Design and Experimental Validation of Deep Reinforcement Learning-Based Fast Trajectory Planning and Control for Mobile Robot in Unknown Environment

DOI:
10.1109/TNNLS.2022.3209154

Design and Experimental Validation of Deep Reinforcement Learning-based Fast Trajectory Planning and Control for Mobile Robot in Unknown Environment

Runqi Chai, *Member, IEEE,* Hanlin Niu, *Member, IEEE,* Joaquin Carrasco, *Member, IEEE,* Farshad Arvin, *Senior Member, IEEE,* Hujun Yin, *Senior Member, IEEE,* Barry Lennox, *Senior Member, IEEE*

***Abstract*—This paper is concerned with the problem of planning optimal maneuver trajectories and guiding the mobile robot toward target positions in uncertain environments for exploration purposes. A hierarchical deep learning-based control framework is proposed which consists of an upper-level motion planning layer and a lower-level waypoint tracking layer. In the motion planning phase, a recurrent deep neural network (RDNN)-based algorithm is adopted to predict the optimal maneuver profiles for the mobile robot. This approach is built upon a recently-proposed idea of using deep neural networks (DNNs) to approximate the optimal motion trajectories, which has been validated that a fast approximation performance can be achieved. To further enhance the network prediction performance, a recurrent network model capable of fully exploiting the inherent relationship between pre-optimized system state and control pairs is advocated. In the lower-level, a deep reinforcement learning (DRL)-based collision-free control algorithm is established to achieve the waypoint tracking task in an uncertain environment (e.g., existence of unexpected obstacles). Since this approach allows the control policy to directly learn from human demonstration data, the time required by the training process can be significantly reduced. Moreover, a noisy prioritized experience replay (PER) algorithm is proposed to improve the exploring rate of control policy. The effectiveness of applying the proposed deep learning-based control is validated by executing a number of simulation and experimental case studies. The simulation result shows that the proposed DRL method outperforms vanilla PER algorithm in terms of training speed. Experimental videos are also uploaded and the corresponding results confirm that the proposed strategy is able to fulfill the autonomous exploration mission with improved motion planning performance, enhanced collision avoidance ability, and less training time.**

***Index Terms*—Mobile robot, Optimal motion planning, Motion control, Recurrent neural network, Deep reinforcement learning, Unexpected obstacles, Noisy prioritized experience replay.**

I. Introduction

This work was supported by EPSRC project No.EP/S03286X/1, EPSRC RAIN project No. EP/R026084/1 and EPSRC RNE project No. EP/P01366X/1. *(Corresponding author: Hanlin Niu)*

R. Chai is with the School of Automation, Beijing Institute of Technology, 100081, Beijing, China. (e-mail: r.chai@bit.edu.cn).

H. Niu, J. Carrasco, H. Yin, and B. Lennox are with the Department of Electrical and Electronic Engineering, The University of Manchester, M13 9PL Manchester, U.K. (e-mail: hanlin.niu@manchester.ac.uk; joaquin.carrasco@manchester.ac.uk; hujun.yin@manchester.ac.uk; barry.lennox@manchester.ac.uk).

F. Arvin is with the Department of Computer Science, Durham University, U.K. (e-mail: farshad.arvin@durham.ac.uk).

THE development of unmanned ground vehicles (UGVs) or mobile robots has received considerable attention over the last two decades due to its potential benefits in terms of sensing the unknown environment, alleviating urban traffic congestion, reducing harmful emissions, and enhancing road safety. Generally speaking, four key modules are involved in a robotic system: environmental perception [1], motion planning [2], motion control [3], [4], and vehicle-to-vehicle communication [5]. Among them, the planning and control modules are mainly responsible for navigating the route and making maneuver decisions. Consequently, it is usually recognised as a key indicator to reflect the intelligence level of a robotic system. However, there are challenges when it comes to planning and steering the motion of mobile robots in an optimal and timely way, considering the complexity of the environment and various physical limitations of the robot. These challenges, together with other mission-dependent requirements, have stimulated both scholars and engineers to explore promising solutions to this kind of problem.

A. Related Works

The mission investigated in this research work focuses on applying the mobile robot to explore unknown target regions. Numerous research works have been carried out in developing advanced path planning and control algorithms for dealing with this problem [6]–[9]. For instance, in [7] the authors proposed a two-level path planning scheme incorporating the standard A-star method and an approximate policy iteration algorithm to generate smooth trajectory for mobile robots. Similarly, Kontoudis and Vamvoudakis [8] proposed a hybrid path planning approach combining rapidly-exploring random tree and Q-learning techniques to achieve collision-free navigation. In [9], a trajectory planner without the time-consuming process of building the Euclidean Signed Distance Field is proposed. EGO-planner explicitly constructs an objective function that keeps the trajectory away from obstacles while ESDF-based methods have to calculate collision information with a heavy computation burden. Although EGO-planner reduces a significant amount of computing time comparing with ESDF-based planners [10] [11], it still needs to parameterize trajectory by a uniform B-spline curve first and then optimize it using smoothness penalty, collision penalty, and

feasibility penalty terms. Moreover, it needs an additional time re-allocation procedure to ensure dynamical feasibility and generate a local optimal collision-free path for unknown obstacle. In this paper, we propose an end-to-end DRL method to deal with unknown obstacle and our DRL method uses lidar sensor data as input of neural network and generates collision avoidance command directly and it does not need manually designed optimization approach like EGO-planner. The DRL method is also integrated with our proposed global optimal path planning method for known obstacles.

These works belong to the class of sample-and-search methods where a finite set of mesh grids is firstly applied to discretize the searching space. Then a satisfactory connection between the initial pose and the target pose is selected.

Another potentially effective strategy is to construct an optimal control model consisting of the vehicle dynamics, and other vehicle-related constraints. Subsequently, well-developed optimization-based control algorithms such as swarm intelligence-oriented algorithms [12], dynamic programming-based methods [13]–[15], and nonlinear model predictive control (NMPC)-based approaches are utilized to plan and steer the motion trajectories [16], [17]. Specifically, by taking into account public traffic restrictions, a model predictive navigation and control framework was established in [16] to plan and steer the UGV trajectory. Besides, the authors of [17] proposed an MPC-scheme, which is capable of anticipating the motion of the front vehicle such that the autonomous car following control task can be successfully fulfilled. Compared to sample-and-search methods, the application of optimization-based planning/control approaches has several advantages: (i) Different mission/environment-related requirement (e.g., constraints) can be explicitly considered; (ii) Since the motion trajectory is directly planned, the path velocity decomposition becomes non-necessary. Hence, these types of algorithms have received an increasing attention in the context of autonomous exploration.

B. Motivations

Different from the aforementioned research works, this paper aims to design, test and validate a fast yet optimal motion planning and control architecture for mobile robot operating under an unstructured and partly unknown environment (e.g., target region exploration, rapid rescue, autonomous parking, etc.). For the considered mission scenarios, the main challenges may result from three aspects:

- Convergence failures can frequently be detected in the optimization process with the consideration of mechanical limitations, nonlinear path/dynamic constraints and non-convex collision avoidance constraints.
- The (re)optimization process requires a large amount of onboard computing power.
- The actual operation environment of the mobile robot is uncertain, meaning that there may exist obstacles suddenly appear within the sensing range of the robot. This will certainly create a hazard and lead to significant safety risks.

These three issues become more prominent when it comes to real-time applications.

C. Main Contributions

The main contribution of the present work lies in dealing with the three challenges stated previously and developing a hierarchical deep learning-based control framework for the autonomous exploration problem. To be more specific, for the first two issues, we make an attempt to address them by exploring the idea of planning the optimal motion trajectory via deep neural network (DNN)-based direct recalling. This type of strategy aims to train DNNs on a pre-optimized state-action dataset. Following that, the trained networks will be applied as an online motion planner to iteratively predict optimal maneuver actions. Such a direct recalling strategy can acquire near-optimal motion planning performance and better real-time applicability in comparison to other optimization-based techniques such as NMPC and dynamic programming [18]–[22]. However, most of the them assumed the network inputs are independent to each other and applied a fully-connected network structure to learn the optimal mapping relationship. This indicates that the inherent relations between the optimized state and control action pairs may not be sufficiently exploited. Therefore, building upon our previous studies [23], we enrich the power of the DNN-based motion approximator by using a gated recurrent unit recurrent deep neural network (GRURDNN) model.

As for the third issue, we make an attempt to address it by taking advantages of deep reinforcement learning (DRL)-based techniques [24], [25]. More precisely, DRL has been used on autonomous vehicles for collision avoidance using lidar [26] or camera data [27]. However, previous research works rely heavily on the training data and the exploration usually starts on random action to gather experience data for replaying. This random action is likely to result in low quality training data and slow convergence speed. In this research, we proposed a noisy prioritized experience replay (PER) algorithm to improve the training speed of the standard PER algorithm. The proposed noisy PER algorithm is integrated into the Deep Deterministic Policy Gradient (DDPG) algorithm. Compared to the vanilla PER algorithm implemented in our previous work [28], this modification can introduce more exploring actions in early training stage and perform higher training speed. The effectiveness of this approach will be validated in the experimental section of this paper.

D. Organization

The remainder of this article is arranged as follows. In Section II, a constrained trajectory optimization model is firstly formulated so as to describe the mobile robot autonomous exploration mission profile. Following that, Section III constructs the designed GRURDNN optimal trajectory approximator, while the DRL-based collision avoidance approach is presented in Section IV such that unexpected obstacles can be avoided during the maneuver process. Simulation verification analysis, along with experimental validation results on a real-world test platform, are demonstrated in Section V to confirm the effectiveness and appreciate the advantages of the proposed design. Some concluding remarks are summarised and presented in Section VI.

II. Problem Formulation

For the considered autonomous exploration task, we are interested in transferring a mobile robot from its current position to an unknown target position for an exploration purpose. During the movement, it is required that the collision risk between the mobile robot and obstacles (e.g., expected objects on the environmental map as well as the unexpected ones) can be removed, while simultaneously minimizing the mission complete time. To formulate this problem, a constrained nonlinear trajectory optimization model is established in this section.

A. Dynamic Model and System Constraints of a Mobile Robot

The kinematic equations of motion for the mobile robot are characterized as [13]:

$$\begin{cases} \dot{p}_x(t) = v(t)\cos(\theta(t)) \\ \dot{p}_y(t) = v(t)\sin(\theta(t)) \\ \dot{\theta}(t) = \omega(t) \\ \dot{v}(t) = a_v(t) \\ \dot{\omega}(t) = a_\omega(t) \end{cases} \tag{1}$$

Eq. (1) can be abbreviated as $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$, in which the state and control variables of the mobile robot are defined as $\boldsymbol{x}(t) = [p_x(t), p_y(t), \theta(t), v(t), \omega(t)]^T \in \mathbb{R}^{5\times 1}$ and $\boldsymbol{u}(t) = [a_v(t), a_\omega(t)]^T \in \mathbb{R}^{2\times 1}$, respectively. (p_x, p_y) stands for the position of the mobile robot, and θ is the orientation angle. The linear and angular velocities are denoted as v and ω, whereas a_v and a_ω are the corresponding accelerations.

Based on Eq. (1), one can derive the nonholonomic constraint $\dot{p}_x(t)\sin(\theta) - \dot{p}_y(t)\cos(\theta) = 0$, indicating that the mobile robot cannot perform lateral movement [29]. However, forward maneuvers can be performed as one has $\dot{p}_x(t)\cos(\theta) + \dot{p}_y(t)\sin(\theta) = v(t)$.

For the exploration problem, the boundary conditions for the mobile robot states at $t_0 = 0$ and t_f can be specified as:

$$\begin{aligned} \boldsymbol{x}(t_0) &= \boldsymbol{x}_0, \\ \boldsymbol{x}(t_f) &= \boldsymbol{x}_f, \end{aligned} \tag{2}$$

where $\boldsymbol{x}_0, \boldsymbol{x}_f \in \mathbb{R}^{5\times 1}$ are scenario-dependent. Due to the existence of mechanical/environmental limitations, system state and control variables must vary in their allowable ranges during the entire movement. This can be written as:

$$\begin{cases} \boldsymbol{x}(t) \in \mathbb{X}, \quad \boldsymbol{u}(t) \in \mathbb{U}, & \text{(3a)} \\ \mathbb{X} = \{\boldsymbol{x}(t) \in \mathbb{R}^{5\times 1} : |\boldsymbol{x}(t)| \le \bar{\boldsymbol{x}}_{\max}\}, & \text{(3b)} \\ \mathbb{U} = \{\boldsymbol{u}(t) \in \mathbb{R}^{2\times 1} : |\boldsymbol{u}(t)| \le \bar{\boldsymbol{u}}_{\max}\}, & \text{(3c)} \end{cases}$$

where $\bar{\boldsymbol{x}}_{\max} = [p_{x_{\max}}, p_{y_{\max}}, \theta_{\max}, v_{\max}, \omega_{\max}]^T$ and $\bar{\boldsymbol{u}}_{\max} = [a_v^{\max}, a_\omega^{\max}]^T$ are, respectively, the maximum allowable values for the state and control variables.

Remark 1. Note that in Eq. (1), the last two dynamic equations are introduced such that the continuity and smoothness of $\theta(t)$ and $v(t)$ can be improved. Meanwhile, it is relatively simple to consider the linear and angular acceleration constraints by imposing explicit constraints on the system control variables (see e.g., (3c)). Note that the continuity and smoothness of the velocity vector can be further improved by considering higher time derivatives of position after acceleration, namely "jerk" or "snap", as used respectively in [30], [31].

B. Collision Avoidance for Expected Obstacles

In real-world scenarios, certain constraints should be imposed to avoid the collision risk of the mobile robot and obstacles existing on the exploration map. Suppose that the position information of the on-map obstacles is available for the motion planner [21], [22], [32]. We can define the region occupied by the mobile robot and obstacles as $\mathbb{A}(\boldsymbol{x}) \in \mathbb{R}^2$ and $\mathbb{O}^{(m)} \in \mathbb{R}^2$ with $m = 1, 2, ...N_o$ being the obstacle index. Then, a collision-free trajectory should satisfy the following condition:

$$\mathbb{O}^{(m)} \cap \mathbb{A}(\boldsymbol{x}) = \emptyset, \quad \forall m = 1, ..., N_o, \tag{4}$$

where the full dimensional mobile robot $\mathbb{A}(\boldsymbol{x})$ and on-map obstacles $\mathbb{O}^{(m)}$ are modeled as convex polytopes:

$$\begin{cases} \mathbb{A}(\boldsymbol{x}) = A(\boldsymbol{x})\mathbb{W} + B(\boldsymbol{x}), & \text{(5a)} \\ \mathbb{O}^{(m)} = \{\boldsymbol{z} \in \mathbb{R}^2 | P^{(m)}\boldsymbol{z} \le q^{(m)}\}. & \text{(5b)} \end{cases}$$

with $\mathbb{W}$ being an initial compact set regulated by (G, g) (e.g., $\mathbb{W} = \{z \in \mathbb{R}^2 | Gz \le g\}$) and similarly, $(P^{(m)}, q^{(m)})$ regulates the shape of the m-th known obstacle. Besides, $A(x)$ and $B(x)$ are the rotation matrix and the translation vector that determine the orientation and position of the mobile robot. By introducing a safety margin parameter $d_{\min}$, the collision between the robot and m-th obstacle can be avoided if the following inequality holds true:

$$\text{dist}(\mathbb{A}(\boldsymbol{x}), \mathbb{O}^{(m)}) > d_{\min} \ge 0. \tag{6}$$

with $\text{dist}(\mathbb{A}(\boldsymbol{x}), \mathbb{O}^{(m)})$ defined by:

$$\text{dist}(\mathbb{A}(\boldsymbol{x}), \mathbb{O}^{(m)}) \colon = \min_{r^{(m)}} \{\|r^{(m)}\| \colon (\mathbb{A}(\boldsymbol{x}) + r^{(m)}) \cap \mathbb{O}^{(m)} = \emptyset\}. \tag{7}$$

Eq. (6) can be directly adhered to the mobile robot trajectory optimization model as path constraint during the entire mission $t \in [0, t_f]$. However, it may result in computational issues such as numerical difficulties and convergence failure for gradient-based optimization algorithms. Thanks to the proposition established in [21] and validated in [22], condition (6) can be transformed to a smoother form such that the computational complexity of the optimization process can be alleviated significantly. To be more specific, imposing Eq. (6) is equivalent to:

$$\begin{aligned} \text{Find} \quad & \boldsymbol{\lambda}^{(m)} \ge 0, \boldsymbol{\mu}^{(m)} \ge 0 \\ \text{subject to} \quad & \forall t \in [0, t_f] \\ & (P^{(m)}B(\boldsymbol{x}) - q^{(m)})^T\boldsymbol{\lambda}^{(m)} - g^T\boldsymbol{\mu}^{(m)} > d_{\min} \\ & A^T(\boldsymbol{x}){P^{(m)}}^T\lambda^{(m)} + G^T\boldsymbol{\mu}^{(m)} = 0 \\ & \|{P^{(m)}}^T\boldsymbol{\lambda}^{(m)}\| \le 1 \end{aligned} \tag{8}$$

in which $(\lambda^{(m)}, \mu^{(m)})$ denotes the dual variables.

C. Trajectory Optimization Formulation

According to the mobile robot dynamic model, mechanical constraints, boundary conditions and collision avoidance constraints introduced previously, we are now in a place to establish a finite-horizon constrained trajectory optimization formulation to describe the overall exploration mission. That is, a collision-free optimal exploration trajectory for the mobile

robot can be obtained by addressing the following optimization model:

$$\begin{aligned}\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\lambda}^{(m)},\boldsymbol{\mu}^{(m)}} \quad & J=\Phi(\boldsymbol{x}(t_f),t_f)+\int_{t_0}^{t_f}L(\boldsymbol{x}(t),\boldsymbol{u}(t))dt\\ \text{subject to} \quad & \forall m\in\{1,...,N_o\},\forall t\in[0,t_f]\\ & \dot{\boldsymbol{x}}(t)=f(\boldsymbol{x}(t),\boldsymbol{u}(t))\\ & (P^{(m)}B(\boldsymbol{x})-q^{(m)})^T\boldsymbol{\lambda}^{(m)}-g^T\boldsymbol{\mu}^{(m)}>d_{\min}\\ & A^T(\boldsymbol{x}){P^{(m)}}^T\lambda^{(m)}+G^T\boldsymbol{\mu}^{(m)}=0\\ & \|{P^{(m)}}^T\boldsymbol{\lambda}^{(m)}\|\leq 1\\ & \boldsymbol{x}(t_0)=\boldsymbol{x}_0,\quad \boldsymbol{x}(t_f)=\boldsymbol{x}_f\\ & \boldsymbol{x}(t)\in\mathbb{X},\quad \boldsymbol{u}(t)\in\mathbb{U}\\ & \lambda^{(m)}\geq 0,\quad \mu^{(m)}\geq 0\end{aligned} \tag{9}$$

in which J stands for the objective function consisting of a terminal cost term $\Phi(\boldsymbol{x}(t_f),t_f)$ and a process cost term $L(\boldsymbol{x}(t),\boldsymbol{u}(t))$. Since we are aiming to fulfill the exploration mission in the shortest time, the objective function is set to $J=t_f$. Note that t_f is not given in advance and is included in the constraints of optimization model (9).

Remark 2. Certain advantages can be obtained if the optimization model (9) is applied to plan the exploration trajectory of mobile robots: (i) Since $\|{P^{(m)}}^T\boldsymbol{\lambda}^{(m)}\|\leq 1$ is a convex constraint, an improved solution time and convergence rate can be expected; (ii) This formulation considers convex obstacles and no further assumptions are applied to the geometric shape of obstacles; (iii) Additional mission-dependent conditions/requirements can be easily considered in (9) by modeling them as algebraic inequalities or equalities; (iv) Though we are aiming for time-optimal exploration motions, it is simple to modify this optimization model to consider other mission objectives.

III. GRURDNN-BASED OPTIMAL MOTION PLANNING

A GRURDNN-based real-time motion planner capable of approximating the optimal exploration trajectories is constructed in this section. This approach can be viewed as an extended version of the fully-connected DNN-based motion planner advocated in our previous work [23]. The main novelty of the proposed one is that it implements a recurrent network structure with gated recurrent unit mechanism to further exploit the inherent relationships between different network inputs (e.g., state variables of the mobile robot). To form the GRURDNN-based optimal motion planner, four main steps are required, which will be discussed in later subsections.

A. Forming the Exploration Trajectory Dataset

Prior to construct the network model, a trajectory dataset $\mathbb{E}$ containing N_e optimized vehicle state and control evolution histories is pre-generated for a particular mobile robot exploration scenario (e.g., $\|\mathbb{E}\|=N_e$). This can be achieved by utilizing the Markov chain Monte-Carlo (MCMC) sampling method. That is, the trajectory optimization model (9) is iteratively solved with $\boldsymbol{x}(t_0)=\boldsymbol{x}_0+\boldsymbol{\Delta}_x$. Here $\boldsymbol{\Delta}_x$ is a random noise value. Note that in real-world applications, it is reasonable to assume the initial configuration of the mobile robot is noise-perturbed due to the existence of localization errors.

Algorithm 1 Optimal parking maneuver dataset generation

procedure
  Initialize $\mathbb{E}=\emptyset$;
  Form a set of temporal node $\{t_k\}_{k=1}^{N_k}$;
  Apply MCMC to sample a set of noise value $\{\boldsymbol{\Delta}_x^i\}_{i=1}^{N_i}$;
  /*Main loop*/
  for $i:=1,2,...,$ **do**
    while $\|\mathbb{E}\|<N_e$ **do**
      (a) Specify

$$x_0=x_0+\boldsymbol{\Delta}_x^i$$

      (b) Use $\{t_k\}_{k=1}^{N_k}$ to construct the discrete-time optimization model of (9):

$$\begin{aligned}\min_{x_k,u_k,\lambda_k^{(m)},\mu_k^{(m)}} \quad & J=\sum_{k=0}^{N_k}L(x_k,u_k)+\Phi(x_{N_k},t_k)\\ \text{s.t.} \quad & \forall m\in\{1,...,N_o\},\quad \forall k\in\{0,...,N_k\}\\ & \boldsymbol{x}_{k+1}=f(\boldsymbol{x}_k,\boldsymbol{u}_k,\Delta t_k)\\ & (P^{(m)}B(\boldsymbol{x}_k)-q^{(m)})^T\boldsymbol{\lambda}_k^{(m)}-g^T\boldsymbol{\mu}_k^{(m)}>d_{\min}\\ & A^T(\boldsymbol{x}_k){P^{(m)}}^T\lambda_k^{(m)}+G^T\boldsymbol{\mu}_k^{(m)}=0\\ & \|{P^{(m)}}^T\boldsymbol{\lambda}_k^{(m)}\|\leq 1\\ & \boldsymbol{x}_0=\boldsymbol{x}_0+\boldsymbol{\Delta}_x^i\\ & \boldsymbol{x}_{N_k}=\boldsymbol{x}_f\\ & \boldsymbol{x}_k\in\mathbb{X},\quad \boldsymbol{u}_k\in\mathbb{U}\\ & \lambda_k^{(m)}\geq 0,\quad \mu_k^{(m)}\geq 0\end{aligned} \tag{10}$$

      (c) Perform Algorithm 2 to generate an initial feasible guess trajectory history $(\boldsymbol{x}^{(0)},\boldsymbol{u}^{(0)})$;
      (d) Address the static nonlinear programming problem (10) via numerical optimization algorithm [33] with $(\boldsymbol{x}^{(0)},\boldsymbol{u}^{(0)})$ as the initial guess;
      if optimization tolerance is satisfied **then**
        (e) Output the optimal time history and perform

$$\mathbb{E}=\mathbb{E}\cup\{(\boldsymbol{x}_k^*,\boldsymbol{u}_k^*)\}$$

      else
        (f) Discard the unconverged solution and perform

$$\mathbb{E}=\mathbb{E}\cup\emptyset$$

      end if
      (g) Update the index $i\leftarrow\!\!\shortmid i+1$;
    end while
  end for
  Divide $\mathbb{E}$ into subsets via Eq.(17);
  Output $\mathbb{E}$, $\mathbb{E}_{\text{tr}}$, $\mathbb{E}_{\text{te}}$ and $\mathbb{E}_{\text{v}}$.
end procedure

To present the process of forming $\mathbb{E}$ in detail, a pseudocode is established (see e.g., Algorithm 1). The final output dataset is divided into three subsets which will be used to train, test and validate the designed GRURDNN network. That is,

$$\mathbb{E}=\mathbb{E}_{\text{tr}}\cup\mathbb{E}_{\text{te}}\cup\mathbb{E}_{\text{v}} \tag{17}$$

Here, N_{tr}, N_{tr} and N_{tr} denote the training, testing and validation sets with $N_{\text{tr}}=\|\mathbb{E}_{\text{tr}}\|$, $N_{\text{te}}=\|\mathbb{E}_{\text{te}}\|$ and $N_{\text{v}}=\|\mathbb{E}_{\text{v}}\|$ being the size of $\mathbb{E}_{\text{tr}}$, $\mathbb{E}_{\text{te}}$, and $\mathbb{E}_{\text{v}}$, respectively. Note that in (10), the state and control variables at time instant t_k are denoted as $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$, respectively. The time step is defined as $\Delta t_k=t_{k+1}-t_k$, with $t_1=0$ and $t_{N_k}=t_f$.

Remark 3. Compared to the trajectory dataset generation strategy designed in [23], [35], an improved reference trajectory generation strategy is designed and embedded in Algorithm

Algorithm 2 Initial feasible trajectory generation

1: **procedure**
2: Create the exploration map based on $(p_{x_{\max}}, p_{y_{\max}})$ and expected obstacle information;
3: Grid the exploration map uniformly with specific resolution level;
4: Dilate the irregular obstacle to meet the safety margin $d_{\min}$ [34];
5: Apply the A-star algorithm to explore a shortest collision-free trajectory $\boldsymbol{X}_p^{(0)} = (p_x^{(0)}, p_y^{(0)})$ between the initial and target points;
6: **for** $k := 0, 1, ..., N_k$ **do**
7: (a) calculate other state profiles via

$$\begin{aligned} \theta^{(0)}(k) &= \sin^{-1}\left(\frac{p_x^{(0)}(k+1) - p_x^{(0)}(k)}{\|\boldsymbol{X}_p^{(0)}(k+1) - \boldsymbol{X}_p^{(0)}(k)\|_2}\right) \\ v^{(0)}(k) &= \frac{\|\boldsymbol{X}_p^{(0)}(k+1) - \boldsymbol{X}_p^{(0)}(k)\|_2}{\Delta t_k} \\ \omega^{(0)}(k) &= \frac{\|\theta^{(0)}(k+1) - \theta^{(0)}(k)\|_2}{\Delta t_k} \end{aligned} \tag{11}$$

8: (b) Modify the oriental angle via

$$\theta^{(0)}(k) = \begin{cases} \theta_{\max} & \text{if} \quad \theta^{(0)}(k) > \theta_{\max}, \\ -\theta_{\max} & \text{if} \quad -\theta^{(0)}(k) < -\theta_{\max}. \end{cases} \tag{12}$$

9: (c) Modify the linear velocity via

$$v^{(0)}(k) = \begin{cases} \bar{v}^{(0)}(k) & \text{if} \quad \frac{v^{(0)}(k+1) - v^{(0)}(k)}{\Delta t_k} > a_v^{\max}, \\ \underline{v}^{(0)}(k) & \text{if} \quad \frac{v^{(0)}(k+1) - v^{(0)}(k)}{\Delta t_k} < -a_v^{\max}, \\ v_{\max} & \text{if} \quad v^{(0)}(k) > \theta_{\max}, \\ -v_{\max} & \text{if} \quad -v^{(0)}(k) < -\theta_{\max}. \end{cases} \tag{13}$$

10: with

$$\begin{aligned} \bar{v}^{(0)}(k+1) &= v^{(0)}(k) + a_v^{\max}\Delta_{t_k} \\ \underline{v}(k+1)^{(0)} &= v^{(0)}(k) - a_v^{\max}\Delta_{t_k} \end{aligned} \tag{14}$$

11: (d) Modify the angular velocities via

$$\omega^{(0)}(k) = \begin{cases} \bar{\omega}^0(k) & \text{if} \quad \frac{\omega^{(0)}(k+1) - \omega^{(0)}(k)}{\Delta t_k} > a_\omega^{\max}, \\ \underline{\omega}^{(0)}(k) & \text{if} \quad \frac{\omega^{(0)}(k+1) - \omega^{(0)}(k)}{\Delta t_k} < -a_\omega^{\max}, \\ \omega_{\max} & \text{if} \quad \omega^{(0)}(k) > \omega_{\max}, \\ -\omega_{\max} & \text{if} \quad -\omega^{(0)}(k) < -\omega_{\max}. \end{cases} \tag{15}$$

12: with

$$\begin{aligned} \bar{\omega}^{(0)}(k+1) &= \omega^{(0)}(k) + a_\omega^{\max}\Delta_{t_k} \\ \underline{\omega}^{(0)}(k+1) &= \omega^{(0)}(k) - a_\omega^{\max}\Delta_{t_k} \end{aligned} \tag{16}$$

13: **end for**
14: Output the initial collision-free trajectory

$$\boldsymbol{x}^{(0)} = [p_x^{(0)}, p_y^{(0)}, \theta^{(0)}, v^{(0)}, \omega^{(0)}]^T \in \mathbb{R}^{5\times N_k+1}$$

15: **end procedure**

1 (see Algorithm 2 for a detailed illustration). This strategy adopts the A-star algorithm to find the initial reference trajectory. It guarantees that the gradient-based numerical optimizer can quickly start the searching process at a collision-free path. As a result, the success rate of optimal solution detection, along with the time required to form the optimized trajectory dataset, is likely to be further improved. Note that designs in [23], [35] are likely to take longer computational time, as a pre-optimization process has to be applied.

Remark 4. Performing Lines 6-14 is to generate the initial reference trajectory for $\theta^{(0)}$, $v^{(0)}$ and $\omega^{(0)}$ such that they can satisfy the dynamic inequality constraints (3b). This is achieved by re-shaping $\theta^{(0)}$, $v^{(0)}$ and $\omega^{(0)}$ according to (12), (13) and (15). This modification may make the trajectory fail to meet system dynamics. It is worth noting that the primary aim of this initial trajectory generation strategy is to quickly find a collision-free maneuver path to warmly start the gradient-based optimization process. It does not need to exactly satisfy the system dynamics at all time nodes.

Remark 5. Note that the exploration trajectory dataset is generated from known static obstacle locations and is therefore environment-specific. However, datasets generated in this way have the possibility to be applicable to other scenarios through methods such as transfer learning.

B. Structuring and Training the Network

RDNN motion planning networks with GRU mechanism are structured and trained on the pre-generated training dataset $\mathbb{E}_{\text{tr}}$ such that the mapping relations between the optimized states and control actions can be approximated. The motivation for the use of DNN with a recurrent structure relies on its capability to dig the dynamic temporal behaviour of the time series data, as it introduces memory block in neurons. Moreover, compared to the widely-used long-short term memory (LSTM) mechanism, the implementation of GRU can result in less number of network structural parameters, thereby easing the training process to some degree. As for the mobile robot autonomous exploration mission, two GRURDNNs, including a linear acceleration control network N_{a_v} and a angular acceleration control network N_{a_ω}, are built in order to approximate the time-optimal actions $\boldsymbol{u}_k^*$ with $\boldsymbol{x}_k$ as the network input. More precisely, we have

$$\begin{aligned} a_v^*(t_k) &:\approx N_{a_v}(\boldsymbol{x}_k), \\ a_\omega^*(t_k) &:\approx N_{a_\omega}(\boldsymbol{x}_k). \end{aligned} \tag{18}$$

For simplicity reasons, the above equation is abbreviated as $\boldsymbol{u}_k :\approx N_{\boldsymbol{u}}(\boldsymbol{x}_k)$ for the rest of the paper. As shown in Fig. 1, supposing the GRURDNN network has N_L layers and N_n neurons at each layer, then at time node t_k, the output of the g-th neuron in the j-th layer o_{j,g,t_k} can be written as:

$$\begin{cases} \boldsymbol{z}_{j,g,t_k} = \sigma_s(W_z \boldsymbol{x}_{j,g,t_k} + U_z \boldsymbol{o}_{j,g,t_{k-1}} + \boldsymbol{b}_z) \\ \boldsymbol{h}_{j,g,t_k} = \sigma_s(W_h \boldsymbol{x}_{j,g,t_k} + U_h \boldsymbol{o}_{j,g,t_{k-1}} + \boldsymbol{b}_h) \\ \hat{\boldsymbol{h}}_{j,g,t_k} = \sigma_h(W_{\hat{h}} \boldsymbol{x}_{j,g,t_k} + U_{\hat{h}}(\boldsymbol{o}_{j,g,t_{k-1}} \odot \boldsymbol{h}_{j,g,t_k}) + \boldsymbol{b}_c) \\ \boldsymbol{o}_{j,g,t_k} = \boldsymbol{z}_{j,g,t_k} \odot \hat{\boldsymbol{h}}_{j,g,t_k} + (1 - \boldsymbol{z}_{j,g,t_k}) \odot \boldsymbol{h}_{j,g,t_{k-1}} \end{cases} \tag{19}$$

where $j = \{1, ..., N_l\}$ and $g = \{1, ..., N_n\}$. $\odot$ stands for the Hadamard product operator. $(\boldsymbol{z}_{(\cdot)}, \boldsymbol{h}_{(\cdot)}, \hat{\boldsymbol{h}}_{(\cdot)}, \boldsymbol{o}_{(\cdot)})$ represents, respectively, the update gate, reset gate, activation, and output vectors. $(W_{(\cdot)}, \boldsymbol{b}_{(\cdot)})$ denotes the weight matrices and bias vectors. $\sigma_s(\cdot)$ and $\sigma_h(\cdot)$ are the activation functions, which can be written as:

$$\begin{aligned} \sigma_s(\boldsymbol{x}) &= \frac{e^x}{1+e^x}, \\ \sigma_h(\boldsymbol{x}) &= \frac{e^x - e^{-x}}{e^x + e^{-x}}. \end{aligned} \tag{20}$$

To train the network and update the weight matrices as well as bias vectors, a performance index should be defined. As suggested in [36], the mean squared error (MSE) measure is

applied to evaluate the performance of the network approximation ability:

$$L(W_{(\cdot)}, \boldsymbol{b}_{(\cdot)}) = \frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} \frac{1}{N_k} \sum_{j=1}^{N_k} \left[\boldsymbol{o}(t_{jT}) - \boldsymbol{x}(t_{jT})\right]^2 \quad (21)$$

in which $T = \frac{t_f}{N_k}$, whereas $\boldsymbol{o}(t_{jT})$ and $\boldsymbol{x}(t_{jT})$ are, respectively, the actual and target network outputs. To train the network, the stochastic gradient descent (SGD) algorithm, incorporating the adaptive learning rate technique [37], is used.

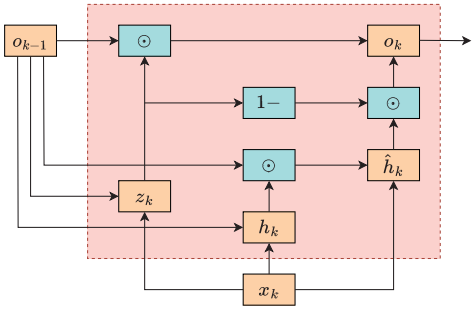


Fig. 1: The architecture of GRURDNN

C. Approximating the Optimized Exploration Trajectory

It should be noted that the construction of the optimized trajectory dataset and GRURDNN networks, along with the training process, can all be performed offline. Subsequently, the trained network can serve as the motion planner to approximate the optimal trajectory in real-time. General procedures of this online approximation process are detailed in the next pseudocode (see Algorithm 3).

Algorithm 3 Approximating time-optimal exploration trajectory

1: **procedure**
2: At each time point $t_k, k = 0, 1, ...$; **do**
3: (a) Measure the actual state of the mobile robot $\boldsymbol{x}_k$;
4: (b) Approximate the optimal motion command $\boldsymbol{u}_k$ via

$$\boldsymbol{u}_k :\approx N_{\boldsymbol{u}}(\boldsymbol{x}_k)$$

5: (c) Apply $N_{\boldsymbol{u}}(\boldsymbol{x}_k)$ to propagate the nonlinear dynamics:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, N_{\boldsymbol{u}}(\boldsymbol{x}_k), \Delta t_k)$$

6: (d) Update the time point $t_k \leftarrow\!\shortmid t_{k+1}$, and go back to (a).
7: **end procedure**

IV. Collision-Free DRL-based Online Control

In order to avoid unexpected collision risk when the mobile robot is following the optimal trajectory planned by the proposed GRURDNN-based algorithm, we proposed a collision-free DRL-based online control algorithm. The proposed algorithm is a DRL-based mapless collision avoidance algorithm that does not need accurate map information and also has the fast computing speed that can cope with the unexpected obstacles. The proposed training method enables the control policy to learn from human demonstration data instead of exploring random data. A simple yet efficient noisy weight method is also proposed to improve the training speed.

A. State and Action Space

To differ the upper motion planning level, the state space at time step t is denoted by vector s_t that consists of lidar data vector l_t, velocity data vector v_{t-1} at previous time step, and relative target position vector p_t. The lidar data is normalized to the range of $[0, 1]$. p_t is represented in the polar coordinate. The action space at time step t is denoted by v_t and consists of linear velocity v_{lt} and angular velocity v_{at}.

B. Reward Space

Mobile robot is controlled to follow the designed optimal trajectory while also taking into account the unexpected obstacles. Reward function is designed to navigate mobile robot to next waypoint whilst avoiding unexpected obstacles. Reward function is defined as

$$r_{\text{sum}} = r_{\text{gd}} + r_{\text{sc}} + r_{\text{v}} \quad (22)$$

where r_{sum} represents the sum reward, r_{gd} describes the goal distance reward, r_{sc} is the safety clearance reward, r_{v} represents velocity control reward. r_{gd} can be calculated by

$$r_{\text{gd}} = \begin{cases} r_{\text{a}} & \text{if } d_{\text{rg}} < d_{\text{rgmin}} \\ k_p \Delta d_p & \text{otherwise} \end{cases} \quad (23)$$

where mobile robot receives arrival reward r_a when the distance between robot and goal d_{rg} is less than threshold d_{rgmin}. Otherwise, r_{gd} is proportional to Δd_p that represents the distance robot moves towards the goal at the last time step. k_p is a user-configurable parameter. Safety clearance reward r_{sc} can be calculated by

$$r_{\text{sc}} = \begin{cases} -r_{\text{cp}} & \text{if } d_{\text{ro}} < d_{\text{romin}} \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

When the distance between robot and obstacle d_{ro} is less than threshold d_{romin}, negative reward/punishment $-r_{\text{cp}}$ is applied. The velocity control reward r_v, angular velocity reward r_{av} and linear velocity r_{lv} reward are given by

$$r_{\text{v}} = r_{\text{av}} + r_{\text{lv}} \quad (25)$$

$$r_{\text{av}} = \begin{cases} -r_{\text{ap}} & \text{if } |v_{\text{a}}| > v_{\text{amax}} \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

$$r_{\text{lv}} = \begin{cases} -r_{\text{lp}} & \text{if } v_{\text{l}} < v_{\text{lmin}} \\ 0 & \text{otherwise} \end{cases} \quad (27)$$

where v_{amax} denotes the angular velocity threshold and v_{lmin} represents the linear velocity threshold. As we want mobile robot to move to goal waypoint at high linear speed and low angular speed, if angular velocity v_a is bigger than v_{amax} or linear velocity v_l is less than v_{lmin}, robot will receive punishment value $-r_{\text{ap}}$ or $-r_{\text{lp}}$, respectively.

C. Actor/Critic Network Structure

The architectures of actor network and critic network are shown in Fig. 2 and Fig. 3, respectively. In Fig. 2, the input

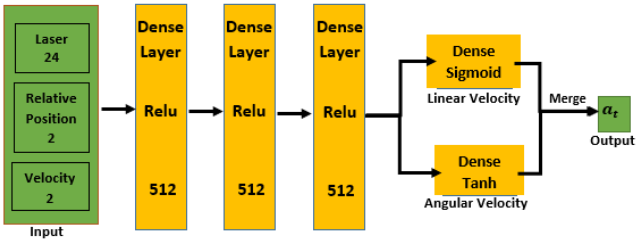


Fig. 2: Actor network architecture: the input layer concatenates laser scanner data, relative target position and velocity, followed by three dense layers. The number in each block means the corresponding dimensional number of each vector or layer.

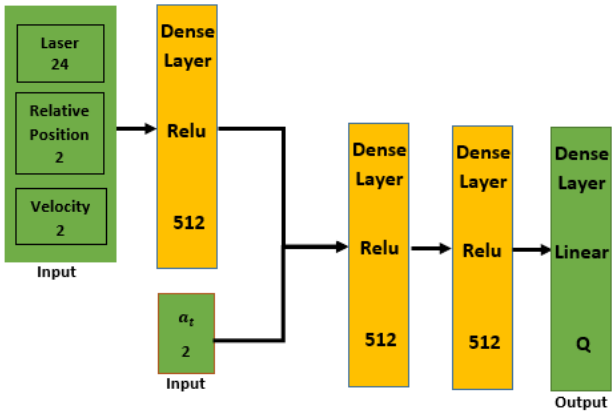


Fig. 3: Critic network architecture: the input layer concatenates laser scanner data, relative target position data, velocity data and action data, followed by three dense layers. The number in each block means the corresponding dimensional number of each vector or layer.

layer concatenates laser scanner data (24-dimensional vector), relative target position (2-dimensional vector) and velocity data (2-dimensional vector). The input layer is followed by three dense layers and each layer includes 512 nodes and rectified linear unit function (ReLU). The output layer is the velocity command, including linear velocity generated by a sigmoid function and angular velocity generated by a hyperbolic tangent function. The input layer and output layer of the actor network architecture are merged and used as the input layer of critic network, as shown in Fig. 3. Critic network also includes three dense layers and each dense layer includes 512 nodes and ReLU function. Q-value is finally generated by a linear activation function.

D. Human Teleoperation Data Collection

Instead of making robot to learn from scratch, this method allows robot to learn from human demonstration data in the beginning. The human demonstration data is received by recording the teleoperation data of one user. The user can control robot to navigate around obstacles and arrive at goal by using keyboard to change the speed and heading of turtlebot under simulated environment. The state data s_t at time step t consists of lidar data l_t, velocity data v_t, relative position data p_t. Velocity command v_t is sent by using keyboard. After one time step, next state s_{t+1} can be recorded. The reward value r is evaluated using (22). Then (s_t, v_t, s_{t+1}, r_t) is stored in experience dataset for later sampling.

E. Noisy Prioritized Experience Replay Algorithm

To improve the training speed of Prioritized Experience Replay (PER) algorithm, we proposed noisy prioritized experience replay algorithm to increase the exploration rate of control policy. We integrated noisy PER algorithm and DDPG to enable control policy to sample the human demonstration data more frequently in the beginning and sample dataset with higher priority more frequently afterwards. The noisy prioritized experience replay algorithm is given as below.

$$P_i = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{28}$$

where P_i represents the sampling probability of the ith transition, α is distribution factor and p_i stands for the priority of a set of transition data, formulated in (29).

$$p_i = \delta_i^2 + \lambda \left| \nabla_a Q(s_i, a_i | \theta^Q) \right|^2 + \epsilon + \epsilon_D \tag{29}$$

where the time difference (TD) error is denoted by δ, the second term $\lambda \left| \nabla_a Q(s_i, a_i | \theta^Q) \right|^2$ denotes the actor loss, λ is a contribution factor, and ϵ represents a small positive sampling probability for each transition that ensuring each transition data can be sampled once. ϵ_D represents the extra sampling probability of the demonstration data to improve the sampling frequency of demonstration data. Each transition is evaluated by:

$$\omega_i^{\text{critic}} = (\frac{1}{N} \cdot \frac{1}{P_i})^\beta \tag{30}$$

$$\omega_i^{\text{actor}} = (\frac{1}{N} \cdot \frac{1}{P_i})^\beta (1 + \mu) \tag{31}$$

in which ω_i^{critic} and ω_i^{actor} represents the sampling weight for updating the critic network and actor network, respectively, indicating the importance of each transition data, N stands for the batch size, and β is a constant value for adjusting the sampling weight. The sampling weight of the actor network has a noisy term μ which can be regarded as the probability density function of the standard normal distribution. Different with noisy parameter method [38], we added noisy term μ to the weight of actor network directly and this will not increase the computational complexity and yet ensure actor network to explore more actions.

F. Overall framework

The processes of the proposed GRURDNN-based motion planner and DRL-based online control algorithm are concluded in Fig. 4 to better illustrate the hierarchical deep learning-based control framework.

V. Simulation Verification and Experimental Validation

In this section, simulation verification and experimental validation results on different autonomous exploration test cases

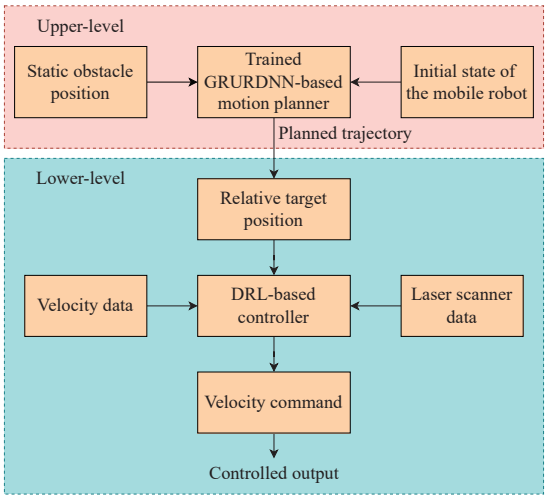


Fig. 4: The whole process of the proposed method

are displayed to evaluate the performance of the proposed hierarchical deep learning-based control framework.

A. Parameter/Scenario Definitions

Three simulation test cases are executed in this study. Table I demonstrates the physical parameters/constraints of the mobile robot.

TABLE I: Physical parameters/constraints of the mobile robot

Parameter/Notation	Values/ranges
Robot length, m	0.281
Robot width, m	0.306
p_x, m	$[-2, 2]$
p_y, m	$[-2, 2]$
θ, rad	$[-3.14, 3.14]$
v, m/s	$[0, 0.26]$
ω, rad/s	$[-0.576, 0.576]$
a_v, m/s^2	$[-0.1, 0.1]$
a_ω, rad/s^2	$[-0.576, 0.576]$
t_f, s	$[0, 100]$

In terms of scenario-related parameters such as the initial and terminal configurations of the mobile robot, they are assigned in Table II. In addition, $N_o = 8, 9, 12$ obstacles are randomly generated for scenarios 1-3, respectively.

TABLE II: Initial/Final configuration for different test cases

Scenario No.	Initial configuration		
	(p_x, p_y, θ)	v	ω
1	$(1.25, -1.5, 1.57)$	0	0
2	$(1.0, -1.3, 3.14)$	0	0
3	$(1.5, -1.25, 3.14)$	0	0
Scenario No.	Terminal configuration		
	(p_x, p_y, θ)	v	ω
1	$(-0.5, 0.75, 3.14)$	0	0
2	$(0.8, 1.2, 1.57)$	0	0
3	$(-0.3, 0.8, 0.79)$	0	0

Other GRURDNN-related parameters are specified as $N_k = 100$, and $d_{\min} = 0.1$m. Perturbations acting on $(p_x(0), p_y(0), \theta(0))$ of the three mission cases was supposed to be normally distributed on $[-1, 1\text{m}]$, $[-0.5, 0.5\text{m}]$ and $[-0.0873, 0.0873\text{rad}]$, respectively. As investigated in [23], the specification of network structural parameters such as N_l, N_n and the number of sequential states N_{se} are likely to result in a significant impact on the approximation performance. Hence, extensive test trials and analysis were performed to determine a proper combination of these parameters using the strategy proposed in [35]. Based on the obtained results, we construct the GRURDNN network with $(N_l, N_n, N_{\text{se}}) = (5, 64, 4)$. Network with more complex structure cannot bring apparent improvement with respect to the approximation ability for the considered problem. It should be noted that the best combination of these parameters may vary among problems, but same determination strategy can be applied.

B. Performance Verification of the motion planner

The performance of applying the proposed GRURDNN-based motion prediction algorithm for the mobile robot autonomous exploration problem is tested and verified in this subsection. Due to the nature of the GRURDNN-based optimal motion predictor, an exploration trajectory can always be generated. Therefore, the solution feasibility as well as the optimality become two main aspects to evaluate the GRURDNN approximation performance.

Fig. 5 to Fig. 8 illustrate the obtained exploration trajectories, variation histories between the actual and pre-planned optimal solutions (e.g., $\boldsymbol{e} = [e_x, e_y, e_\theta]^T$), and velocity/acceleration-related profiles. First, attention is given to the feasibility of the obtained GRURDNN solutions. From the profiles shown in the left column of Figs. 5-8, a collision-free maneuver trajectory can be produced by applying the proposed method for all the three test scenarios. That is, the mobile robot is able to move from its initial position to the final target point without colliding with obstacles existing in the exploration map. Besides, from the linear and angular velocity/acceleration profiles, it it obvious that the variable path constraints are satisfied during the entire maneuver process, which further confirms the feasibility of the obtained exploration solution.

Next, we focus on the optimality of the obtained exploration solutions. It can be seen from the optimization formulation (9) that the command controls (a_v, a_ω) appear in the motion dynamics linearly. Moreover, they do not appear in the path constraint explicitly. As a result, a bang-singular-bang mode can be expected in the command control profiles over $t \in [0, t_f]$ if a time-optimal solution is found. According to the acceleration the corresponding linear velocity profiles displayed in the right column of Figs. 5-8, we can see that the proposed motion planner is able to generate a near-optimal time-optimal exploration trajectory for all the test cases.

Furthermore, to highlight the advantages of applying GRURDNN, comparative studies were carried out. The methods selected for comparison are the fully-connected DNN-based motion planner established in [23], and the NMPC tracking scheme constructed in [18], [20]. The corresponding results are demonstrated in Fig. 5 to Fig. 8, from where it is obvious that all the methods studied in this paper can produce

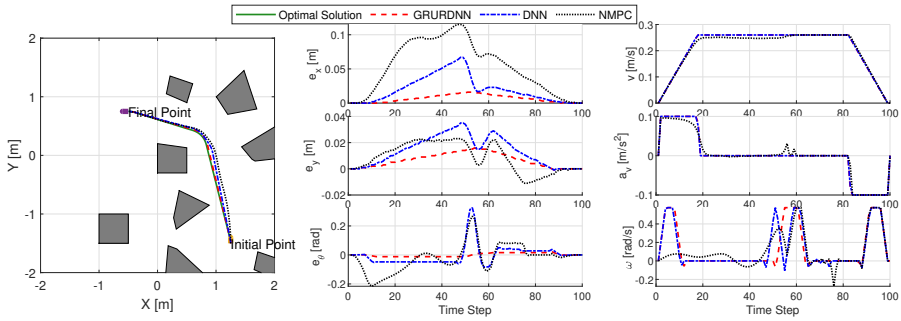


Fig. 5: Autonomous exploration results: Scenario 1

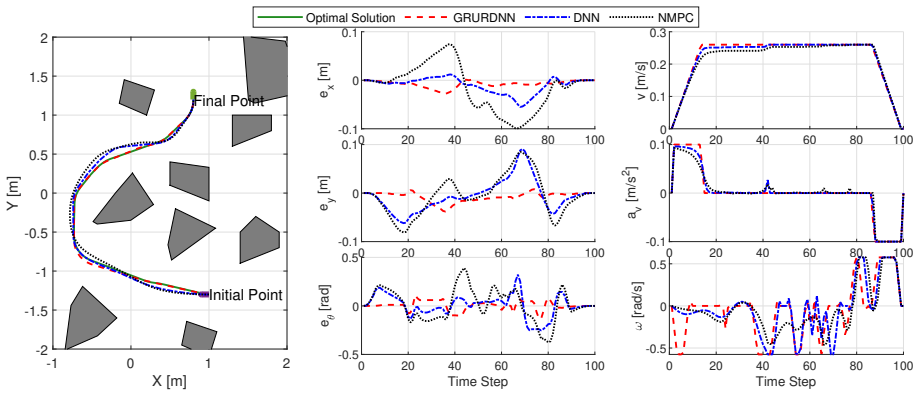


Fig. 6: Autonomous exploration results: Scenario 2

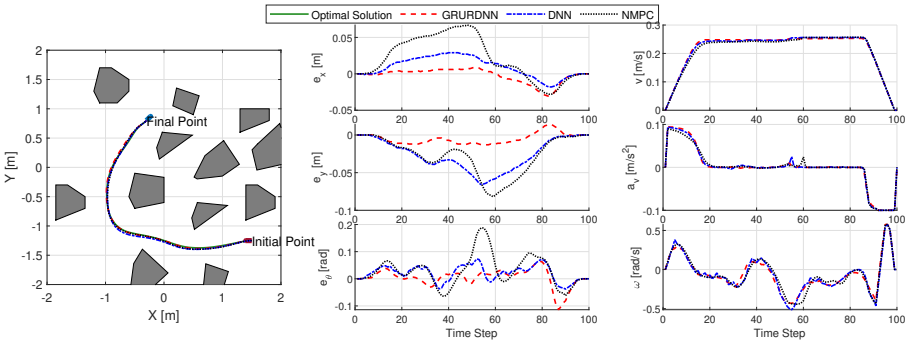


Fig. 7: Autonomous exploration results: Scenario 3

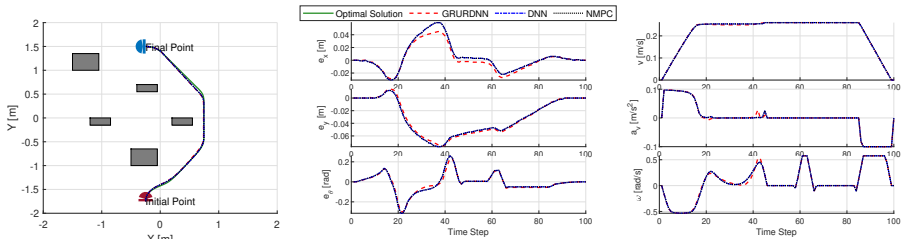


Fig. 8: Autonomous exploration results: Scenario 4

feasible exploration solutions for the considered mission cases. In terms of the algorithm performance, although differences can be identified among obtained solutions, the GRURDNN is more likely to produce an exploration trajectory closer to the pre-planned optimal solution. To further evaluate the performance of different algorithms, we pay attention to the objective function value, the accumulation of variation (e.g., $\int_{t_0}^{t_f} \boldsymbol{e}(t)dt$), and the average execution time required for each time step t_{cpu}. These results are tabulated in Table III.

It can be seen from Table III that the proposed method can better approximate the pre-planned optimal solution than the fully-connected DNN-based method suggested in [23], as

TABLE III: Performance evaluation of different algorithms

Method	Scenario 1		
	J^*	$\int_{t_0}^{t_f} \boldsymbol{e}(t)dt$	t_{cpu}
GRURDNN	15.440s	0.0044	12.6ms
DNN	15.510	0.0950	10.6ms
NMPC	16.016	0.2200	727.7ms
Method	Scenario 2		
	J^*	$\int_{t_0}^{t_f} \boldsymbol{e}(t)dt$	t_{cpu}
GRURDNN	20.526	0.0520	13.2ms
DNN	20.669	0.3135	10.6ms
NMPC	21.812	0.6967	841.2ms
Method	Scenario 3		
	J^*	$\int_{t_0}^{t_f} \boldsymbol{e}(t)dt$	t_{cpu}
GRURDNN	20.851	0.0262	13.7ms
DNN	20.974	0.0556	10.8ms
NMPC	21.211	0.1444	956.4ms
Method	Scenario 4		
	J^*	$\int_{t_0}^{t_f} \boldsymbol{e}(t)dt$	t_{cpu}
GRURDNN	12.667	0.0021	12.5ms
DNN	12.876	0.0131	9.87ms
NMPC	12.975	0.0467	653.2ms

lower values of J^* and $\int_{t_0}^{t_f} \boldsymbol{e}(t)dt$ are obtained. This can be attributed to the recurrent network structure with GRU mechanism. However, the price we paid for this performance enhancement is that the computation time required for each step tends to experience a slight increase, which is still in an acceptable level. On the contrary, the performance of NMPC algorithm is much worse than the two DNN-based direct re-calling strategies. Due to the consideration of different system constraints and multiple collision avoidance constraints, some NMPC online re-optimization process might not be maturely solved. Thus, the control performance is likely to become worse and the required computation time tends to increase. In summary, the above results and discussions not only confirm the feasibility of the proposed GRURDNN motion planner, but also appreciate its enhanced approximation performance. Consequently, it is advantageous to implement the proposed GRURDNN-based approach to guide the mobile robot to fulfill the time-optimal autonomous exploration problem.

C. Experimental Setup

It is worth noting that simply considering collision-free constraints for expected obstacles might result in significant safety issues for the mobile robot. This is because in real-world scenarios, an object can suddenly appear around the robot due to the uncertainty of the environment or inaccurate map information. Therefore, we perform the real-world experiments in an uncertain environment where unexpected obstacles will suddenly appear around the robot.

In the real world experiment, we used Turtlebot3 Waffle Pi mobile robot, as shown in Fig. 9, to validate the proposed hierarchical GRURDNN-DRL control framework. Turtlebot3 supports open-source robot operating system (ROS). The proposed algorithms were implement under ROS. The computing platform of Turtlebot3 is Raspberry Pi 3 board, which receives and executes the command from the host computer with Nvidia GTX 1080 GPU and Intel Core i9 CPU (2.9 GHZ). Turtlebot3 is equipped with a 360° laser scanner for sensing the environment. The lidar detecting range was configured as 1 meter in this experiment.

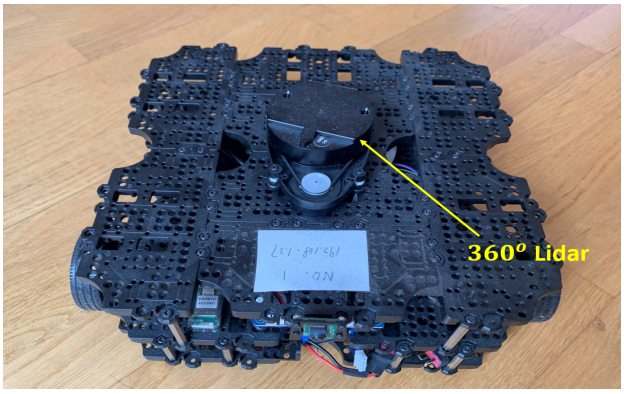


Fig. 9: Experimental mobile robot: TurtleBot3 Waffle Pi.

D. Validation of the Collision-Free DRL-based Control

The DRL-based collision avoidance algorithm was trained in Gazebo simulator with two environments, as shown in Fig. 10 and Fig. 15. The turtlebot mobile robot needs to reach target while avoiding all the obstacles illustrated via cubes, cuboids, cylinders, etc. The target is represented by red ball in the top-right corner. The laser detection range was shown as the blue rays. It was assumed that mobile robot will only detect the obstacles in front of it and its detection range is 180 degree. In environment 1, the target is placed at a fixed position, as shown in Fig. 10. In environment 2, when mobile robot collides with collision, the red ball position will be reset randomly outside of the maze and mobile robot will be placed in the middle of maze by default. In terms of the reward function parameters, the threshold values d_{rgmin} = 0.2 m, d_{romin} = 0.20 m, v_{amax} = 0.8 rad/s, v_{lmin} = 0.052 m/s are set according to the geometry and capacity of Turtlebot3. Note that these values can be reconfigured for other robotic platforms. We set N = 512, λ = 0.2, ϵ = 0.2 and ϵ_D = 0.4 in DRL algorithm based on empirical experience. The reward and Q value of proposed algorithm are compared with vanilla PER algorithm in both environment, as shown in Fig. 11, Fig. 12, Fig. 16 and Fig. 17, respectively. In environment 1, the proposed algorithm achieve average reward value 5 in less than 10000 steps, however, using the vanilla PER needs around 20000 training steps. The Q value has the same trend of reward. In environment 2, it is found that the average reward of the proposed algorithm arrives at 6 only using 16000 steps and the vanilla PER algorithm needs 31000 steps. The reason is that the proposed algorithm introduces more exploring action by using noisy term in the sampling weight in order to improve the training speed of PER algorithm. The Q value has the same trend of reward. It can be found that the proposed algorithm outperforms the vanilla PER algorithm in terms of training speed.

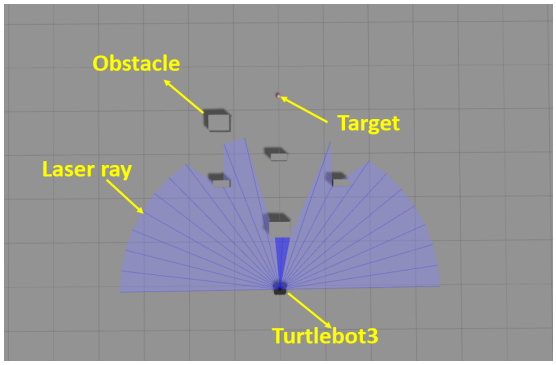


Fig. 10: Gazebo environment 1.

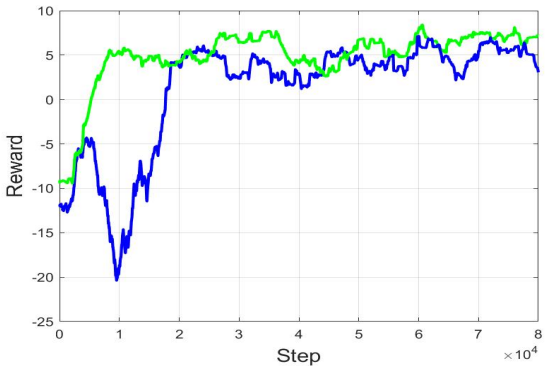


Fig. 11: Reward comparison in environment 1: DDPG+PER versus the proposed algorithm.

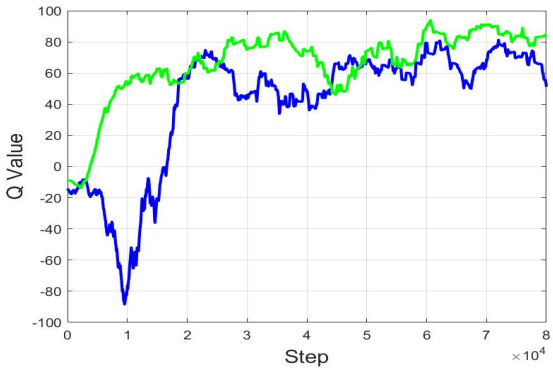


Fig. 12: Q value comparison in environment 1: DDPG+PER versus the proposed algorithm.

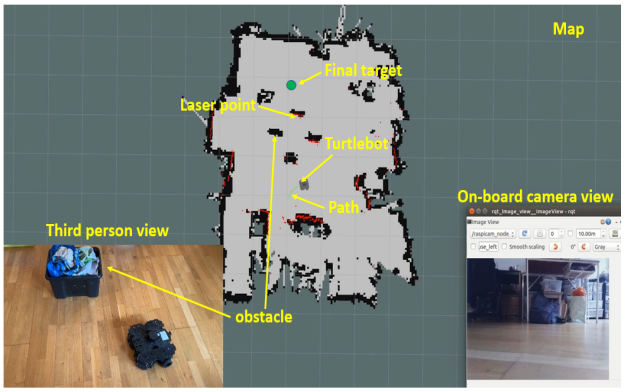


Fig. 13: Algorithm validation in real environment 1.

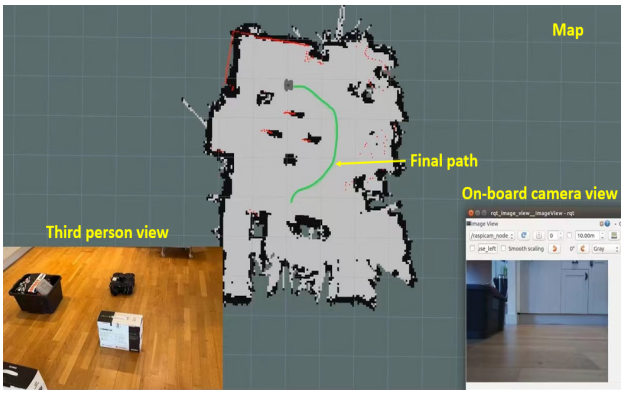


Fig. 14: Final trajectory of environment 1.

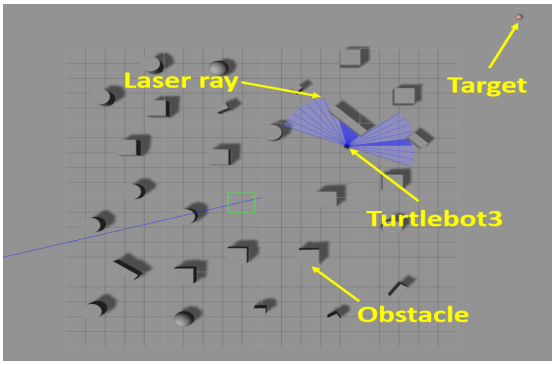


Fig. 15: Gazebo environment 2.

E. Validation of the Proposed Algorithms in Real Environment

The trained model was transferred to the real robot directly without fine-tuning and tested in a living room environment. Both static environment (environment 1) and suddenly changed environment (environment 2) are tested. Turtlebot published its velocity, heading, laser data to ROS master that is a laptop running deep reinforcement learning model and ROS master publishes the velocity and heading command to mobile robot. The living room map was pre-scanned using gmapping ROS node, and an optimal path was calculated by the proposed deep learning algorithm. The mobile robot will follow the pre-planned path to reach it. If an obstacle appears around the robot and blocks the way to the target position, the DRL-based algorithm will be triggered to avoid collision and navigate the robot to the next target waypoint.

As shown in Fig. 13 and Fig. 18, the rviz software, third person view and on-board camera view were recorded. The robot localization was realized using ROS built-in adaptive Monte Carlo localization (AMCL) algorithm. The turtlebot mobile robot needs to avoid all the static obstacles and also an unexpected obstacle which is manually placed during the mission. The static obstacles are pre-scanned using ROS built-in gmapping algorithm and they are represented as black area on the map. The red dots represent laser points. As we can see, only a certain range of the obstacle can be scanned around the turtlebot during the mission. The final target was illustrated as green circle.

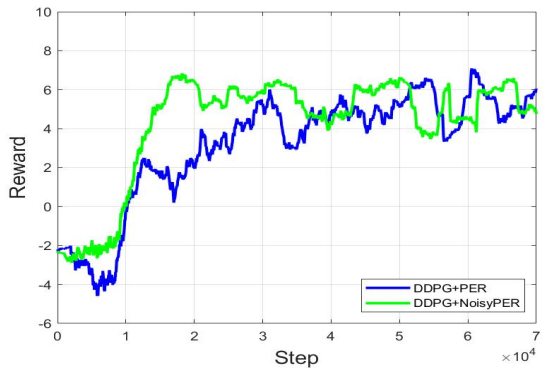


Fig. 16: Reward comparison in environment 2: DDPG+PER versus the proposed algorithm.

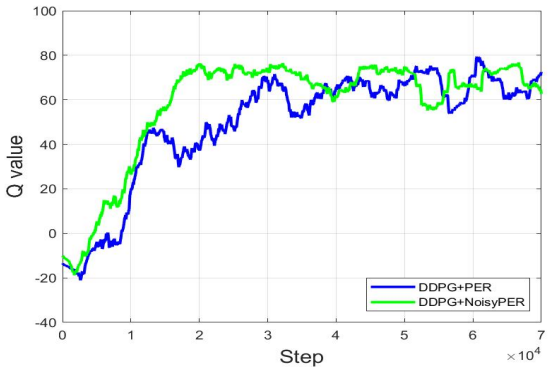


Fig. 17: Q value comparison in environment 2: DDPG+PER versus the proposed algorithm.

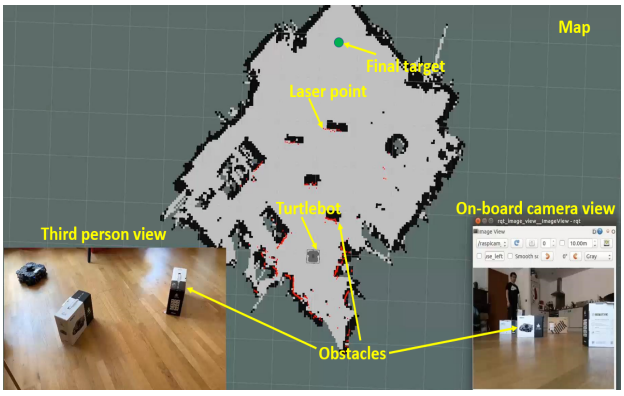


Fig. 18: Algorithm validation in real environment 2.

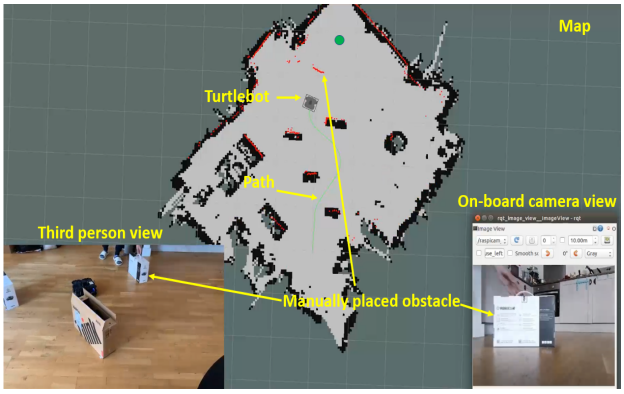


Fig. 19: Manually placed obstacle during the mission.

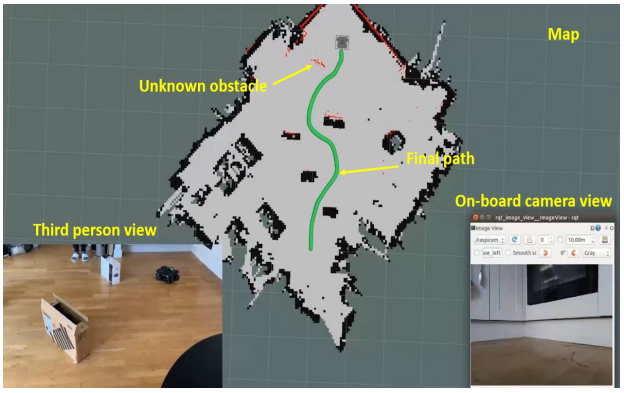


Fig. 20: Exploration trajectory of the mission in environment 2.

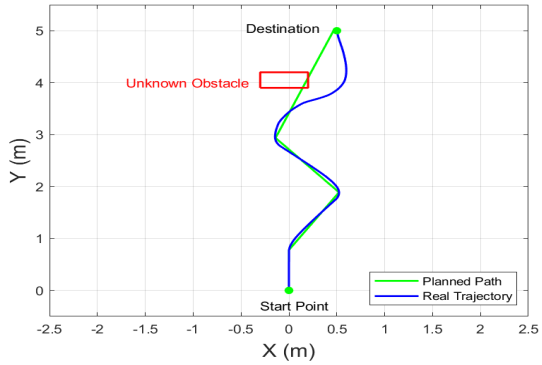


Fig. 21: Comparison between GRURDNN path and real trajectory in environment 2.

In environment 1, all the obstacles were static and obstacles positions were similar with the Scenario 4 in Section V.B. The path was generated by the proposed path planning algorithm and mobile robot followed the path to avoid the static obstacles. As shown in Fig. 14, the final path was similar to the path generated in Scenario 4 which validated the proposed path planning algorithm. In environment 2, after the mobile robot avoided the fourth obstacle, we manually placed one obstacle in front of the robot and the obstacle edge was scanned as a red line, as shown in Fig. 19. It is worth noting that the placed obstacle was not marked as black area, indicating that the placed obstacle was unknown to the mobile robot. Deep reinforcement learning algorithm was then triggered to navigate the robot so as to reach the final target. Note that due to the limitation of rviz software, the turtlebot path was originally recorded as light and thin green line, as shown in Fig. 19. For better illustration, we highlighted the actual robot trajectory in green as shown in Fig. 20.

The odometry data of turtlebot was also recorded and shown in Fig. 21. The real trajectory is shown as blue line and the pre-planned GRURDNN path is represented as green line. The maximum distance between the planned path and the real trajectory is around 4.5 cm that is acceptable for the mobile robot to operate in the 25 m^2 size area. When unknown obstacles occurs in the map, shown as red rectangle,

DRL algorithm was triggered to navigate the mobile robot to avoid it finally. As the DRL method is end-to-end based method and doesn't need to compute ESDF information and optimization equation, its computing time is only 1.2 ms, that is comparable with the state-of-art EGO-planner that is 0.81 ms. Note that all of our codes are developed under python as the deep learning library tensorflow is also developed using python. EGO-planner is developed using C++ that is much more computationally efficient than python. We believe that our DRL method can be equivalent or even faster than EGO-planner if it is re-developed under C++ code in the future. In total, it can be observed that the mobile robot was able to avoid both pre-scanned and unexpected obstacles throughout the way to the destination.

Comparing with conventional control algorithm that needs a specific rule for mobile robot to follow, for example, we could specify that the translational velocity is proportional to the inverse of angular velocity, deep reinforcement learning algorithm does not need a specific rule to follow and it can explore the optimal control policy by itself. For instance, in the experimental video, we can notice that when we placed the unexpected obstacles in front the turtlebot, it slowed down immediately and made a big turn to avoid the obstacle by itself and we didn't design a specific control law of translational velocity and angular velocity.

Experimental video was also recorded to support this conclusion. Interested readers can find it at supplementary material or via the link: https://youtu.be/Wbpj8RxhsmA.

VI. Conclusion

In this article, the problem of time-optimal autonomous exploration for mobile robot operating in an unknown environment has been investigated. An upper-lower hierarchical deep learning-based control framework is designed which leverages a GRURDNN motion planning algorithm and a DRL-based online collision avoidance approach. The proposed GRURDNN motion planning scheme was trained offline on a pre-generated optimized maneuver trajectory dataset such that it can predict the optimal motion in real-time. In this way, the time-consuming online optimization process can be omitted. Subsequently, the DRL-based online control scheme is applied to deal with unexpected obstacles which are frequently identified in real-world scenarios. After performing a number of simulation and experimental studies, some concluding remarks can be summarised:

(i) It is advantageous to embed the GRU mechanism in the DNN-based motion planner, as the inherent relationships between optimized robot state and control time histories are better exploited.

(ii) The designed DRL-based online control scheme is able to directly learn the control policy from human demonstration data and generate more exploration than vanilla PER algorithm, thereby reducing the time required by the training process.

(iii) The proposed hierarchical deep learning-based control framework has the capability of fulfilling different autonomous exploration cases effectively and efficiently in the presence of both expected and unexpected obstacles.

Consequently, we believe the proposed control scheme can serve as a promising tool for solving the mobile robot autonomous exploration problem. In the future work, the proposed strategy can be extended to address multi-mobile robot trajectory planning problems, where the collision avoidance among the mobile robots should be considered. Moreover, transfer learning can be used to extend the network as a motion planner for different scenarios. In terms of deep reinforcement learning, 2D lidar data can be extended to 3D point cloud data that can be used as input of the neural network to perceive the 3D environment. To improve the training efficiency, federated learning using multiple agents can also be applied.

References

[1] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car-Part I: Distributed system architecture and development process," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7131–7140, Dec. 2014.

[2] Y. Li, R. Cui, Z. Li, and D. Xu, "Neural network approximation based near-optimal motion planning with kinodynamic constraints using rrt," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 11, pp. 8718–8729, Nov. 2018.

[3] C. Yang, Z. Li, R. Cui, and B. Xu, "Neural network-based motion control of an underactuated wheeled inverted pendulum model," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 11, pp. 2004–2016, Nov. 2014.

[4] S. Liu, Z. Hou, T. Tian, Z. Deng, and Z. Li, "A novel dual successive projection-based model-free adaptive control method and application to an autonomous car," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3444–3457, Nov. 2019.

[5] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car-Part II: A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 8, pp. 5119–5132, Aug. 2015.

[6] L. Chen, Y. Shan, W. Tian, B. Li, and D. Cao, "A fast and efficient double-tree rrt*-like sampling-based planner applying on mobile robotic systems," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 6, pp. 2568–2578, Dec. 2018.

[7] L. Zuo, Q. Guo, X. Xu, and H. Fu, "A hierarchical path planning approach based on a$*$ and least-squares policy iteration for mobile robots," *Neurocomputing*, vol. 170, pp. 257–266, 2015.

[8] G. P. Kontoudis and K. G. Vamvoudakis, "Kinodynamic motion planning with continuous-time q-learning: An online, model-free, and safe navigation framework," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 12, pp. 3803–3817, Dec. 2019.

[9] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2021.

[10] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[11] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[12] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, and Y. Xia, "Two-stage trajectory optimization for autonomous ground vehicles parking maneuver," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3899–3909, Jul. 2019.

[13] C. Lian, X. Xu, H. Chen, and H. He, "Near-optimal tracking control of mobile robots via receding-horizon dual heuristic programming," *IEEE Transactions on Cybernetics*, vol. 46, no. 11, pp. 2484–2496, Nov. 2016.

[14] J. Li, Q. Yang, B. Fan, and Y. Sun, "Robust state/output-feedback control of coaxial-rotor mavs based on adaptive nn approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 12, pp. 3547–3557, Dec. 2019.

[15] L. Liu, D. Wang, Z. Peng, C. L. P. Chen, and T. Li, "Bounded neural network control for target tracking of underactuated autonomous surface vehicles in the presence of uncertain target dynamics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 1241–1249, Apr. 2019.

[16] T. Weiskircher, Q. Wang, and B. Ayalew, "Predictive guidance and control framework for (semi-)autonomous vehicles in public traffic," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2034–2046, Nov. 2017.

[17] N. Wan, C. Zhang, and A. Vahidi, "Probabilistic anticipation and control in autonomous car following," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 1, pp. 30–38, Jan. 2019.

[18] Z. Sun and Y. Xia, "Receding horizon tracking control of unicycle-type robots based on virtual structure," *International Journal of Robust and Nonlinear Control*, vol. 26, no. 17, pp. 3900–3918, 2016.

[19] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3263–3274, Nov. 2016.

[20] Z. Sun, L. Dai, K. Liu, Y. Xia, and K. H. Johansson, "Robust mpc for tracking constrained unicycle robots with additive disturbances," *Automatica*, vol. 90, pp. 172–184, 2018.

[21] R. Soloperto, J. Köhler, F. Allgöwer, and M. A. Müller, "Collision avoidance for uncertain nonlinear systems with moving obstacles using robust model predictive control," in *Proc. Eur. Control Conf.*, Jun. 2019, Naples, Italy, pp. 811–817.

[22] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, May. 2021.

[23] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Design and implementation of deep neural network-based control for automatic parking maneuver process," *IEEE Transactions on Neural Networks and Learning Systems*, 2020, to be published, doi: 10.1109/TNNLS.2020.3042120.

[24] H. Hu, S. Song, and C. L. P. Chen, "Plume tracing via model-free reinforcement learning method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 8, pp. 2515–2527, Aug. 2019.

[25] J. Zhu, J. Zhu, Z. Wang, S. Guo, and C. Xu, "Hierarchical decision and control for continuous multitarget problem: Policy evaluation with action delay," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 464–473, Feb. 2019.

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

[27] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6008–6014.

[28] H. Niu, Z. Ji, F. Arvin, B. Lennox, H. Yin, and J. Carrasco, "Accelerated sim-to-real deep reinforcement learning: Learning collision avoidance from human player," in *2021 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2021, pp. 144–149.

[29] M. S. Catherine and E. Lucet, "A modified hybrid reciprocal velocity obstacles approach for multi-robot motion planning without communication," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, Conference Proceedings, pp. 5708–5714.

[30] H. Eslamiat, Y. Li, N. Wang, A. K. Sanyal, and Q. Qiu, "Autonomous waypoint planning, optimal trajectory generation and nonlinear tracking control for multi-rotor uavs," in *2019 18th European Control Conference (ECC)*, 2019, pp. 2695–2700.

[31] Y. Li, H. Eslamiat, N. Wang, Z. Zhao, A. K. Sanyal, and Q. Qiu, "Autonomous waypoints planning and trajectory generation for multi-rotor uavs," in *Proceedings of the Workshop on Design Automation for CPS and IoT*, ser. DESTION '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 31–40.

[32] R. Jiang, H. Zhou, H. Wang, and S. S. Ge, "Road-constrained geometric pose estimation for ground vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 748–760, Apr. 2020.

[33] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[34] C. Sun, Q. Li, and L. Li, "A gridmap-path reshaping algorithm for path planning," *IEEE Access*, vol. 7, pp. 183 150–183 161, 2019.

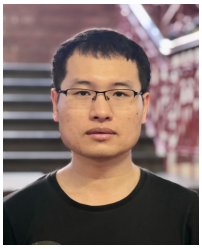
[35] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Six-dof spacecraft optimal trajectory planning and real-time attitude control: A deep neural network-based approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 5005–5013, Nov. 2020.

[36] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F. Wang, "Capturing car-following behaviors by deep learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 910–920, Mar. 2018.

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

[38] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

Runqi Chai (Member, IEEE) received the Ph.D. degree in Aerospace Engineering from Cranfield University, Cranfield, U.K, in August 2018. He was a research fellow at Cranfield University from 2018 to 2021. He is currently a professor at Beijing Institute of Technology. His research interests include trajectory optimization, networked control systems, multi-agent control systems, and autonomous vehicle motion planning and control.



Hanlin Niu (Member, IEEE) received the B.Eng. degree in Mechanics from Tianjin University in 2012 and the Ph.D degree in Aeronautical Engineering from Cranfield University in 2018. From July 2017 to January 2020, he worked as a Research Associate in Robotics at Cardiff University. Since January 2020, he has been working as a Research Associate in Robotics at the University of Manchester, being involved in Robotics and AI in Nuclear project, one of the four big robotics and AI projects funded by the Engineering and Physical Sciences Research Council. Currently, he is a member of Robotics for Nuclear Environments project (EPSRC no. EP/P01366X/1). His research interests include path planning, path following, collision avoidance, deep reinforcement learning of autonomous vehicles and tele-operation of robotic arm.

Joaquin Carrasco (Member, IEEE) was born in Abarn, Spain, in 1978. He received the B.Sc. degree in physics and the Ph.D. degree in control engineering from the University of Murcia, Murcia, Spain, in 2004 and 2009, respectively. He is a Reader with the Control Systems Centre, Department of Electrical and Electronic Engineering, University of Manchester, U.K. From 2009 to 2010, he was with the Institute of Measurement and Automatic Control, Leibniz Universitt Hannover, Hannover, Germany. From 2010 to 2011, he was a Research Associate with the Control Systems Centre, School of Electrical and Electronic Engineering, University of Manchester, U.K. His current research interests include absolute stability, multiplier theory, and robotics applications.

Farshad Arvin (Senior Member, IEEE) received the BSc degree in Computer Engineering, the MSc degree in Computer Systems Engineering, and the PhD degree in Computer Science, in 2004, 2010, and 2015, respectively. Farshad is an Associate Professor in Robotics in the Department of Computer Science at Durham University in UK. Prior to that, he was a Senior Lecturer in Robotics at The University of Manchester, UK. He visited several leading institutes including Artificial Life Laboratory with the University of Graz, Institute of Microelectronics, Tsinghua University, Beijing, and Italian Institute of Technology (iit) in Genoa as a Senior Visiting Research Scholar. His research interests include swarm robotics and autonomous multi-agent systems. He is the Founding Director of the Swarm & Computation Intelligence Laboratory formed in 2018, `www.SwaCIL.com`.

Hujun Yin (Senior Member, IEEE) is Professor of Artificial Intelligence at The University of Manchester. He received the BEng degree in Electronic Engineering in 1983 and the MSc degree in Signal Processing in 1986, both from Southeast University, and the PhD degree in Neural Networks in 1997 from University of York. His research interests include artificial intelligence, machine learning, neural networks, data analytics, and interdisciplinary and industrial applications. He has also been the Head of Business Engagement in AI and Data for the Faculty of Science and Engineering since 2019. He is a Turing Fellow of the Alan Turing Institute, a senior member of the IEEE, a member of the EPSRC Peer Review College, and the Vice Chair of the IEEE CIS UK Ireland Chapter.

Barry Lennox (Senior Member, IEEE) received the BEng degree in Chemical Engineering in 1991 and the PhD degree in Control Systems in 1996 from Newcastle University. He is a Professor of Applied Control and Nuclear Engineering Decommissioning and holds a Royal Academy of Engineering Chair in Emerging Technologies. He is a Fellow of the Royal Academy of Engineering, Fellow of the IET and InstMC, and a Chartered Engineer. He was the Director of the Robotics and Artificial Intelligence for Nuclear (RAIN) Hub and is an expert in applied control and its use in robotics and process operations and has considerable experience in transferring leading edge technology into industry.