 |
| image size | $48 \times 48$ |
| episode length | 5000 |

TABLE I: Hyper-parameters of our training algorithm.

## III. Experiments

In this section, we evaluate the approach with multiple scenarios both in simulation and the real world. The demonstration video can be found at https://youtu.be/_KwpzBEMqjs.

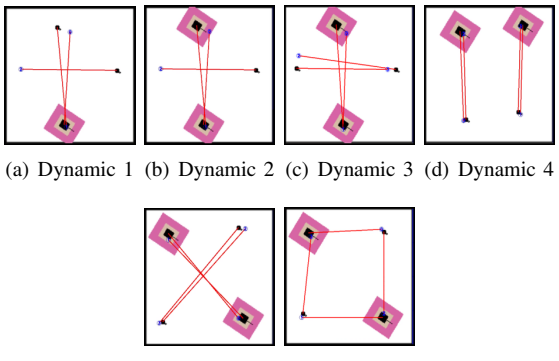
### A. Reinforcement Learning Setup

We trained our collision avoidance policy for the Ackermann-steering vehicle following the DPPO algorithm with the hyper-parameters listed in Table I.

The training environments are constructed by a customized simulator based on OpenCV[5]. Both the policy network and the value network are implemented in Pytorch[6] and trained with the Adam optimizer [20]. The training hardware is a computer with an i7-10700 CPU and a single NVIDIA GeForce RTX 3090 GPU. The entire training process takes about 35 hours for the policy to achieve a good performance.

We use success rate, i.e., the ratio of the episodes that end with the vehicle reaching its target without any collision, and expected return, i.e., the average of the sum of rewards of episodes, to evaluate the performance of the collision avoidance policies for different approaches.

### B. Simulation Experiments



(a) Dynamic 1 (b) Dynamic 2 (c) Dynamic 3 (d) Dynamic 4

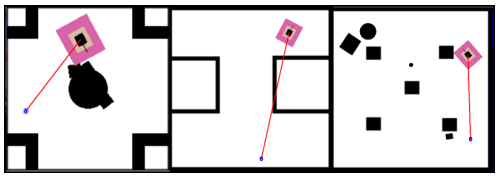
(e) Dynamic 5 (f) Dynamic 6

Fig. 7: Testing scenarios with randomly placed vehicles and robots.

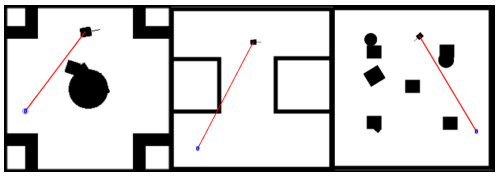
[4]http://wiki.ros.org/purepursuit_planner.
[5]https://opencv.org/.
[6]https://pytorch.org/.

| Scenario | Success rate | Scenario | Success rate |
|---|---|---|---|
| Dynamic 1 | 95.9% | Dynamic 4 | 82% |
| Dynamic 2 | 88.2% | Dynamic 5 | 94.8% |
| Dynamic 3 | 89.1% | Dynamic 6 | 90.4% |
| Random | 93.3% | Circular | 91.3% |

TABLE II: Performance of the trained model for dynamic scenarios.



(a) Shaping



(b) Non-shaping

Fig. 8: Static training scenarios for "Shaping" and "Non-shaping".

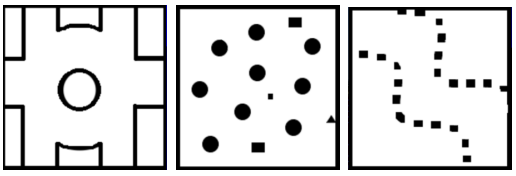
Notice that, we only use environments in the random scenario and circular scenario for the training. Here we evaluate the performance of the trained model on multiple unseen scenarios as illustrated in Figure 7. The experimental results are summarized in Table II, which are calculated from the averaging results of 500 randomly constructed environments for corresponding scenarios. In particular, the results for the random and circular scenario in Table II are calculated on newly generated environments in both scenarios. These results show that the trained policy performs well in testing scenarios with a high success rate.

### C. Ablation Studies

We conduct ablation studies here to evaluate the effects of using our reward shaping and the multi-stage parallel curriculum learning strategies.

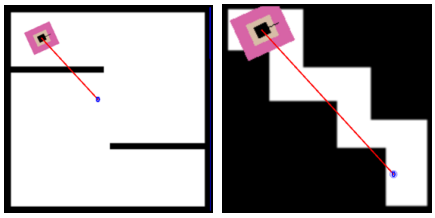
Notice that, the experiments were performed for scenarios with only static obstacles, as DPPO failed to converge for dynamic scenarios if either of these strategies were not adopted. Moreover, DPPO also failed to converge for these static scenarios if none of these strategies were adopted. We first consider the reward shaping strategy. We use "Shaping" to denote the DPPO approach with our reward shaping and "Non-shaping" to denote the approach without the reward shaping. Note that, multi-stage parallel curriculum learning is applied in both approaches. We construct scenarios with randomly placed static obstacles for the training of both approaches as shown in Figure 8. We also construct static scenarios for the testing as shown in Figure 9. The experimental results are summarized in Table III,

which are calculated from the averaging results of 200 randomly constructed environments for corresponding scenarios. These results show that the reward shaping strategy is important for the performance.



(a) Test 1 (b) Test 2 (c) Test 3

Fig. 9: Static testing scenarios for "Shaping" and "Non-shaping".



(a) Corridor 1 (b) Corridor 2

Fig. 10: Training scenarios for "Curr" and "Non-curr".

Now we consider the multi-stage parallel curriculum learning strategy. We use "Curr" to denote the DPPO approach with multi-stage parallel curriculum learning and "Non-curr" to denote the one without. Note that, the reward shaping is also applied in both approaches. Similarly, we construct two scenarios for the training of both approaches as shown in Figure 10. "Non-curr" has already failed to converge for these training scenarios. The experimental results are summarized in Table IV, which are calculated from the averaging results of 100 tests for corresponding scenarios. These results show that multi-stage parallel curriculum learning is crucial for the performance.

### D. Real-World Experiments

In this section, we deploy the trained collision avoidance policy to an Ackermann-steering vehicle, as shown in Figure 1, in the real world. In specific, the vehicle has a 16 laser-beam LiDAR, a RTK-GNSS receiver, an IPC with an i7-8700 CPU and a NVIDIA 2080Ti GPU. The LiDAR is used to generate the egocentric local grid map with the size $6 \times 6$ $(m^2)$ and the resolution $0.1$ $(m^2)$ for the size of a cell.

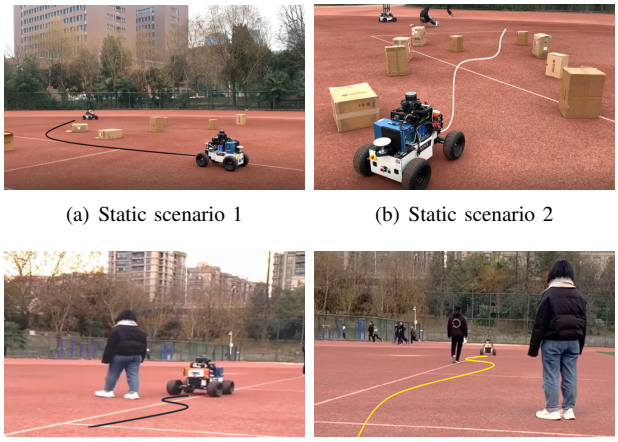
We introduce a series of real-world tests for the vehicle to evaluate the performance of our map-based DPPO approach. We place paper boxes as static obstacles and consider walking pedestrians as moving obstacles in the environment. Figure 11 illustrates the performance of the vehicle in scenarios with static and moving obstacles.The

| Scenario | Method | Success rate | Expected return |
|---|---|---|---|
| Test 1 | Non-shaping | 78% | 74.3 |
| | Shaping | **100%** | **276** |
| Test 2 | Non-shaping | 82% | 44.7 |
| | Shaping | **88.5%** | **133** |
| Test 3 | Non-shaping | 78.5% | 63.6 |
| | Shaping | **87%** | **131** |

TABLE III: Performance of "Non-shaping" and "Shaping".

| Scenario | Method | Success rate | Expected return |
|---|---|---|---|
| Corridor 1 | Non-curr | 0% | −732 |
| | Curr | **99%** | **118** |
| Corridor 2 | Non-curr | 0% | −524 |
| | Curr | **80%** | **1.03** |

TABLE IV: Performance of "Non-curr" and "Curr".



(a) Static scenario 1 (b) Static scenario 2

(c) Dynamic scenario 1 (d) Dynamic scenario 2

Fig. 11: Experiments in the real world.

experimental result shows that the trained model can be easily deployed to an Ackermann-steering vehicle to perform collision avoidance in environments with static and moving obstacles. The demonstration video can be found at https://youtu.be/_KwpzBEMqjs.

## IV. Conclusions

In this paper, we propose a map-based DRL approach for collision avoidance of Ackermann-steering vehicles in environments with static and moving obstacles. We use the egocentric local grid map of the vehicle to represent the environmental information around it including its shape and observable appearances of obstacles, robots, and other vehicles, which can be easily generated by using multiple sensors or sensor fusion. Then we apply DPPO to train a convolutional neural network that directly maps three frames of egocentric local grid maps and the positions of the vehicle's targets into a collision-free and drivable trajectory, which would be tracked by the vehicle. We apply multi-stage parallel curriculum learning to train networks using simulation environments in the random scenario and circular scenario, where a specified reward shaping is used to accelerate and stabilize the training process. At last, we deploy the trained model to the vehicle to perform collision avoidance in its navigation without tedious parameter tuning.

We evaluate the approach with multiple scenarios both in simulation and the real world. Experimental results show that the approach performs well to unseen scenarios with a high success rate. We also conduct ablation studies showing the positive effects of applying our improvements. These experiments show that our approach is effective, easy to be deployed to an Ackermann-steering vehicle, and performs well in the real world.

For future work, we plan to deploy the train model to multiple Ackermann-steering vehicles and differential-drive mobile robots of different shapes in distributed and communication-free scenarios. We intend to evaluate the performance of the approach for heterogeneous multi-robot collision avoidance in the real world.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

## References


[1] J. L. Blanco, M. Bellone, and A. Gimenez-Fernandez, "Tp-space rrt–kinematic path planning of non-holonomic any-shape vehicles," *International Journal of Advanced Robotic Systems*, vol. 12, no. 5, p. 55, 2015.

[2] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[3] Y. Dong, Y. Zhang, and J. Ai, "Experimental test of unmanned ground vehicle delivering goods using rrt path planning algorithm," *Unmanned Systems*, vol. 5, no. 01, pp. 45–57, 2017.

[4] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[5] W. Zhang, S. Wei, Y. Teng, J. Zhang, X. Wang, and Z. Yan, "Dynamic obstacle avoidance for unmanned underwater vehicles based on an improved velocity obstacle method," *Sensors*, vol. 17, no. 12, p. 2742, 2017.

[6] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[7] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[8] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–8.

[9] 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.

[10] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 360–366.

[11] 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.

[12] G. Chen, S. Yao, J. Ma, L. Pan, Y. Chen, P. Xu, J. Ji, and X. Chen, "Distributed non-communicating multi-robot collision avoidance via map-based deep reinforcement learning," *Sensors*, vol. 20, no. 17, p. 4836, 2020.

[13] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1343–1350.

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

[15] P. Regier, L. Gesing, and M. Bennewitz, "Deep reinforcement learning for navigation in cluttered environments," 2020.

[16] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, I. Bratko and S. Dzeroski, Eds. Morgan Kaufmann, 1999, pp. 278–287.

[17] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[18] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[19] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.