Robot Navigation With Reinforcement Learned Path Generation and Fine-Tuned Motion Control

Longyuan Zhang, Ziyue Hou, Ji Wang, Ziang Liu, and Wei Li, Member, IEEE

Abstract—In this letter, we propose a novel reinforcement learning (RL) based path generation (RL-PG) approach for mobile robot navigation without a prior exploration of an unknown environment. Multiple predictive path points are dynamically generated by a deep Markov model optimized using an RL approach for the robot to track. To ensure safety when tracking the predictive points, the robot’s motion is fine-tuned by a motion fine-tuning module. Such an approach, using a deep Markov model with RL algorithm for planning, focuses on the relationship between adjacent path points. We analyze the benefits of our proposed approach and show it is more effective and has higher success rates than the RL-based approach DWA-RL (Patel et al. 2021) and a traditional navigation approach APF (Chen et al. 2021). We deploy our model on both simulation and physical platforms and demonstrate our model performs robot navigation effectively and safely.

Index Terms—Motion and path planning, RL-based path generation, collision avoidance.

I. INTRODUCTION

THE path planning task is one of the most common tasks for mobile robots or autonomous vehicles. Especially in an unknown environment without mapping in advance, how a robot manages collision avoidance, trajectory smoothing, and avoiding sub-optimal solutions present challenges for path planning.

While traditional map-based approaches such as A* and RRT algorithms can adapt to unseen environments, they require a pre-exploration phase, which some other approaches [3], [4] do not require such phase. When faced with complex environmental situations or high dimensional input spaces, these approaches still suffer from slow computational speed. Though probabilistic approaches like PRM and RRT reduce the computational complexity of true classical approaches, they are vulnerable to environmental changes and PRM requires pre-computation, which is not a limitation of the proposed approach in the letter. Another defect is that the traditional path planning algorithms are prone to be affected by noise-filled raw sensor data, which will lead to a more difficult deployment of traditional path planning approaches on physical robots.

To solve the shortcomings of traditional algorithms in path planning problems, many learning-based approaches (e.g., [1], [5], [6]) have been proposed. Some imitation learning-based methods (e.g., [7]) have achieved fast inference speed in drone, autonomous driving, and mobile robot navigation tasks. However, it requires a large amount of training data. The expert trajectories in datasets are not guaranteed to be the most optimal. More importantly, the student network may not learn the corresponding trajectories with unseen observation spaces, which may lead to unexpected behavior. Some other approaches utilize the RL-based End-to-End approaches (e.g., [1], [5], [8], [9]) to learn from partial observation space to directly output driving commands. While these RL approaches can use safety layers and can generalize well, they produce low-quality trajectories with jitter, making the approach presented in this letter a needed advance to provide smoothness and better collision avoidance guarantees.

Here we present a RL-based predictive path generation approach with fine-tuned motion control to drive the mobile robots in a variety of complex environments without any prior exploration while having only access to onboard sensors and computation. Different from other RL-based End-to-End approaches to output driving actions, our deep RL-based method generates path points. This formulation could firstly fully utilize advantages of learning-based methods such as fast inference speed and robustness to learn a mapping from raw sensor data to various types of outputs. Another great benefit is that our approach manages to decouple trajectory generation and motion control since our RL-based approach’s action space and credit assignment are both based on planned trajectory, which means our path generation method is more robust to various environments. Once a model is trained, substituting its controller in a modular way could fulfill navigation task requirements.

Our actor policy designed for generating path points is based on a deep Markov model. When generating paths, we fully consider the sequential relationship of adjacent path points. That is, at each time step, each predictive path point is obtained based solely on the previous path point and partially observable space, and we treat this point set as a planning trajectory. During the path generating process, the robot’s position, posture, and sensor information are dynamically changed. Due to robot kinematic limitations, the robot’s motion may not be consistent with the generated path, resulting in collision. Thus the motion is, at each time step, each predictive path point is obtained based solely on the previous path point and partially observable space, and we treat this point set as a planning trajectory. During the path generating process, the robot’s position, posture, and sensor information are dynamically changed. Due to robot kinematic limitations, the robot’s motion may not be consistent with the generated path, resulting in collision. Thus the motion
Fig. 1. An overview of the framework. The obstacle data is collected by the lidar on Turtlebot3. The goal point is set up in advance. $Q_{i-1}$ and $Q_i$ represent the two adjacent path points. The network is a deep Markov model based policy network. During the robot movement, the path generation module dynamically generates multiple local paths based on varying sensor information and positions. The motion control module fine-tunes the paths and then sends execution commands to the robot.

The fine-tuning module is proposed to fix the problem and improve safety. The overall framework is shown in Fig. 1.

The main contributions of our work include:

- A deep RL-based path generation with fine-tuned motion control is proposed for robot’s navigation in an unknown environment without prior exploration. The prediction of path points enables navigation and collision avoidance, while motion control is only used to ensure safety of the robot in case of emergency.
- A novel deep Markov model under deep reinforcement learning framework to dynamically and iteratively plan path points. This predictive reinforcement learning based ‘action’ space later in simulation and experiments proves to be more effective when the robot explores and navigates in unknown environments.

II. RELATED WORK

To represent paths, the traditional approaches (e.g., [10]) use mostly points and line segments. Splitting the path into points and line segments is convenient for path expression and calculation. Traditional methods of path planning (e.g., [3], [11]), typically generate a predicted segment of trajectory based on its state and convert it into executable instructions. As the number of predicted nodes increases, the computation time and resource consumption of the whole algorithm increase dramatically.

A number of path planning methods combined with learning algorithms require prior exploration [12] of environments. For example, the long-range path planning method (PRM-RL) [8], [9] uses a traditional method PRM for path planning of a globally known map, and then uses reinforcement learning method to generate robot’s actions for movement. RL-RRT [13], similar to PRM-RL, uses RRT algorithm to plan the global path and uses RL method to control robot’s motion for dynamic obstacle avoidance. These approaches require a pre-exploration phase for navigation, and the role of RL algorithms is to avoid colliding with obstacles.

Some grid-map-based learning methods (e.g. [6], [14], [15], [16]) search for the higher scoring grids in the grid map to form a path. These discrete approaches are difficult to fulfill the kinematic constraints of physical robots. Some other similar studies focus on using RL based method to generate the global path. In the known map, studies (e.g., [17], [18], [19]) present approaches based on RL that train networks with global information to generate all path points from the start point to the goal point once for the robots to follow. Instead, our approach focuses on utilizing reinforcement learning methods to make local path planning using obstacles information scanned by robot’s lidar without a map built in advance, and use fine-tuned motion control for robot movement.

Some other methods that do not require a prior exploration of the map, such as a local planner trained by RL (e.g. [1], [5], [20], [21]), only plan the robot’s motion instructions. Point-Goal navigation, proposed in DDPPO [22], is used to infer the robot’s forward or rotation actions based on the images and GPS information using PPO method by giving robot the relative position of the target point. Here, the global map is of low importance and mainly assists the robot to locate the goal position. For trajectory generation based on learned approaches, the Performer-MPC work [23] combines cost function learning with model predictive control to roll out paths, fine-tuned by a safety layer. The Adaptive Planner Parameter Learning (APPL) lines of work (e.g., [24], [25], [26]) combines various forms of learning from feedback to learn parameters for classical motion planning for improved performance.

III. PROBLEM FORMULATION

We here define how the robot’s path is expressed and transformed in the coordinate frames used in our approach. We use
both an absolute world coordinate frame and a robot-relative local polar coordinate in our definition.

We use a set of points to represent a path. In the world coordinate system, the position and orientation of the robot is defined as \( Q_w(X,Y,\theta) \), where \( X, Y \) represent the coordinates of the robot in the world coordinate system and \( \theta \) represents the orientation of the robot. When converting \( Q_w \) into robot’s local polar coordinate \( Q_0 \), its local polar coordinates can be expressed as \( Q_0(0,0) \). We define the robot’s facing direction as the direction of the x-axis in the robot’s local polar coordinate system. The orientation \( \theta_0 \) in polar coordinate at \( Q_0 \) is 0.

Then we define a generated path in the local coordinate as a set
\[
L = \{ Q_i(\rho_i, \alpha_i) | i \in \{1, 2, \ldots, n\} \}. \tag{1}
\]
Here \( \rho_i \) represents the displacement of \( Q_i \) with respect to the previous point \( Q_{i-1} \), \( \alpha_i \) represents the angular deflection of \( Q_i \) with respect to the previous point \( Q_{i-1} \), shown in Fig. 2.

In another word, \( Q_i(\rho_i, \alpha_i) \) represents the predicted distance the robot needs to travel and the predicted angle the robot needs to rotate based on the \( (i-1) \)th path point \( Q_{i-1}(\rho_{i-1}, \alpha_{i-1}) \) and the orientation \( \theta_{i-1} \), shown in Fig. 2. Here the orientation \( \theta_i \) is accumulated by all \( \alpha_i \).
\[
\theta_i = \sum_{j=1}^{i} \alpha_j. \tag{2}
\]

We calculate the world coordinates of point \( i \) using \( \rho_i, \theta_i \) and its previous predicted point state \( Q_w(i-1, y_{i-1}, \theta_{w_{i-1}}) \). Here \( x_{i-1} \) and \( y_{i-1} \) are the world coordinates of previous points \( Q_w(i-1) \), and \( \theta_{w_{i-1}} \) is accumulated orientation of all the previous points in the world coordinates. Specially, for the first point \( Q_w(1) \), \( x_0, y_0 \) and \( \theta_{w_0} \) represent robot’s initial state.

When \( i \geq 1 \), for the \( i \)th point, we have
\[
x_i = x_{i-1} + \rho_i \cos(\theta_{w_i}), \quad y_i = y_{i-1} + \rho_i \sin(\theta_{w_i}). \tag{3}
\]

So in the world coordinate we have
\[
L_w = \{ Q_w(i, y_i, \theta_{w_i}) | i \in \{1, 2, \ldots, n\} \}. \tag{4}
\]
We use the robot’s local coordinate system to generate the robot’s path point set \( Q \) corresponding to itself and then transform the point set in the world coordinate for the controller to track.

For the goal point, we define it as \( G(\rho_g, \theta_g) \) in the robot’s local polar coordinate. For the obstacle data, we use 180-dimensional obstacle data in 3 consecutive frames scanned by lidar.

IV. APPROACH

A. Policy Representation

1) Observation Space: For each path point \( Q_i \), the observation \( o_i \) consists of three parts: the latest 3 frames of obstacle data \( o'_{gw} \), the goal point \( o^g_{gw} \) and the previous point \( Q_{i-1} \). Obstacle data is obtained from a rotating lidar sensor which returns distance to obstacles. Lidar is sampled every 1° from \(-90° \) to \(90° \) with \(0° \) as the robot’s forward heading. Three complete rotations of the lidar can then be combined into a history of three obstacle data readings as the obstacle data matrix \( o'_g \).
\[
o'_g = \begin{bmatrix} D_1 \n D_2 \n D_3 \end{bmatrix}, \quad n D_i = \{d_i | i \in [-90, 90] \cap \mathbb{Z}, n = 1, 2, 3\}. \tag{5}
\]
Here \( i \) represents the \( i \)th angle divided equally by 180-degree from \(-90° \) to \(90° \), \( n \) represents the \( n \)th frame of the obstacle data, and \( d_i \) represents the distance between the lidar and the obstacle scanned on the angle \( i \) in the \( n \)th frame. As the robot is moving, \( n D_i \) are scanned at different positions and postures.

The goal point information needs to be transformed. We convert the goal point from the world coordinate to the local polar coordinate according to the goal point position \( G_w(x_{gw}, y_{gw}) \) calibrated in the world coordinate system. We define the current state information of the robot mentioned in Section III as \( Q_w(X,Y,\theta) \). Here we have
\[
x_g = (x_{gw} - X) \cos \theta + (y_{gw} - Y) \sin \theta. \tag{6}
\]
\[
y_g = - (x_{gw} - X) \sin \theta + (y_{gw} - Y) \cos \theta. \tag{7}
\]
\[
(\rho_g, \theta_g) = \left( \sqrt{x_g^2 + y_g^2}, \arctan(y_g/x_g) \right). \tag{8}
\]
Thus we have the goal information in local polar coordinate \( o'^g_{gw}(\rho_g, \theta_g) \).

Therefore, the observation space \( o_i \) consists of three parts.
\[
o_i = (o'_{gw}, o^g_{gw}, Q_{i-1}). \tag{9}
\]
2) Action Space: In each cycle when generating a path, the path point \( Q_i \) is one of the actions according to the data in the
observation space and the network. Here the action space $\mathbf{a}^t$ can be represented by $Q_i$. During the process of generating the path, all the actions are combined to be a complete path. The architecture of the deep Markov model based policy network is shown in Fig. 3. According to the definition of the path in (1), we have

$$\mathbf{a}^t = \{Q_i| i \in (1, 2, \ldots, n)\}. \quad (10)$$

3) Reward Structure: We design the reward by judging the three elements of the path point. We call these three parts as:

- $r_c$, if the predicted path collides with obstacles
- $r_n$, if the robot approaches the goal point
- $r_s$, smoothness judgment

The first is whether the generated path collides with the obstacle scanned by the lidar. We need to judge whether the path generated by the robot conflicts with the obstacles, as shown in Fig. 2. For each frame of data swept by the lidar, we compute the scanned point and compute the distance $\Delta$ between the scanned point and the line segment connected by the path point. If $\Delta$ is less than the robot’s radius, it is judged as a collision.

$$r_c = \begin{cases} -15, & \text{break if } \Delta_i < \text{radius }, i \in [-90, 90] \cap \mathbb{N} \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

If a collision occurs, we set $r_c = -15$, and terminate the training of the current process, which determines that the task failed.

For whether the path is approaching the goal, we compare the distance from the path points to the goal point and the distance from the robot to the goal point. For each point $i$, $s_i$ represents the distance between the i-th path point and the goal point, and $d$ represents the distance between the robot and the goal point. If $s_i$ is smaller, the path point is closer to the goal, and the feedback is positive. Otherwise, it is negative. Thus

$$r_n = \sum_{i=1}^{n} \left( \frac{d - s_i}{i} \right). \quad (12)$$

For the smoothness judgment, if the second parameter $\alpha_d$ of the generated path points $Q_i(\rho_i, \alpha_d)$ is large, it means that the angle the robot needs to turn is large. Then, we can limit the size of the angle of each path point $\alpha_d$ to solve the problem of path smoothness. Thus, for all points in the path,

$$r_s = -\lambda \sum_{i=1}^{n} \alpha_d^2, \lambda = 0.0005. \quad (13)$$

Combining $r_c, r_n$ and $r_s$, we will obtain the total reward

$$r = r_c + r_n + r_s. \quad (14)$$

4) Actor-Critic Network: Our policy network is trained and inferred in an iterative fashion. Given the input $o^t$ mentioned in (9) and output $a^t$ mentioned in (10), our policy could iteratively compute the mapping from observation space $o^t$ to action space $a^t$.

Note that not all observation space is observable. After the first iteration, path points are no longer obtainable. Only at the first iteration, our $Q_i$ as part of observation space is obtained directly as the point that represents the current robot location. For the rest of the iteration steps, we mark those unobservable positional points as ‘virtual’ position states that are generated by the previous iteration step, meaning the current single trajectory point depends on and only on partially observable environment space (in our space, lidar scan) and assume positional status at current iteration step which is exactly the output from the previous step. If $\pi$ represents the policy, we can generate all $Q$s from

$$Q_i \sim \pi(Q_i|o^t_i, o^g_i, Q_{i-1}) \sim \pi(Q_i|o^t_i, o^g_i, \pi(Q_{i-1}|o^t_i, o^g_i, Q_{i-2}))) \sim \pi(Q_i|o^t_i, o^g_i, \pi(\ldots \pi(Q_1|o^t_0, o^g_0, Q_0)))). \quad (15)$$

Our network shares the same weights and parameters at each iteration step. It is comprised of two convolutional layers to convolve $3 \times 180$ dimensional data $o^t_i$ to $1 \times 256$, then it is concatenated with other two inputs $o^g_i$ and $Q_{i-1}$ to form $260 (256+2+2)$ dimensional data. Then it comes through a fully connected layer that is added with leaky rectified linear units (ReLUs). The output of the network are variables: mean and standard deviation of Gaussian Distribution, as required by the PPO method to generate continuous and more diverse action space, followed by a sampling method thus to return final positional point as our action space. After all iteration steps are finished, our final action space as path points are obtained by concatenating all single-step action space, shown in (10).

B. Training

For training, we refer to multi-process training approaches (e.g., [5], [6], [27], [28]) to improve the training efficiency. Multi-process Proximal Policy Optimization (PPO) [5] enables
ψ
ψ
i

\[ F(i) = \alpha \sum_{j=i-5}^{i+5} (d_j - \text{max\_range})^2 + \beta \psi^2 \]  
\[ i = \arg \min_{i \in [-85, 85]} F(i) \]

We record \( i \) when \( F(i) \) is minimal and regard it as the execution direction. Here \( i \) is in the range \([-85, 85]\) to ensure the value \( d \) exists in (16). Here we set the first point of the predicted path as the main moving direction. For the obstacle data scanned by lidar, if the obstacles are within 0.5 m of the robot, the motion is fine-tuned to a direction away from the obstacles.

V. RESULTS AND DISCUSSION

We demonstrate that our proposed approach has a better structure than other approaches through comparative experiments. Meanwhile, we aim to find the best path structure by ablation experiments.

In our comparison experiments, we focus on the comparison with the DWA-RL approach proposed in [1] and traditional approach Artificial potential field (APF) approach [2], [29]. To compare the corresponding effects, the obstacle information acquisition of APF and DWA-RL approaches are adjusted accordingly in the experiment environments.

We test our model to demonstrate that our proposed approach can be applied to other different complex environments. In the ablation experiments, we test the performance of different number of path points \( N \) and the distance \( \rho \) between adjacent path points.

A. Evaluation Metrics

The following metrics are used to compare the performance of our approach with other approaches and to make further analysis in ablation studies.

- **Average Trajectory Length** - Average length of trajectory the robot travels from start point to goal point in the same test map.
- **Time Cost** - The average time cost from start point to goal point in the same test map.
- **Success Rate** - The rate of robot successfully travelling in an episode without any collision and finally reaching the goal. If a collision happens, the current test is immediately terminated and marked as failed.

B. Comparison Experiments

1) Simulation Experiments: We test our approach with DWA-RL approach [1] and APF approach [2], [29] on the same testing maps shown in Fig. 6 based on the simulator Stage. In these testing maps, the corner formed by walls are tricky, and thus robots have difficulty finding the balance between moving toward a target point and avoiding obstacles, which may lead...
Fig. 6. Testing maps. We designed different maps to test our trained model and compared RL-PG approach with DWA-RL and APF. The size of Obstacle Map is 33 m × 10 m, the size of Maze Map is 15 m × 15 m, and the size of Zigzag Map is 30 m × 30 m.

Fig. 7. We compare our model with DWA-RL approach and APF approach in testing maps shown in Fig. 6. The three different colors of trajectories drawn by the robot correspond to three different algorithms.

Table I shows the detailed comparison results of the robot navigation tasks with different approaches. In Maze Map and Obstacle Map, DWA-RL and APF approaches fail to find the goal due to the deceptive obstacles. For trajectory length, our RL-PG approach is better in a deceptive map shown in Fig. 7, while DWA-RL or APF approaches perform better in a simple environment. The max speed is all set to 0.3 m/s and we can find that the RL-PG costs less time in most testing maps. As for the success rate, our RL-PG is higher, while DWA-RL and APF fail in some of the testing environments.

2) Real Scenario Experiments: We designed physical experiments using Turtlebot3 robot with RPlidar_A2 on a 3 m × 3 m area and the data is communicated through ROS. We reduced the map size compared to the scene of the simulated environment. Some experimental screenshots and their corresponding trajectories are shown in Fig. 8. We found the pitfalls of the DWA-RL approach in real-world settings. In the real experimental scene, the DWA-RL approach trajectories are longer and the time costs are more than our approach as shown in Fig. 8.

3) Expansion Experiments: We tested the trained model using BARN [30] benchmark in Gazebo simulation. We chose 100 different maps and tested each map for 5 times to count the Success Rate. In the testing maps, RL-PG performs well and the Success Rate is 86.3%, demonstrating that RL-PG approach can be applied to other environments. Two of the scenarios and their corresponding trajectories are shown in Fig. 9(a)(b).

We also tested our model in our lab environments, from a room to another. The robot found the correct path to reach target points, without collision with walls or other obstacles. The environment and the trajectory are shown in Fig. 9(c).

C. Ablation Studies

In the simulation environment, we did ablation studies to observe the effects by different distances $\rho$ between adjacent path points and different number of the path points $N$. We trained each model for 1000 episodes to compare the results. Here we recorded the test data shown in Table II. From an overall perspective, when $N = 10$ and $\rho = 0.30$, the average trajectory length is minimum, and when $N = 5$ and $\rho = 0.05$, the average time cost is minimum. When $N$ is large, the success rate is lower.

We find that the performances of different distances between adjacent path points are close. The difference is that with smaller distances between adjacent path points, the network infers a higher probability of a predicted path that does not collide with an obstacle. The reason is that the larger the $\rho$, the easier the
generated path is to conflict with the scanned obstacles. For $N$, when $N$ is small, the path points are more dispersed and require greater angular deflection in path tracking, which makes robot consume more time to turn a larger angle. However when $N$ is large, the time cost also increases and it takes more time to find the feasible, uncollided path.

VI. CONCLUSION

In this letter, we propose a novel RL-based robot path generation (RL-PG) approach with fine-tuned motion control. A deep Markov model suitable for path generation is trained using RL approach, which generates local paths in complex environments. The corrective effect of the controller is also able to correct the path generated by the deep Markov model to avoid collisions and improve the safety performance of the robot system.

With the experiments of comparing our approach with DWA-RL approach and traditional APF approach, we demonstrate that the RL-based path generation (RL-PG) with fine-tuned motion control is more effective and more safe. What is more, our approach is able to find feasible paths in complex maze-like environments. In future work, we will try to apply the approach of path generation in the motion planning for multiple robots.

REFERENCES

[1] 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 Proc. IEEE Int. Conf. Robot. Automat., 2021, pp. 6057–6063.
[2] Y. Chen, G. Bai, Y. Zhan, X. Hu, and J. Liu, “Path planning and obstacle avoiding of the USV based on improved ACO-APF hybrid algorithm with adaptive early-warning,” IEEE Access, vol. 9, pp. 40728–40742, 2021.
[3] M. Missura and M. Bennewitz, “Predictive collision avoidance for the dynamic window approach,” in Proc. IEEE Int. Conf. Robot. Automat., 2019, pp. 8620–8626.
[4] T. P. Nascimento, C. E. Dórea, and L. M. G. Gonçalves, “Nonholonomic mobile robots’ trajectory tracking model predictive control: A survey,” Robotics, vol. 36, no. 5, pp. 676–696, 2018.
[5] T. Fan, P. Long, W. Liu, and J. Pan, “Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios,” Int. J. Robot. Res., vol. 39, no. 7, pp. 856–892, 2020.
[6] G. Sartoretti et al., “PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning,” IEEE Robot. Automat. Lett., vol. 4, no. 3, pp. 2378–2385, Jul. 2019.

[7] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, “Vision-based autonomous car racing using deep imitative reinforcement learning,” IEEE Robot. Automat. Lett., vol. 6, no. 4, pp. 7262–7269, Oct. 2021.

[8] A. Francis et al., “Long-range indoor navigation with PRM-RL,” IEEE Trans. Robot., vol. 36, no. 4, pp. 1115–1134, Aug. 2020.

[9] A. Faust et al., “PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 5113–5120.

[10] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, “EGO-planner: An ESDF-free gradient-based local planner for quadrotors,” IEEE Robot. Automat. Lett., vol. 6, no. 2, pp. 478–485, Apr. 2021.

[11] M. Ryll, J. Ware, J. Carter, and N. Roy, “Efficient trajectory planning for high speed flight in unknown environments,” in Proc. IEEE Int. Conf. Robot. Automat., 2019, pp. 732–738.

[12] P. Anderson et al., “On evaluation of embodied navigation agents,” 2018, arXiv:1807.06757.

[13] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, “RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies,” IEEE Robot. Automat. Lett., vol. 4, no. 4, pp. 4298–4305, Oct. 2019.

[14] M. Fei, H. An, B. Liu, and C. Wang, “An improved dyna-Q algorithm for mobile robot path planning in unknown dynamic environment,” IEEE Trans. Syst., Man, Cybern. Syst., vol. 52, no. 7, pp. 4415–4425, Jul. 2022.

[15] X. Zhang, X. Kang, K. Wei, J. Li, and K. Ma, “Reinforcement learning combined with heuristic search for solving discrete space planning problems,” in Proc. IEEE 33rd Chin. Control Decis. Conf., 2021, pp. 2142–2147.

[16] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, “UAV path planning for wireless data harvesting: A deep reinforcement learning approach,” in Proc. IEEE GLOBECOM Glob. Commun. Conf., 2020, pp. 1–6.

[17] D. Perille, A. Truong, X. Xiao, and P. Stone, “Benchmarking metric ground navigation,” in Proc. IEEE Int. Symp. Saf. Secur. Rescue Robot., 2020, pp. 116–121.

[18] P. Gao, Z. Liu, Z. Wu, and D. Wang, “A global path planning algorithm for robots using reinforcement learning,” in Proc. IEEE Int. Conf. Robot. Biomimetics, 2019, pp. 1693–1698.

[19] J. Yu, Y. Su, and Y. Liao, “The path planning of mobile robot by neural networks and hierarchical reinforcement learning,” Front. Neurorobot., vol. 14, 2020, Art. no. 63.

[20] G. Bellegarda and K. Byl, “An online training method for augmenting MPC with deep reinforcement learning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2020, pp. 5453–5459.

[21] Z. Ullah, Z. Xu, L. Zhang, L. Zhang, and W. Ullah, “RL and ANN based modular path planning controller for resource-constrained robots in the indoor complex dynamic environment,” IEEE Access, vol. 6, pp. 74557–74568, 2018.

[22] E. Wijmans et al., “DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames,” 2019, arXiv:1911.00357.

[23] X. Xiao et al., “Learning model predictive controllers with real-time attention for real-world navigation,” 2022, arXiv:2209.10780.

[24] X. Xiao et al., “APPL: Adaptive planner parameter learning,” Robot. Auton. Syst., vol. 154, 2022, Art. no. 104132.

[25] Z. Wang, X. Xiao, B. Liu, G. Warnell, and P. Stone, “APPLI: Adaptive planner parameter learning from interventions,” in Proc. IEEE Int. Conf. Robot. Automat., 2021, pp. 6079–6085.

[26] Z. Wang, X. Xiao, G. Warnell, and P. Stone, “APPLE: Adaptive planner parameter learning from evaluative feedback,” IEEE Robot. Automat. Lett., vol. 6, no. 4, pp. 7744–7749, Oct. 2021.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

[28] 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 Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 6252–6259.

[29] Y. Huang et al., “A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach,” IEEE Trans. Ind. Electron., vol. 67, no. 2, pp. 1376–1386, Feb. 2020.