Deep reinforcement learning oriented for real world dynamic scenarios

Diego Martínez, Luis Riazuelo and Luis Montano[1]

***Abstract*— Autonomous navigation in dynamic environments is a complex but essential task for autonomous robots. Recent deep reinforcement learning approaches show promising results to solve the problem, but it is not solved yet, as they typically assume no robot kinodynamic restrictions, holonomic movement or perfect environment knowledge. Moreover, most algorithms fail in the real world due to the inability to generate real-world training data for the huge variability of possible scenarios. In this work, we present a novel planner, DQN-DOVS, that uses deep reinforcement learning on a descriptive robocentric velocity space model to navigate in highly dynamic environments. It is trained using a smart curriculum learning approach on a simulator that faithfully reproduces the real world, reducing the gap between the reality and simulation. We test the resulting algorithm in scenarios with different number of obstacles and compare it with many state-of-the-art approaches, obtaining a better performance. Finally, we try the algorithm in a ground robot, using the same setup as in the simulation experiments.**

I. Introduction

Motion planning and navigation in dynamic scenarios is a complex problem that has not a defined solution. Traditional planners fail in environments where the map is mutable or obstacles are dynamic, leading to suboptimal trajectories or collisions. Those planners typically consider only the current obstacles' position measured by the sensors, without considering the future trajectories they may have.

New approaches that try to solve this issue include promising learning based methods. Nevertheless, they do not work properly in the real world: They do not consider robot kinodynamic constraints, only consider dynamic obstacles or assume perfect knowledge of the environment. Moreover, they would need huge real-world data to train the algorithms for the real world, and generating it is not possible.

We propose a planner that is able to navigate through dynamic and hybrid real-world environments. The planner is based on the Dynamic Object Velocity Space (DOVS) model, presented in [1], which reflects the scenario dynamism information. In that work, the kinodynamics of the robot and the obstacles of the environment are used to establish the feasible velocities of the robot that do not lead to a collision. In our approach, the DOVS model is used in a new planner called DQN-DOVS, which utilizes deep reinforcement learning techniques. The planner applies the rich information provided by the DOVS as an input, taking advantage over other approaches that use raw sensor measurements and are not able to generalize. Once the agent learns how to interpret the DOVS, it is able to navigate in any scenario (it does not need real-world data of a huge variety of scenarios); and the training weights learned in the simulated world work as well in real-world environments without fine-tuning. In addition, it uses a dynamic window in the robot velocity space to define the set of actions available keeping robot kinodymanics.

Fig. 1. Scenario of a robot with static and dynamic obstacles and the RVIZ visualization of the scenario sensed.

The DQN-DOVS algorithm is trained and tested in a real-world simulator, where all information is extracted from the sensor measurements, even the own robot localization. A comparison of the model with other planners of the state-of-the-art is also provided, as well as other experiments working with a real robot, like in Figure 1.

II. Background

A. Related work

Motion planners of static and continuous environments may not be used to deal with dynamic obstacles, as they lead to collisions and suboptimal trajectories. Some traditional approaches for dynamic environments include artificial potential fields [2], are probability based [3] or use a reciprocal collision avoidance (ORCA) [4].

A big group of works are velocity space based. The *Velocity Obstacle (VO)*, introduced in [5], refers to the set of velocities of the robot that could lead to collide with an obstacle that has a certain velocity in the near-future, which should not be chosen. Based on the VO concept, the *Dynamic Object Velocity Space (DOVS)* is defined in [1] as the velocity-time space for non-holonomic robots, which includes the unsafe robot velocities for all the obstacles and the time to collision for computing safe robot velocities in a time horizon. In the work, a planner based on strategies is also defined, the S-DOVS. A planner based on basic reinforcement learning on top of the DOVS is also proposed in [6], making decisions based on Q-values stored in tables.

[1]The authors are with the Robotics, Perception and Real Time Group, Aragon Institute of Engineering Research (I3A), Universidad de Zaragoza, 50018 Zaragoza, Spain. diegomartinez,riazuelo,montano@unizar.es

Reinforcement learning is a method used to learn to estimate the optimal policy that optimizes the cumulative reward obtained in an episode. In [7], the Q-values are estimated with a deep neural network, defining the first Deep Q-Network (DQN). Many extensions have been proposed to this original algorithm. Some works have proven the best performances of the state-of-the-art, including DQN with multiple modifications [8], distributed reinforcement learning [9] or actor-critic methods [10]. The study presented in [11] shows that combining reinforcement learning with curriculum learning, could give useful results, specially to learn problems that could be too difficult to learn from scratch.

Some works offer analysis of the importance of reinforcement learning in robot motion planning and the limitations of traditional planners in dynamic environments. Defining strategies for every situation that may be found in the real world is intractable, and reinforcement learning may be used to solve the decision-making problem, which is complex and has many degrees of freedom. [12] proposes a deep reinforcement learning model that takes as the input of the model LIDAR measurements and the position of the goal, obtaining better results than conventional planners.

The work described in [13] (SARL) simulates a crowd and try to make the robot anticipate the crowd interactions with other robots and with each others, comparing its method with ORCA [4] and other two deep reinforcement learning methods: CADRL [14] and LSTM-RL [15]. ORCA fails in this crowded environment due to the need of the reciprocal assumption, and CADRL fails because it does not take into account the whole crowd, just a single pair for interaction. In the environment presented, both SARL and LSTM-RL have the best performance. In all of these approaches, the simulators used are non-realistic and no kinodynamic restrictions are considered.

An example of an approach that considers the restrictions is [16], which combines deep reinforcement learning with DWA [17], but only achieving a success rate of 0.54 in sparse dynamic scenarios.

B. Dynamic Object Velocity Space (DOVS)

The DOVS model presented in [1], which models the dynamism of the environment, is used as a basis of this work. To build the model, the robot size is reduced to a point and the obstacles are enlarged with the robot radius (the final collision areas are the same). The area swept by each moving obstacle (collision band) is computed using the trajectory of the obstacles, which is assumed to be known or estimated from the sensor information. Then, the maximum and minimum velocities that can avoid a collision with that obstacle are calculated, repeating the process for every obstacle. Using that information with maximum and minimum velocities of the robot the limits of the *Dynamic Object Velocity* (DOV) are obtained. The key of this model is that the set of free velocities is recomputed in every time step, so the obstacles' trajectory estimation needs to be precise only for the next few time steps.

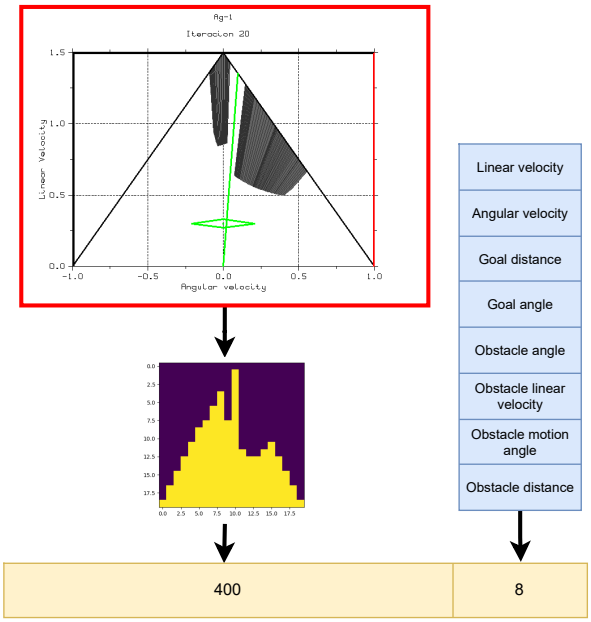


Fig. 2. Representation of the state of the agent. The DOVS is in the red square and the extracted velocity grid below, concatenated to other 8 robot variables to construct the 408 elements input vector of the learning system.

The velocities in the DOV are unsafe, as they lead to collisions in a time horizon, while the rest of the velocities are available to navigate. Velocities inside the DOV could be chosen if the following commands lead to a free velocity. The navigation in that space is achieved using a dynamic window that considers robot kinodynamics. The DOVS is built including all the information in the robot velocity space, and it may be represented as seen in Figure 2, inside a red square. Linear velocities v are in the Y-axis, angular velocities ω in the X-axis, DOV are in black, the green rhombus is the dynamic window centered around the robot current velocity, the green line represents the velocities (ω, v) that lead to the goal following a circular trajectory ($radius = v/\omega$), and the big black triangle the differential-drive kinematic restriction (the robot may not go at maximum linear and angular velocity at the same time). In this way, all the information about the dynamism of the environment and the own robot needed for the robot motion planning is modeled.

C. Contribution

The works presented in the state of the art have some limitations when they are to be applied in real environments. Some of them use only the raw sensor measurements as the input or use some processed information, like obstacle position and velocities. The main problem with those approaches is the impossibility to generate appropriate real-world training data. They are trained in non-realistic simulators, and in different real world scenarios they would not do what to do. Furthermore, only few approaches do not use holonomic robots, and even less consider robot kinodynamic restrictions.

The contribution of this work is a deep reinforcement learning motion planner that:

- Uses a very complete and descriptive information of the environment as the input, as it is the DOVS. The information from the obstacles is extracted with an obstacle tracker in the same simulation and the real world from the laser sensor measurements, and used to build the DOVS (with safe and unsafe velocities). Once the agent learns how to interpret the DOVS, it will be able to navigate in any kind of real-world scenario with the same trained weights (overcoming the unsolvable the problem of generating real-world data).
- Takes into account dynamic and static obstacles (people, robots, walls...).
- Is trained in a realistic simulator, considering occlusions and obstacle velocities estimation errors.
- Publishes differential-drive motion commands that take into account kinodynamic restrictions of the robot.
- Is able to brake considering the deceleration constraint when it is reaching the goal (others do not, as they they do not consider kinodynamic constraints).
- Receives all the information from sensors.

III. Approach

A. State and action spaces

The state should describe the environment. In our approach, the DOVS is used as the main part of the state to model the needed information of the obstacles in a fixed way. Using raw velocity information of obstacles would require training with obstacles with every possible shape, velocity or radius a robot could face, which is impossible; so using the DOVS is a big advantage. The information of safe and unsafe velocities are extracted from the DOVS as a 20x20 grid, assigning value of 1 to free velocities and -1 to obstacle velocities (DOV), to keep the relationships between the velocities in the velocity space. Other few variables are also added to the state, to describe the current robot situation and the information of the closest obstacle in case there is an imminent collision. The whole state is represented in Figure 2.

The action space chosen is a discrete action set, relative to the current robot velocity and using the rhombus dynamic window (DW), to respect real robot kinodynamics. There are up to 8 available actions defined. 5 may be always chosen: The 4 corners of the DW (using maximum accelerations) and keeping the current velocity. The other three are only available if the velocities that lead to the goal are in DW: The two intersections of the line that leads to the goal with the dynamic window (maximum and minimum velocities for the next control period) and heading the goal with the current linear velocity. The whole set is represented in Figure 3. It is chosen as they are the actions that are more likely to be taken by natural trajectories, suitable for any differential drive robot. It is a discrete space instead of continuous to simplify the problem to improve training.

B. DQN-DOVS

The deep reinforcement learning algorithm chosen for the implementation is a Deep Q-Network with extensions,

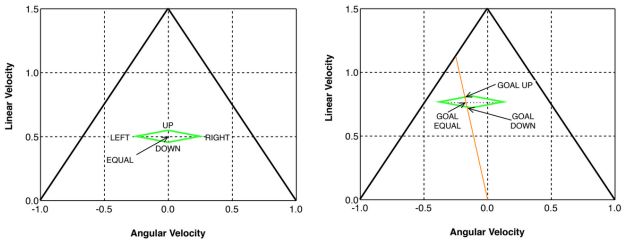


Fig. 3. Action set in the DQN-DOVS algorithm.

similar to the Rainbow DQN [8], as it offers results comparable to all the state-of-the-art in most of the problems with discrete action spaces. A decaying ϵ-greedy strategy is used to balance exploration and exploitation.

The extensions used from the original DQN approach are Huber loss [18] (more robust to outliers), Double Q-learning [19] (avoiding the overestimation of the Q-values), Prioritized Experience Replay, [20] (sampling more often transitions the agent may learn more from), Dueling DQN [21] (using the advantage function to compute the Q-values) and N-step bootstrapping [22] (bootstrapping the reward of several steps in the target value computation). Invalid actions (actions that lead to the goal when they are ineligible) are also taken into account in both the policy (what action to choose) and the target error computation to update the network weights. To do it, the invalid actions of each transition are stored in the replay buffer.

The Q-network is divided in three different parts sequentially connected: A feature network, a linear network and a dueling network. The feature network process separately the parts of the state. The DOVS image is fed into a convolutional network to extract the relationships among safe and unsafe velocities that are close to each other in the velocity space. The other 8 state variables are processed with a fully connected layer to exploit the relationships among them and increase their importance in the decision process. The outputs of both streams are concatenated into a linear network, which combines them. Finally, the dueling network computes the final Q-values of the actions by computing the state value and the advantage values in two streams. The structure of the network is shown in Figure 4.

C. Reward function

The goal of the agent is reaching and stopping in the goal while avoiding collisions in the shortest time possible. To achieve this behavior, the reward functions proposed in [23] and [24] are used as inspiration. It is defined with a simple equation that discriminates between terminal and non-terminal states:

$$r^t = \begin{cases} r_{goal}, & d^t_{goal} < 0.15 \text{ and } v^t < 0.2 \\ r_{collision}, & \text{collision detected} \\ -r_{dist}\Delta d^t_{goal} + r^t_{safedist}, & \text{otherwise} \end{cases} \quad (1)$$

The robot receives a reward of r_{goal}, set to 15, when it reaches and stops in the goal, using thresholds for the

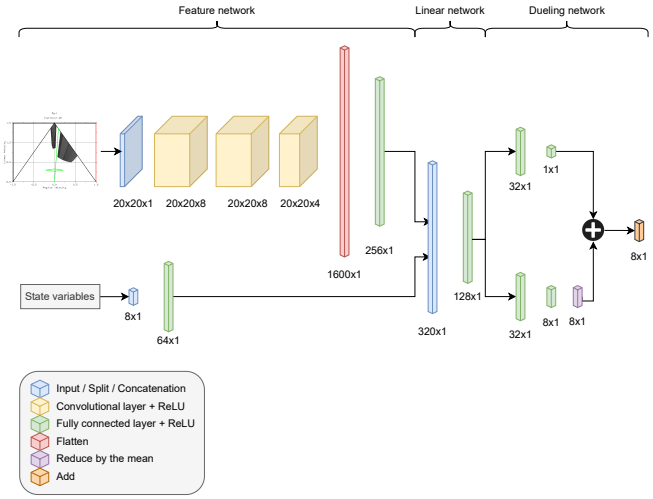


Fig. 4. The structure of the network of the DQN-DOVS.

distance of the robot to the goal, d^t_{goal}, and for the linear velocity, v^t; and a negative reward of -15 when it collides ($r_{collision}$). Reward shaping is used to accelerate training in non-terminal states by encouraging the agent to get closer to the goal (Δd^t_{goal} is the increment in the distance to the goal in consecutive time steps, r_{dist} is set to 2.5), and by penalizing the agent if it is too close to an obstacle (d^t_{obs} is the distance of the robot to the closest obstacle):

$$r^t_{safedist} = \begin{cases} -0.1|0.2 - d^t_{obs}|, & d^t_{obs} < 0.2 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

D. Training the network

The policy is trained in a modified version of the Stage simulator, which allows simulating robots similarly to the real world. An extended version of the work proposed by [25] is used to detect static and dynamic obstacles and estimate their position, radius, heading angle, angular and linear velocity in real time, from 2-D LIDAR measurements.Training in this kind of conditions makes the robot face occlusions or estimation errors already in training.

The training approach proposed in [24] has been used as influence to design the curriculum learning strategy applied for training. In our work, the policy has been trained in a 8x8 m scenario delimited with walls; with random positions for the robot, its goal and the other obstacles (and random obstacle velocities). The stages used are shown in Table I, trying to make the agent progressively learn how to reach the goal, avoid static obstacles and avoid dynamic obstacles; to fine-tune everything in the last stage. The distance to the goal and number of obstacles is also progressively increased inside some stages; and the ϵ value is decayed or not, depending on whether the task to learn is new.

IV. Results

A. Training details

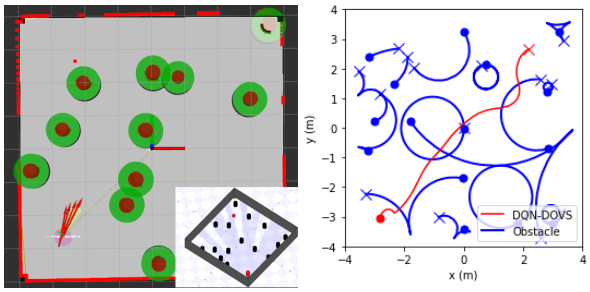
The agent was trained in the way presented in Section III-D. If the simulation took more than 500 time steps, the episode was finished (the robot gets stuck). The time step is set to 0.2 s. The dynamic obstacles had a predefined random linear and angular velocity each and may not avoid the agent. The network converges in about 10 hours in a computer with a Ryzen 7 5800x processor, a NVIDIA GeForce RTX 3060 graphics card and 64 GB of RAM. The key parameters used are a decaying learning rate of 0.0003 to 0.0001 with an Adam optimizer [26], discount factor of 0.97, n-step of 5 and a period of 100 to update the target network.

Episodes	Distance	ϵ	Obstacles	Type
1000	1 to any	Decay	0	–
1000	Any	Decay	0 to 12	Static
1000	Any	0.05	0 to 12	Static
1000	Any	Decay	0 to 12	Dynamic
1000	Any	0.05	0 to 12	Dynamic
2500	Any	0.05	Random	Both

TABLE I
Curriculum learning stages

B. Simulation evaluation

To evaluate the model, different open source implementations of other methods have been used. We compare our DQN-DOVS method with S-DOVS [1], the LSTM-RL (LSTM-RL-D) implementation of [15], the SG-D3QN [27], and the implementations of ORCA, SARL, CADRL and LSTM-RL (LSTM-RL-H) offered in [13] with the weights or the training scripts they provide. DQN-DOVS and S-DOVS consider every kind of kinodynamic restrictions, LSTM-RL-D and SG-D3QN only consider differential drive restrictions (but, for example, use infinite acceleration) and the rest do not consider any restriction and are holonomic. They are tested with the restrictions they consider.



(a) RVIZ and Simulator (b) Trajectories

Fig. 5. Scenario of a DQN-DOVS red agent and 15 dynamic obstacles. The final position of the robot and obstacles are marked with a *X*. Obstacle poses and velocities relative to the robot are computed from the simulated on board LIDAR sensor.

The methods have been tested in 200 episodes of the same kind of scenarios and obstacles described in Section IV-A, varying the number of agents, from 1 to 15 with 85% of the obstacles being dynamic (the same set of scenarios for every method). Although the scenarios may seem similar visually, what the robot perceives is completely different, depending

on the obstacles density and velocities. The results obtained are shown in Table II, with the success rate and the consumed time rate with respect to the DQN-DOVS.

The results show that the new DQN-DOVS algorithm outperforms the rest of the state-of-the-art algorithms; even though the DQN-DOVS agent may not go through the walls (some agents do), it takes into account acceleration constraints and needs to brake to stop in the goal. Moreover, it performs better even being differential drive (while LSTM-RL-H version performs better than LSTM-RL-D with the only difference of being holonomic).

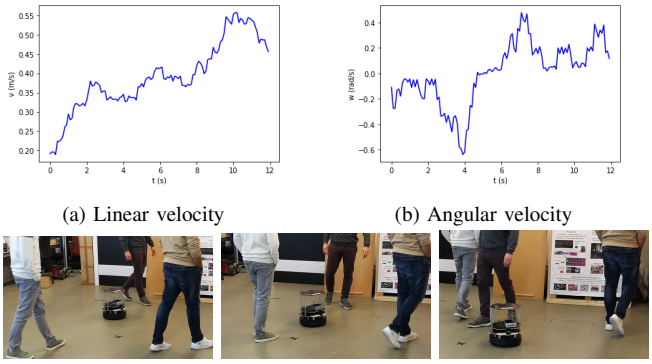
One of the differences between the new DQN-DOVS and the other algorithms is they assume perfect information knowledge. The results show that our method is the one that deals the best with those limitations, by moving in a way that both reduces occlusions and estimation errors and is less risky. The results also show that the premise on which this work is based is correct. The models that use raw or barely processed information as the input of the reinforcement learning algorithm do not perform successfully when the environment changes, while using the pre-processed information as DOVS generalizes better to any scenario. The DQN-DOVS improves strategies based methods as S-DOVS by using deep reinforcement learning, and other learning methods by using the DOVS.

DQN-DOVS is faster than the other algorithm that consider kinodynamic restrictions (S-DOVS). Times of LSTM-RL-D are comparable, even though LSTM-RL-D assumes no acceleration constraints (and success rates are not close). The SG-D3QN algorithm perform very risky maneuvers and very short trajectories, because it assumes the obstacles are going to behave exactly like the ones seen in training (with no occlusions or estimation errors too), and that is why its success rates are so low and it is so fast, as well as its lack of acceleration constraints that makes it have maximum velocity at any time. The time with respect to the other methods is not comparable because they do not have constraints and are holonomic. An example of behavior of the DQN-DOVS agent in a random scenario may be seen in Figure 5.

C. Real robot experiments

The whole system was integrated in a Turtlebot 2 platform with a NUC with Intel Core i5-6260U CPU and 8 GB of RAM. The sensor used is a 180º Hokuyo 2D-LIDAR. The approach taken from the beginning of the work was applying the simulation similarly as in the real world, using ROS. Thus, the same network weights and nodes used in simulation were used in the ground robot, adapting the obstacle tracker node to detect people and other objects and using AMCL for localization.

The experiments performed were setting the robot in a 8x8 m scenario where it had to navigate through different goals dynamically sent. Several people were wandering, acting as dynamic obstacles, and they were told not to look at the robot so that the avoidance was completely performed by it. In Figure 6, the robot velocity profiles during a time period in a experiment are shown, where the robot clearly respects the differential-drive kinodynamic restrictions, as it speeds up or decelerates respecting the acceleration constraints. In that specific situation, three moving obstacles surround the robot. The robot takes into account the obstacles trajectories and accelerates, turns to the right (second 3) and to the left (seconds 6, 11), and accelerates again, avoiding collisions successfully. The experiments performed show that the robot tries to predict obstacles trajectories in advance, and choose natural trajectories with maximum velocities to reach the goal, instead of avoiding obstacles clearly stopping and turning as other planners.



(a) Linear velocity (b) Angular velocity

(c) Pictures of the interval.

Fig. 6. Velocity measurements of the real robot in a short interval of time.

V. Conclusion

This work presents a motion planner and navigation algorithm for dynamic environments, that uses a differential drive robot, and the DOVS to model the environment as an input of a deep reinforcement learning technique to select the motion commands that lead the agent to reach the goal whilst avoiding collisions in the lower time possible. In addition, it proposes a training framework that uses curriculum learning in realistic scenarios to help the network to converge. The work is tested in random scenarios in a simulator where all the information is captured using a 2-D LIDAR sensor as in real life, outperforming existing methods that should have advantage, as they do not consider kinodynamic constraints of the robot and some of them use holonomic robots. The algorithm is also tested in a ground robot walking through pedestrians. Future work will include extending the model to collaborative collision avoidance with multi-robot navigation, trying more complex algorithms, and adapting the model for 3-D navigation and UAV (the model and the approach is extensible for 3-D spaces).

Acknowladgement

This work was partially supported by the Spanish projects MCIN/AEI/PID2019-105390RB-I00, and Aragon Government_FSE-T45_20R.

References

[1] M.-T. Lorente, E. Owen, and L. Montano, "Model-based robocentric planning and navigation for dynamic environments," *The International Journal of Robotics Research*, vol. 37, no. 8, pp. 867–889, 2018.

Obs	DQN-DOVS	S-DOVS	LSTM-RL-D	SG-D3QN	CADRL	ORCA	SARL	LSTM-RL-H
1	**0.94**	0.94(1.13)	**0.94**(0.85)	0.90(0.44)	0.93	0.93	0.94	0.94
2	**0.96**	0.91(1.13)	0.92(0.72)	0.81(0.43)	0.92	0.90	0.91	0.92
3	**0.89**	0.85(1.04)	0.84(0.84)	0.77(0.41)	0.82	0.87	0.84	0.83
4	0.80	**0.81**(1.03)	0.78(0.86)	0.70(0.48)	0.79	0.76	0.79	0.78
5	**0.82**	0.73(1.07)	0.76(0.91)	0.55(0.46)	0.78	0.76	0.76	0.75
6	**0.80**	0.71(1.06)	0.63(0.84)	0.57(0.47)	0.68	0.65	0.65	0.65
7	**0.74**	0.71(1.14)	0.63(0.92)	0.52(0.43)	0.72	0.70	0.70	0.73
8	**0.70**	0.55(1.03)	0.57(1.06)	0.44(0.41)	0.54	0.60	0.63	0.60
9	**0.72**	0.60(1.17)	0.53(1.18)	0.30(0.51)	0.57	0.57	0.52	0.58
10	**0.66**	0.58(1.13)	0.53(1.00)	0.30(0.44)	0.52	0.55	0.56	0.52
11	**0.70**	0.48(1.06)	0.54(1.04)	0.29(0.48)	0.51	0.49	0.55	0.54
12	**0.69**	0.61(1.02)	0.48(1.05)	0.27(0.42)	0.42	0.49	0.46	0.47
13	**0.55**	0.53(1.04)	0.34(1.19)	0.23(0.54)	0.48	0.50	0.46	0.40
14	**0.58**	0.52(1.18)	0.39(1.12)	0.21(0.38)	0.46	0.43	0.39	0.45
15	0.53	**0.56**(1.00)	0.27(1.37)	0.27(0.42)	0.45	0.45	0.41	0.47

TABLE II

SUCCESS RATE OF EVERY METHOD FOR DIFFERENT OBSTACLES. IN BRACKETS, TIME RATE WITH RESPECT TO THE DQN-DOVS ($\frac{\text{time of the method}}{\text{time of DQN-DOVS}}$).

[2] C. Qixin, H. Yanwen, and Z. Jingliang, "An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3331–3336.

[3] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[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] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[6] A. K. Mackay, L. Riazuelo, and L. Montano, "Rl-dovs: Reinforcement learning for autonomous robot navigation in dynamic environments," *Sensors*, vol. 22, no. 10, p. 3847, 2022.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[9] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," in *International Conference on Learning Representations*, 2018.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[11] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, pp. 1–50, 2020.

[12] L. Kästner, T. Buiyan, L. Jiao, T. A. Le, X. Zhao, Z. Shen, and J. Lambrecht, "Arena-rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6456–6463.

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

[14] 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.

[15] 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.

[16] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6057–6063.

[17] 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.

[18] P. J. Huber *et al.*, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[19] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *Proc. of ICLR*, 2015.

[21] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

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

[23] P. Long, T. Fan, 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.

[24] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 345–11 352.

[25] M. Przybyła, "Detection and tracking of 2d geometric obstacles from lrf data," in *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*. IEEE, 2017, pp. 135–141.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] Z. Zhou, P. Zhu, Z. Zeng, J. Xiao, H. Lu, and Z. Zhou, "Robot navigation in a crowd by integrating deep reinforcement learning and online planning," *Applied Intelligence*, pp. 1–17, 2022.