Robotica (2023), **41**, pp. 1931–1946
doi:10.1017/S0263574723000127

RESEARCH ARTICLE

Potential field-based dual heuristic programming for path-following and obstacle avoidance of wheeled mobile robots

Yaoqian Peng, Xinglong Zhang*, Haibin Xie* and Xin Xu

College of Intelligence Science and Technology, National University of Defense Technology, Changsha, 410073, China
*Corresponding authors. E-mail: zhangxinglong18@nudt.edu.cn; xhb2575_sx@sina.com

Received: 28 June 2022; **Revised:** 8 September 2022; **Accepted:** 18 October 2022; **First published online:** 27 April 2023

Keywords: dual heuristic programming, wheeled mobile robots, path following, obstacle avoidance, potential field

Abstract
Path-following control of wheeled mobile robots has been a crucial research topic in robotic control theory and applications. In path-following control with obstacles, the path-following control and collision avoidance goals might be conflicting, making it challenging to obtain near-optimal solutions for path-following control and obstacle avoidance with low tracking error and input energy consumption. To address this problem, we propose a potential field-based dual heuristic programming (P-DHP) algorithm with an actor–critic (AC) structure for path-following control of mobile robots with obstacle avoidance. In the proposed P-DHP, the path-following control and collision avoidance problems are decoupled into two ones to resolve the control conflict. Firstly, a neural network-based AC is constructed to approximate the near-optimal path-following control policy in a no-obstacle environment. Then, with the trained path-following control policy fixed, a potential field-based control policy structure is constructed by another AC network to generate opposite control forces as the robot moves toward the obstacle, which can guarantee the robot's control safety and reduce the tracking error and input energy consumption in obstacle avoidance. The simulated and experimental results show that P-DHP can realize near-optimal path-following control with the satisfaction of safety constraints and outperforms state-of-the-art approaches in control performance.

1. Introduction

In recent years, mobile robots have been continuously expanded to industries, services, transportation, and other fields [1–4]. The application of mobile robots in complex, cluttered environments requires robots to avoid obstacles safely while following their desired path. Thus, the path-following and obstacle avoidance of wheeled mobile robots have been an active research direction and have been widely studied (see refs. [5–7]).

Among the recent works, the path-following and obstacle avoidance of mobile robots in complex environments can be achieved via two general methods [8]: reactive methods [9] and motion planning methods [10].

Motion planning methods, such as model predictive control (MPC) [10, 11], obtain a smooth collision-free path according to the receding horizon principle, which solves the optimal path-following control problem with collision avoidance. However, the classic MPC with numerical solvers will have a high computational cost, especially when using nonlinear dynamical models. Despite the fact that the computational efficiency can be enhanced by the improved MPC [12] with efficient solvers, this approach is still restrictive in environments with a large number of obstacles of different sizes. The reactive algorithms have advantages in the above scenarios. As a reactive method, the potential field-based (PF) approach [9], which treats each obstacle as a repulsive point to keep the robot away from it, is widely implemented in mobile robots. The PF methods for path-following and obstacle avoidance are addressed in refs. [8, 13, 14]. This approach, however, does not take into account the high-level objectives, such as energy consumption in obstacle avoidance. Moreover, its learning and optimization capabilities in complex environments need to be strengthened.

As a branch of machine learning, reinforcement learning (RL) and approximate dynamic programming (ADP) are advanced tools to solve nonlinear optimal control problems and have aroused the interest of researchers nowadays [15, 16]. Many works have been developed for path-following control with obstacle avoidance of wheeled mobile robots [17–19]. The integrated path-following control and obstacle avoidance of mobile robots can be transformed into a learning-based optimal control problem with constraints.

Related research works in the safe RL setting focus on optimal control with state constraints. These approaches transform the constrained optimal control problem into an unconstrained one, such as the reward/cost shaping-based RL approach, where the cost function is incorporated with safety guarantee terms (such as the Lyapunov functions [20] or barrier functions [18, 21, 22]) in the training process. By embedding costs of safety constraints into the reward function, a barrier-based online learning ADP algorithm was addressed in ref. [23] for optimal control problems with the safe guarantee. In ref. [24], a barrier function-based system transformation method was presented, transforming the original problem into an unconstrained one with virtual state variables. With the idea of constraint system transformation, a constrained cross-entropy-based approach was developed in ref. [22] for a safe RL problem with constraints defined as the expected cost. Some other exciting works for safety-critical systems may refer to [25], with the unification of control barrier functions and control Lyapunov functions for automotive applications. In ref. [18], a model-based ADP method was proposed for motion planning with constraints of moving avoidance regions.

Noteworthy, these techniques may not deal with conflicting control objectives as the optimal control objective and constraint satisfaction are contradictory. For example, in path-following control of wheeled mobile robots with obstacle avoidance, ensuring both optimal tracking control and obstacle avoidance is a difficult task since the convergence of the actor–critic (AC) might not be guaranteed by the reward/cost shaping-based RL approach due to conflicting learning objectives. In addition, the energy loss and control costs are valuable to consider in the obstacle avoidance process.

In this paper, to decouple the impact of path-following control and obstacle avoidance on mobile robots, we propose a potential field-based dual heuristic programming (P-DHP) algorithm with two progressive AC structures to achieve a trade-off between control cost and safe performance. To decouple the conflicting learning goals, the training process was decoupled into two parts in the proposed P-DHP. Firstly, a path-following control policy was learned by value iteration with an initial AC based on the standard neural network. Then, with the neural network weights fixed, the PF term is used to construct another AC structure to learn a safe control policy with collision avoidance capability. Moreover, another cost is designed in the second part of the learning process. This cost is minimized to reduce the energy loss and state errors of the robot with obstacle avoidance.

Different from ref. [19], in our approach, the control policy's learning mode for path-following control and obstacle avoidance is asynchronous to guarantee convergence in the case of conflicting learning objectives. Besides, this paper also contributes to the optimization of reducing the energy loss and state errors in the obstacle avoidance process.

The remainder of the paper is constructed as follows. Section II formulates the tracking error model and problem formulation. In Section III, we present the proposed P-DHP approach for mobile robots, while the simulated and experimental results are demonstrated in Section IV.

Notation

For a vector $h \in \mathbb{R}^m$, $\|h\|_2$ is defined as the Eulidean norm, and $\|h\|_R^2$ is denoted as $h^\top Rh$. For a function $\phi(\nu(k))$ with an input variable $\nu(k)$, we denote $\phi(k)$ to represent $\phi(\nu(k))$ for short.

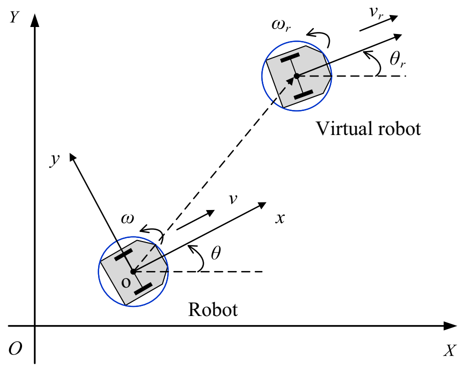

Fig. 1. *Tracking error model of the wheeled robot.*

2. Tracking error model and problem formulation

2.1. Tracking error model

The wheeled mobile robot is shown in Fig. 1, and it is assumed that its particle and center point are coincident. The model is described as

$$[\dot{x}, \dot{y}, \dot{\theta}, \dot{v}, \dot{\omega}]^\top = [v\cos\theta, v\sin\theta, \omega, a_v, a_\omega]^\top, \tag{1}$$

where $q = (x, y)$ is the position of the robot in the global coordinate frame and θ is the heading angle related to the robot's kinematic model. v and ω denote the linear velocity and angular velocity of the robot, respectively. a_v and a_ω are the accelerations of v and ω, respectively, related to the dynamic model of the robot. The control input is $u = [a_v, a_\omega]^\top$, and the state of the robot is defined as $p = [x, y, \theta, v, \omega]^\top$.

Suppose a virtual robot guides the robot to move forward, and its state is described as $p_r = [x_r, y_r, \theta_r, v_r, \omega_r]^\top$, satisfying:

$$[\dot{x}_r, \dot{y}_r, \dot{\theta}_r, \dot{v}_r, \dot{\omega}_r]^\top = [v_r\cos\theta_r, v_r\sin\theta_r, \omega_r, a_{v,r}, a_{\omega,r}]^\top, \tag{2}$$

where $u_r = [a_{v,r}, a_{w,r}]^\top$ is the control input of the virtual leader robot.

The tracking error model of the robot can be written as

$$e = T(p_r - p), \tag{3}$$

where T is the coordinate transformation matrix from the global coordinate frame XOY to the robot's local coordinate frame xoy, which is

$$T = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & I_3 \end{bmatrix}. \tag{4}$$

Then, we can get the tracking error model in derivative form by differentiating both sides of (3), which is

$$\dot{e} = T(\dot{p}_r - \dot{p}) + \dot{T}(p_r - p). \tag{5}$$

According to Eqs. (1)–(2) and (5), the discrete-time tracking error model according to the forward-Euler discretization can be written as

$$\begin{cases} e_x(k+1) = e_x(k) + \Delta h\left(\omega(k)\, e_y(k) - v(k) + v_r(k)\cos e_\theta(k)\right) \\ e_y(k+1) = e_y(k) + \Delta h\left(-\omega(k)\, e_x(k) + v_r(k)\sin e_\theta(k)\right) \\ e_\theta(k+1) = e_\theta(k) + \Delta h\left(\omega_r(k) - \omega(k)\right) \\ e_v(k+1) = e_v(k) + \Delta h\left(a_{v,r}(k) - a_v(k)\right) \\ e_\omega(k+1) = e_\omega(k) + \Delta h\left(a_{\omega,r}(k) - a_\omega(k)\right), \end{cases} \tag{6}$$

where $e = [e_x, e_y, e_\theta, e_v, e_\omega]^\top$, $v(k) = v_r(k) - e_v(k)$, $\omega(k) = \omega_r(k) - e_\omega(k)$, k is the discrete-time index, and Δh is the adopted sampling interval.

The nonlinear tracking error model can be abbreviated as

$$e(k+1) = f(e(k)) + g(u(k)), \tag{7}$$

where f is the error transition function, and g is the input mapping function.

2.2. Problem formulation

We define the performance index of path-following control as

$$J(e(k)) = \sum_{k=0}^{\infty} \gamma^k r(e(k), u(k)), \tag{8}$$

where

$$r(e(k), u(k)) = \|e(k)\|_Q^2 + \|u(k) - u_r(k)\|_R^2, \tag{9}$$

and $\gamma \in (0, 1]$ is the discounting factor, $Q^\top = Q \in \mathbb{R}^{5\times 5}$, $R^\top = R \in \mathbb{R}^{2\times 2}$, $Q \succ 0$, $R \succ 0$.

The cost $J(e(k))$ is minimized subject to the following constraints $q(k) \in \mathcal{X}, \forall k \in [1, \infty]$, where

$$\mathcal{X} = \{q \mid \|q - q_o\|_2 > r + r_o\}, \tag{10}$$

$q_o, o \in \mathbb{N}_1^n$, is the position of the obstacle o. r and r_o are the radii of the smallest approximate circles surrounding the robot and obstacle o, $o \in \mathbb{N}_1^n$, respectively.

Below, we give the definition of the potential field function and the assumption on the detection rules for the robot.

Definition 1 *(Potential field function)*:

For the robot's position q and the obstacle's position $q_o \in \mathbb{R}^2$, $o \in \mathbb{N}_1^n$, the potential field function $\mathcal{B}_o$ is defined as

$$\mathcal{B}_o(q) = \begin{cases} \frac{1}{2}\kappa_o\left(\frac{1}{\|q - q_o\|_2} - \frac{1}{r_o + r}\right)^2, & \|q - q_o\|_2 \le r_o + r \\ 0, & \|q - q_o\|_2 > r_o + r, \end{cases} \tag{11}$$

where κ_o is the relaxation factor of the potential field. $\|q - q_o\|_2$, $o \in \mathbb{N}_1^n$, represents the distance between the robot and the obstacle o. The depiction of the above parameters is shown in Fig. 2.

Assumption 1 *(Detection rules)*: The robot can detect the position and angle of obstacles in a sector area (with the robot as the center and a radius of r_d).

3. P-DHP for path-following control with collision avoidance

3.1. Design of P-DHP

To solve the path-following control problem with safety constraints, we propose a control policy that has two control parts, which is described as

$$u(k) = \bar{u}^{i_1}(e(k)) + \sum_{o=1}^{n} \eta_o(k) \nabla \mathcal{B}_o(k), \tag{12}$$

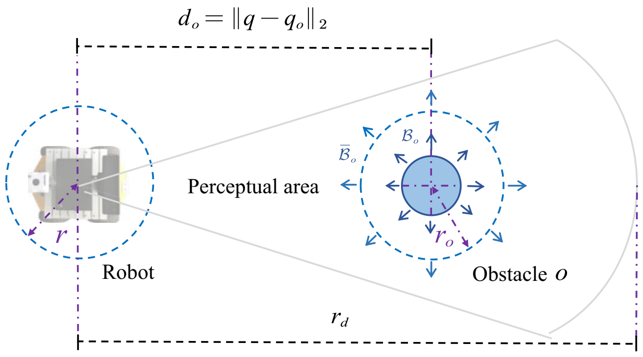

Fig. 2. *Depiction of robot's parameters in obstacle avoidance.*

where $\bar{u}^{i_1}(e(k))$ is a control policy that corresponds to a fixed weight matrix without the obstacles' constraints. $\eta_o \in \mathbb{R}^{2\times 2}$, $o \in \mathbb{N}_1^n$, are variables to be optimized for obstacles of different sizes and positions (see Section 3.2). $\nabla\mathcal{B}_o = \partial\mathcal{B}_o/\partial q$.

Remark 1: As the robot approaches the obstacle, the input item $\nabla\mathcal{B}_o$ could generate a repulsive force to drive $q(k)$ to satisfy the constraint (10). Meanwhile, the parameter η_o is continuously optimized to ensure safety in path tracking. In the case that the input item $\nabla\mathcal{B}_o$ is identically equal to 0, the policy $\bar{u}(k)$ is to approach the path-following control policy under no obstacle.

To decouple the conflicting learning objectives of path-following control and obstacle avoidance, the learning process is divided into two parts, where $\bar{u}$ and η_o in (12) are learned asynchronously by value iteration to guarantee convergence.

Then, the performance index of path-following control is redefined as

$$J_1(\bar{e}(k)) = \sum_{k=0}^{\infty} \gamma^k r_1(\bar{e}(k), \bar{u}(k)), \tag{13}$$

where $r_1(\bar{e}(k), \bar{u}(k)) = \|\bar{e}(k)\|^2_{Q_1} + \|\bar{u}(k) - u_r(k)\|^2_{R_1}$, $Q_1^\top = Q_1 \in \mathbb{R}^{5\times 5}$, $R_1^\top = R_1 \in \mathbb{R}^{2\times 2}$, $Q_1 \succ 0$, $R_1 \succ 0$. $\bar{e}(k+1)$ is the state of the robot that satisfies $\bar{e}(k+1) = f(\bar{e}(k)) + g(\bar{u}(k))$.

Define J_1^* as the optimal value function of J_1 under the optimal control policy $\bar{u}^*$. According to the Hamilton–Jacobi–Bellman (HJB) equation, one has

$$J_1^*(k) = \min_{\bar{u}(k)} \; r_1(\bar{e}(k), \bar{u}(k)) + \gamma J_1^*(k+1), \tag{14}$$

and

$$\bar{u}^*(k) = \operatorname*{argmin}_{\bar{u}(k)} \; r_1(\bar{e}(k), \bar{u}(k)) + \gamma J_1^*(k+1). \tag{15}$$

Define i as the number of iterations. Firstly, the control policy $\bar{u}(k)$ in the unconstrained scenario is learned within iterations $i = 1, \ldots, i_1$, and the update rule in each iteration of $J_1(k)$ and $\bar{u}(k)$ can be obtained as

$$J_1^{i+1}(k) = r_1(\bar{e}(k), \bar{u}^i(k)) + \gamma J_1^i(k+1), \tag{16}$$

$$\bar{u}^{i+1}(k) = \operatorname*{argmin}_{\bar{u}(k)} \; r_1(\bar{e}(k), \bar{u}(k)) + \gamma J_1^{i+1}(k+1), \tag{17}$$

where i_1 is the iterative value as $J_1^i(k)$ converges (see Algorithm 1 for details).

With the fixed control policy $\bar{u}^{i_1}(k)$, the variable η_o in the second item of (12) is updated within iterations $i = i_1, \ldots, i_2$ to minimize the reconstructed cost function $J_2(k)$ for collision avoidance constraints, which is

$$J_2(k) = \sum_{o=1}^{n} J_{2,o}(k), \tag{18}$$

$$J_{2,o}(k) = \sum_{k=0}^{\infty} \gamma^k r_2(\mathcal{B}_o(k), e(k), u(k)), \tag{19}$$

where $r_2(\mathcal{B}_o(k), e(k), u(k)) = \|e(k) - \bar{e}(k)\|^2_{Q_2} + \|u(k) - \bar{u}^{i_1}(k)\|^2_{R_2} + \mu\mathcal{B}_o(k)$, $Q_2^\top = Q_2 \in \mathbb{R}^{5\times5}$, $R_2^\top = R_2 \in \mathbb{R}^{2\times2}$, $Q_2 \succ 0$, $R_2 \succ 0$, and $\mu > 0$ is a tuning parameter.

Remark 2: The control objective is to minimize the cost J_2, which is equivalent to reducing the tracking error and input energy consumption while satisfying the obstacle avoidance constraint. The trade-off among obstacle avoidance, reduced tracking error and input energy consumption can be adjusted by the value of μ and the weight of matrix Q_2 and R_2, respectively.

Let $\overrightarrow{\eta}$ be the sequence of $\eta_1, \ldots, \eta_n$. Then J_2 and $\overrightarrow{\eta}$ can be updated in each iteration $i = i_1, \ldots, i_2$ by

$$J_2^{i+1}(k) = \sum_{o=1}^{n} r_2(\mathcal{B}_o(k), e(k), u^i(k)) + \gamma J_2^i(k+1), \tag{20}$$

$$\overrightarrow{\eta}^{i+1}(k) = \underset{\overrightarrow{\eta}(k)}{\text{argmin}} \sum_{o=1}^{n} r_2(\mathcal{B}_o(k), e(k), u(k)) + \gamma J_2^{i+1}(k+1). \tag{21}$$

The main steps of the proposed P-DHP can be seen in Algorithm 1.

In the following, we prove the convergence of our proposed P-DHP.

Theorem 1 (Convergence of (16) and (17)): *If the initial condition satisfies* $J_1^0(k) \geq r_1(\bar{e}(k), \bar{u}^0(k)) + \gamma J_1^0(k+1)$, *then it follows that*

1) $J_1^{i+1}(k) \leq J_1^i(k)$.

2) $\lim_{i\to\infty} J_1^i(k) = J_1^*(k)$.

Proof. For the proof details please refer to refs. [26] and [19]. □

Theorem 2 (Convergence of (20) and (21)) [19]: *If* $J_2^{i_1}(k) \geq \sum_{o=1}^{n} r_2(\mathcal{B}_o(k), e(k), u^{i_1}(k)) + \gamma J_2^{i_1}(k)$ *and* $u^{i_1}(k) = \bar{u}^{i_1}(k) + \sum_{o=1}^{n} \eta_o^{i_1}(k)\nabla\mathcal{B}_o(k)$ *is a safe control policy to restrain* $q^{i_1}(k) \in \mathcal{X}$, *then*

1) $J_2^{i+1}(k) \leq J_2^i(k)$.

2) $u^i(k) = \bar{u}^{i_1}(k) + \sum_{o=1}^{n} \eta_o^i(k)\nabla\mathcal{B}_o(k)$, $i \geq i_1$, *is a safe control policy.*

Proof. 1): The proof is similar to that in Theorem 1, which is to replace $J_1^i(k)$ with $J_2^i(k)$.

2): As $u^{i_1}(k)$ is a safe control policy, one has $J_2^{i_1}(k) \leq J_2^f$, where J_2^f is the threshold of the safe value function. According to Theorem 1, for $i \geq i_1$, it satisfies $J_2^i(k) \leq J_2^{i_1}(k) \leq J_2^f$, i.e., $q^i(k) \in \mathcal{X}$. Therefore, $u^i(k)$, $i \geq i_1$, is a safe control policy. □

3.2. AC learning for P-DHP

In this section, the AC learning structures (see Fig. 3) are introduced to implement Algorithm 1. The AC learning structures consist of two decoupling parts. The actor-1–critic-1 network corresponds to the iteration of $J_1(k)$ and $\bar{u}(k)$, and the actor-2–critic-2 network corresponds to the iteration of $J_2(k)$ and $\overrightarrow{\eta}$, respectively. The AC network weights are updated in an incremental learning mode.

The critic-1 network is constructed as

$$\nabla\hat{J}_1(k) = W_c^\top(k) h_c(\bar{e}(k)), \tag{22}$$

where the costate $\nabla\hat{J}_1(k) = \partial\hat{J}_1(k)/\partial\bar{e}(k)$, $W_c \in \mathbb{R}^{5\times5}$ is the weighting matrix, and h_c is the activation function.

Pseudocode of P-DHP.

Algorithm 1. Pseudocode of P-DHP.

Require: $\epsilon_1 > 0$, $\epsilon_2 > 0$, $i = 0$.

```
    Learn $\bar{u}$:
1:  Randomly initialize the control input $\bar{u}^0(1)$.
2:  for $k = 1, 2, \ldots$ do
3:      repeat
4:          Generate the control input $\bar{u}^i(k)$.
5:          Compute $\bar{e}(k+1)$ with $\bar{u}^i(k)$ via (6).
6:          Compute the reward $r_1(\bar{e}(k), \bar{u}^i(k))$.
7:          One step policy evaluation:
8:          $J_1^{i+1}(k) = r_1(\bar{e}(k), \bar{u}^i(k)) + \gamma J_1^i(k+1)$
9:          Control policy update:
10:         $\bar{u}^{i+1}(k) = \underset{\bar{u}(k)}{\operatorname{argmin}}\ r_1(\bar{e}(k), \bar{u}(k)) + \gamma J_1^{i+1}(k+1)$
11:         $i \leftarrow i+1$
12:     until $|J_1^{i-1}(k) - J_1^i(k)| \le \epsilon_1$
13:     $i_1 = i$
14: end for

    Learn $u$:
13: Randomly initialize the sequence $\overrightarrow{\eta}^{i_1}(1)$.
14: for $k = 1, 2, \ldots$ do
15:     repeat
16:         Generate the control input $u^i(k)$.
17:         Compute $e(k+1)$ with $u^i(k)$ via (6).
18:         Compute the reward $r_2(\mathcal{B}_o(k), e(k), u^i(k))$.
19:         One step policy evaluation:
            $J_2^{i+1}(k) = \sum_{o=1}^{n} r_2(\mathcal{B}_o(k), e(k), u^i(k)) + \gamma J_2^i(k+1)$
20:         Control policy update:
            $\overrightarrow{\eta}^{i+1}(k) = \underset{\overrightarrow{\eta}(k)}{\operatorname{argmin}}\ \sum_{o=1}^{n} r_2(\mathcal{B}_o(k), e(k), u(k)) + \gamma J_2^{i+1}(k+1)$
            $u^{i+1}(k) = \bar{u}^{i_1}(k) + \sum_{o=1}^{n} \eta_o^{i+1}(k)\nabla\mathcal{B}_o(k) \triangleright \bar{u}^{i_1}$ is a weight fixed control policy'
21:         $i \leftarrow i+1$
22:     until $|J_2^{i-1}(k) - J_2^i(k)| \le \epsilon_2$
23:     $i_2 = i$
24:     if $\mathcal{B}_o(k-1) \geqslant \mathcal{B}_o(k)$ then
25:         Set $\nabla\mathcal{B}_o(k) = 0$.
26:     end if
27: end for
```

Define $\nabla J_1^* = \partial J_1^*/\partial \bar{e}$. To minimize the bias between $\nabla \hat{J}_1$ and ∇J_1^*, we designed a target to be followed by $\nabla \hat{J}_1$, i.e.,

$$\nabla J_1^d(k) = 2Q_1\bar{e}(k) + \gamma \nabla f(k)^\top \nabla \hat{J}_1(k+1), \tag{23}$$

where $\nabla f = \partial f/\partial \bar{e}$.

Let the quadratic approximation error of the critic-1 network be defined as $E_c(k) = ||\nabla \hat{J}_1(k) - \nabla J_1^d(k)||_2^2$. The goal of the critic-1 network is to minimize the distance between $\nabla \hat{J}_1(k)$ and $\nabla J_1^d(k)$ by

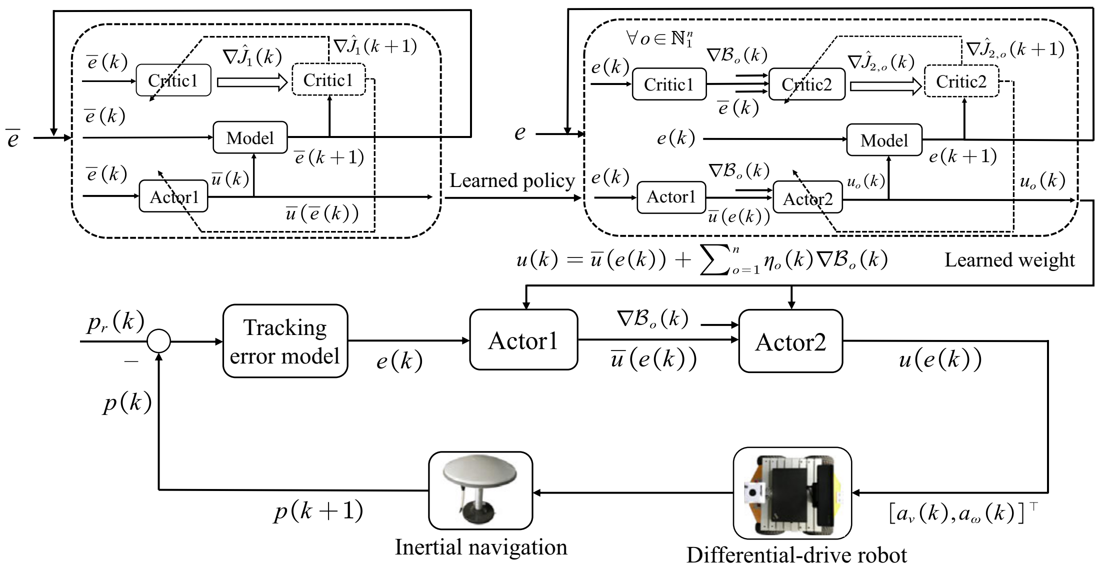

Fig. 3. *The structure of the actor–critic learning algorithm.*

updating weights. At each time instant k, the update rule of W_c is

$$W_c(k+1) = W_c(k) - \alpha_c \frac{\partial E_c(k)}{\partial W_c(k)}, \tag{24}$$

where α_c is the learning rate of W_c.

Then we design the actor-1 network for the robot, which is

$$\hat{\bar{u}}(k) = W_a^\top(k) h_a(\bar{e}(k)), \tag{25}$$

where the $W_a \in \mathbb{R}^{5\times 2}$ is the weighting matrix, and h_a is the activation function.

In view of (17), we define the $\bar{u}^d(k)$ as the desired value of $\hat{\bar{u}}(k)$, which is expressed as

$$\bar{u}^d(k) = -\frac{1}{2}\gamma R_1^{-1} \nabla g(k)^\top \nabla \hat{J}_1(k+1) + u_r(k), \tag{26}$$

where $\nabla g = \partial g / \partial \bar{u}$, and $\nabla \hat{J}_1$ is the output of the critic-1 network.

The quadratic approximation error of actor-1 network can be defined as $E_a(k) = ||\hat{\bar{u}}(k) - \bar{u}^d(k)||_2^2$, and the update rule of W_a is

$$W_a(k+1) = W_a(k) - \alpha_a \frac{\partial E_a(k)}{\partial W_a(k)}, \tag{27}$$

where α_a is the learning rate of W_a. Note that the learning process (27) is performed in i_1 iterations. We define the converged matrix W_a as $\mathbf{W}_a$ and the learned control policy $\hat{\bar{u}}$ as $\hat{\bar{\mathbf{u}}}$, respectively.

Suppose that the potential fields of obstacles do not overlap. In the following design, we decompose the actor-2–critic-2 network into n subnetworks to learn the variable η_o, $o \in \mathbb{N}_1^n$, in (12) respectively, which corresponds to obstacles o, $o \in \mathbb{N}_1^n$, of different sizes and positions.

The sub-critic-2 network for each obstacle o is given as

$$\nabla \hat{J}_{2,o}(k) = \mu_c \mathbf{W_c}^\top h_c(e(k)) + \rho_o(k) \nabla \mathcal{B}_o(k), \tag{28}$$

where $\rho_o \in \mathbb{R}^{2\times 2}$, and $\nabla \hat{J}_{2,o} = \partial \hat{J}_{2,o} / \partial e$. $\mathbf{W}_c$ is the learned matrix of W_c, and $\mu_c > 0$ is a tuning parameter.

The desired value of $\nabla \hat{J}_{2,o}$ i.e., $\nabla J_{2,o}^d$ is defined as

$$\nabla \hat{J}_{2,o}^d(k) = 2Q_2(e(k) - \bar{e}(k)) + \mu \left(\frac{\partial q(k)}{\partial e(k)} \right)^\top \nabla \mathcal{B}_o(k) + \gamma \nabla f(k)^\top \nabla \hat{J}_{2,o}(k+1). \tag{29}$$

where $\nabla f = \partial f / \partial e$.

Denote $E_{c,o}(k) = ||\nabla\hat{J}_{2,o}(k) - \nabla J^d_{2,o}(k)||^2_2$ as the quadratic approximation error of the sub-critic-2 network. The update rule of ρ_o can be obtained as

$$\rho_o(k+1) = \rho_o(k) - \beta_c \frac{\partial E_{c,o}(k)}{\partial \rho_o(k)}, \tag{30}$$

where β_c is the learning rate of ρ_o, $o \in \mathbb{N}^n_1$.

In view of (12), the sub-actor-2 network for each obstacle o is designed with the following form:

$$\hat{u}_o(k) = \mathbf{W}_\mathbf{a}^\top h_a(e(k)) + \eta_o(k)\nabla\mathcal{B}_o(k), \tag{31}$$

where $\hat{u}_o(k)$ is the control policy for the robot to avoid the obstacle o, which is equal to $\hat{u}(k)$ in the case of only one obstacle o to be avoided.

Define $\varepsilon_o = 2R_2(\hat{u}_o - \hat{\bar{\mathbf{u}}})$. Then the desired value of $\varepsilon_o(k)$ is redefined as

$$\varepsilon^d_o(k) = -\gamma\nabla g(k)^\top\nabla\hat{J}_{2,o}(k+1). \tag{32}$$

where $\nabla g = \partial g/\partial u$.

In the same way, the quadratic approximation error of the sub-actor-2 network and the update rule of η_o are given as

$$E_{a,o}(k) = ||\varepsilon_o(k) - \varepsilon^d_o(k)||^2_2, \tag{33}$$

$$\eta_o(k+1) = \eta_o(k) - \beta_a \frac{\partial E_{a,o}(k)}{\partial \eta_o(k)}, \tag{34}$$

where β_a is the learning rate of η_o, $o \in \mathbb{N}^n_1$.

Given that all variables η_o, $o \in \mathbb{N}^n_1$, have been trained until converging, the overall control policy to be deployed to the robot is

$$\hat{u}(k) = \mathbf{W}_\mathbf{a}^\top h_a(e(k)) + \sum\nolimits_{o=1}^{n} \eta_o(k)\nabla\mathcal{B}_o(k). \tag{35}$$

4. Simulation and experimental results

We first test our algorithm by numerical simulation in a MATLAB environment and deploy the control policy on a real-world differential-drive wheeled robot platform. Suppose the robot is equipped with sensors that can measure the obstacles' approximate size, relative distance, and angle in the simulation and experiment.

4.1. Simulation experiments

4.1.1. Parameter settings and training process

In the simulation, a relaxed potential function is defined and used as follows to guarantee the smoothness in the control process, i.e.,

$$\mathcal{B}^r_o(q) = \begin{cases} \mathcal{B}_o(q), & d_o \geq \delta_o \\ \sigma_o(q), & d_o < \delta_o, \end{cases} \tag{36}$$

where $d_o = \|q - q_o\|_2$ and $0 < \delta_o < d_o$ is an adjustable relaxing factor. $\sigma_o(q)$ is a derivative and monotonic function, where $\sigma_o(q) = \frac{1}{2}\sigma_1(d_o - \delta_o)^2 + \sigma_2(d_o - \delta_o) + \sigma_3$, $\sigma_1 = \nabla^2_{d_o}\mathcal{B}_o(q)|_{d_o=\delta_o}$, $\sigma_2 = \nabla_{d_o}\mathcal{B}_o(q)|_{d_o=\delta_o}$, $\sigma_3 = \mathcal{B}_o(q)|_{d_o=\delta_o}$.

Also, the perceived radius values of obstacles are properly expanded to construct $\nabla\bar{\mathcal{B}}_o(q)$, which replaces $\nabla\mathcal{B}_o(q)$ in (12) to ensure control safety. The reconstructed $\bar{\mathcal{B}}_o(q)$ is defined as

$$\bar{\mathcal{B}}_o(q) = \begin{cases} \frac{1}{2}\bar{\kappa}_o\left(\frac{1}{d_o} - \frac{1}{r_d}\right)^2, & d_o \leq r_d \\ 0, & d_o > r_d, \end{cases} \tag{37}$$

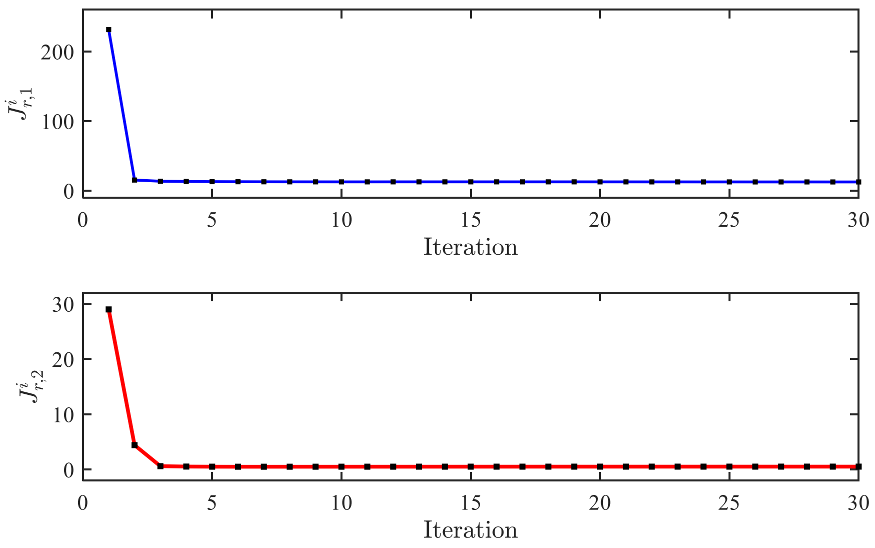

Fig. 4. *Variations of $J^i_{r,1}$ and $J^i_{r,2}$ in the learning process.*

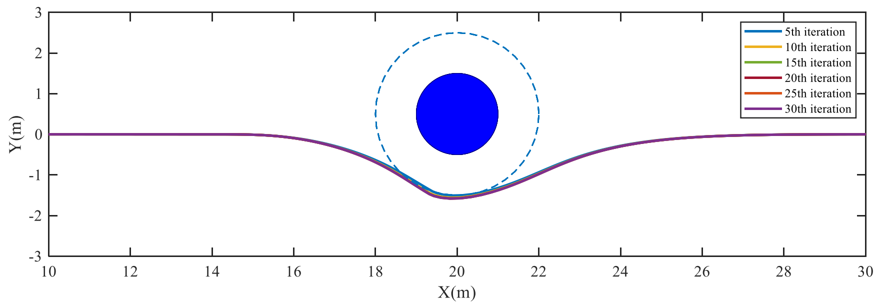

Fig. 5. *Iteration of robot's trajectories in a single-obstacle scenario based on P-DHP algorithm. The radii of the robot and obstacle are $r = 1$ m and $r_o = 1$ m, respectively.*

where r_d indicates the maximum distance that the robot can detect the obstacles, and the definitions of relevant parameters can be seen in Fig. 2. The parameters satisfy $\kappa_o < \bar{\kappa}_o$ and $r_o + r < r_d$. Similar to $\mathcal{B}^r_o(q)$ in (36), $\bar{\mathcal{B}}_o(q)$ is deflated into $\bar{\mathcal{B}}^r_o(q)$ by $\bar{\delta}_o$, $0 < \bar{\delta}_o < r_d$.

In the training process, the discounting factor was $\gamma = 0.95$. The penalty matrices were set as $Q_1 = I_5$, $R_1 = 0.01I_2$, $Q_2 = 0.001I_5$, $R_2 = 0.001I_2$, $\mu = 1$, $\mu_c = 0.001$, $\alpha_a = 0.2$, $\alpha_c = 0.4$, $\beta_a = 0.4$, $\beta_c = 0.4$, $\delta_o = 1.5$, $\bar{\delta}_o = 5$, $\kappa_o = 5.5$, $\bar{\kappa}_o = 600$, $r_d = 6$, and $\Delta h = 0.05$ s. In the learning process, the weights were initialized with uniformly distributed random values. The simulations were implemented in the Matlab 2019b environment on a laptop with an AMD Ryzen 7 4800H CPU.

Firstly, we tested our algorithm in a straight-line path scenario with one obstacle in the center of the road. The recorded performance costs were defined as $J^i_{r,1} = \sum_{k=1}^{K_1} J^i_1(k)$ and $J^i_{r,2} = \sum_{k=1}^{K_2} J^i_2(k)$ with $K_1 = 1000$ and $K_2 = 1000$ in each training iteration. The variation results of $J^i_{r,1}$ and $J^i_{r,2}$ in Fig. 4 show that both the performance costs decrease with iteration increases and converge in 30 iterations. Moreover, the *robot's* trajectories in the learning process and the actor weights of the robot in a single-obstacle scenario are displayed in Figs. 5 and 6. The results illustrate that the robot gradually moves away from the obstacle by continuous iterative learning until the robot's trajectory is circumscribed by the edge of the obstacle's potential field.

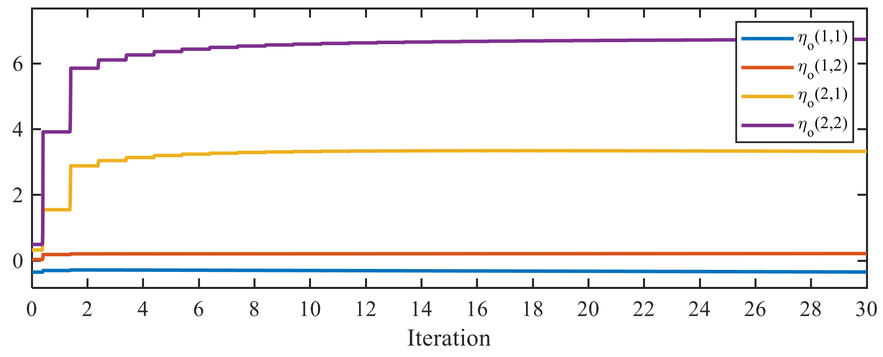

Fig. 6. *Actor-2 weights of robot with the increase of iteration, where $\eta_o(i,j)$ refers to the element in the i-th row and j-th column of matrix η_o, and $o = 1$.*

Table I. *Comparative results of P-DHP and CS-based RL in 100 tests.*

Methods	**Actor learning rate**	**Learning success rate (number of success/total number)**
Ours	$\beta_a = 0.05$	100% (100/100)
CS-based RL	$\alpha_a = 0.05$	100% (100/100)
Ours	$\beta_a = 0.1$	100% (100/100)
CS-based RL	$\alpha_a = 0.1$	98% (98/100)
Ours	$\beta_a = 0.2$	100% (100/100)
CS-based RL	$\alpha_a = 0.2$	95% (95/100)
Ours	$\beta_a = 0.4$	100% (100/100)
CS-based RL	$\alpha_a = 0.4$	90% (90/100)

4.1.2. Comparison results with state-of-the-art approaches

In this subsection, we compared our approach with the reward/cost shaping-based RL approach (CS-based RL) [27], the MPC using a disjunctive chance constraint (MPC-dc) [28], and the MPC using a linearized chance constraint (MPC-lc) [29]. Please refer to Fig. 9 for the adopted reference paths in the simulation tests.

In the CS-based RL approach, the cost function was constructed as $r = \|e(k)\|^2_{Q_1} + \|u(k) - u_r(k)\|^2_{R_1} + \mu\nabla\mathcal{B}_o(k)$, and the parameters of $\nabla\mathcal{B}_o$ were fine-tuned. For a fair comparison, the same DHP-based AC learning structure(actor-1-critic-1) was used in CS-based RL. The comparative results in 100 tests with different learning rates are illustrated in Table I and Fig. 7, which show that our approach outperforms the CS-based RL approach in the success rate of the learning process. Although the success rate between ours and CS-based RL is similar as the learning rate is set small, as shown in Fig. 8, the CS-based RL approach has a probability of failure to avoid obstacles due to the lack of corrective terms $\nabla\mathcal{B}_o$ in the control policy. As the learning rate increases, the learning failure rate of the CS-based RL approach also increases and has a longer obstacle avoidance distance with more energy loss, whereas our method possesses stable control performance for different learning rates. This illustrates, in part, that only shaping the cost function may not be sufficient to learn a convergent control policy efficiently via a standard AC learning algorithm. To solve this problem, our P-DHP approach are proposed to resolve the conflict between the path-following control and obstacle avoidance goals in the learning process.

The MPC-dc [28] and MPC-lc [29] algorithms were implemented by the CasADi toolbox with an Ipopt solver [30]. The stage costs in MPC-dc and MPC-lc were designed the same as (9) in our approach. The performance indicator J_e was defined as $J_e = 1/\bar{K}\sum_{k\in K}\|e(k)\|_2$, where K is the time interval in

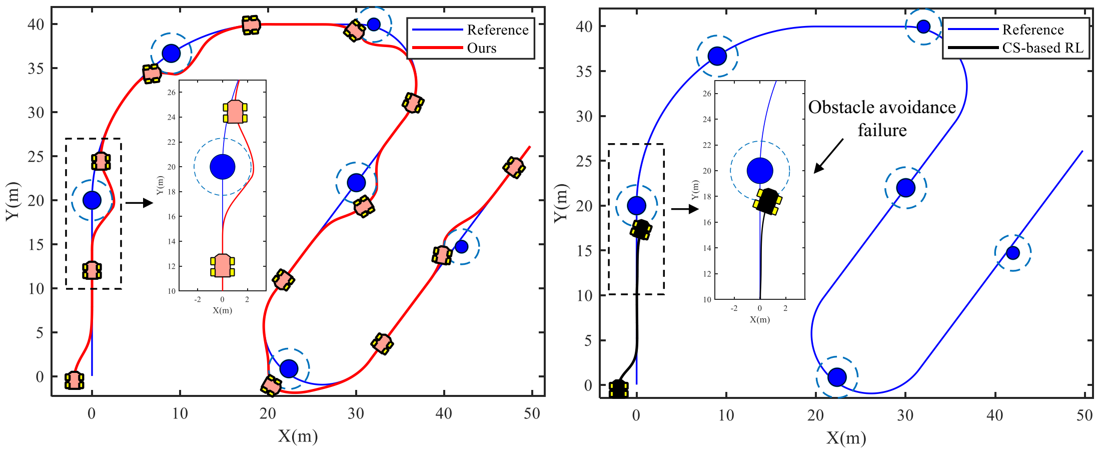

Fig. 7. Simulation results of our approach and the CS-based RL [27] with $v_r = 1$ m/s, $r = 1.3$ m, $r_o = 0.7$ m or $r_o = 1$ m. The CS-based RL method has a certain probability of learning failure, see Table I for details.

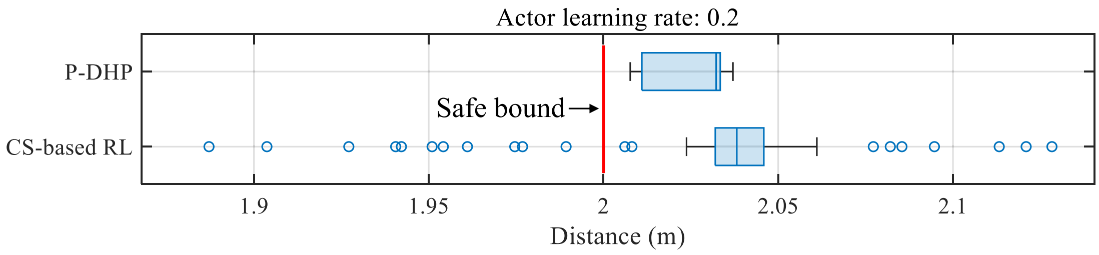

Fig. 8. Statistics on the closest distance from the robot to the obstacle in 100 repetitive tests. The comparisons were made between P-DHP and the CS-based RL [27] in the single-obstacle scenario (shown in Fig. 5). The safe bound is represented in red, and the outliers are drawn as blue circles.

obstacle avoidance, and $\bar{K}$ is the length of K. The comparative study was carried out under the scenario with three different sizes of obstacles, where the side lengths of the bounding boxes were set as $2.3/\sqrt{2}$ m, $2/\sqrt{2}$ m, and $2.6/\sqrt{2}$ m. The prediction horizon was set as $N = 30$. For the details of the simulation parameters, please refer to ref. [28].

The comparative simulation results are illustrated in Fig. 9 and Tables II and III. The results show that our approach outperforms the fixed-parameters MPC-lc and MPC-dc with smaller cost values and shorter path length in obstacle avoidance, demonstrating that our approach has better control performance at different reference velocities. This is due to the learning optimization mechanism and policy design of our algorithm. In addition, our approach has advantages in computational efficiency (see Table III).

4.2. Real-world experiments

We also tested our algorithm on the experimental platform of a real-world differential-drive wheeled robot. As shown in Fig. 10, the robot was controlled by a laptop equipped with the Ubuntu operating system, and the state of the robot and the positions of obstacles were obtained by mounted satellite inertial. In the experiment, we deployed the offline training weights from the simulation to generate control policy $u = [a_v, a_\omega]^\top$ to drive the underlying system of the robot, and the sampling interval was $\Delta h = 0.1$ s.

Table II. *Numerical comparison of P-DHP, MPC-dc and MPC-lc.*

Velocity (m/s)	**Methods**	**Path length (m)**	**J_e (in coll. avoid.)**
$v_r = 1$ m/s	Ours	72.739	1.091
	MPC-dc [28]	74.795	1.277
	MPC-lc [29]	76.766	1.543
$v_r = 1.3$ m/s	Ours	72.793	1.169
	MPC-dc	74.494	1.296
	MPC-lc	75.892	1.593
$v_r = 1.6$ m/s	Ours	73.026	1.311
	MPC-dc	75.823	1.506
	MPC-lc	76.133	1.793

Table III. *Computational time comparison ($v_r = 1$ m/s).*

	Ours				
Methods	**Path foll.**	**Coll. avoid.**	**Total**	**MPC-dc [28]**	**MPC-lc [29]**
One-step average computational time (ms)	1.22	1.59	2.81	191.2	272.9

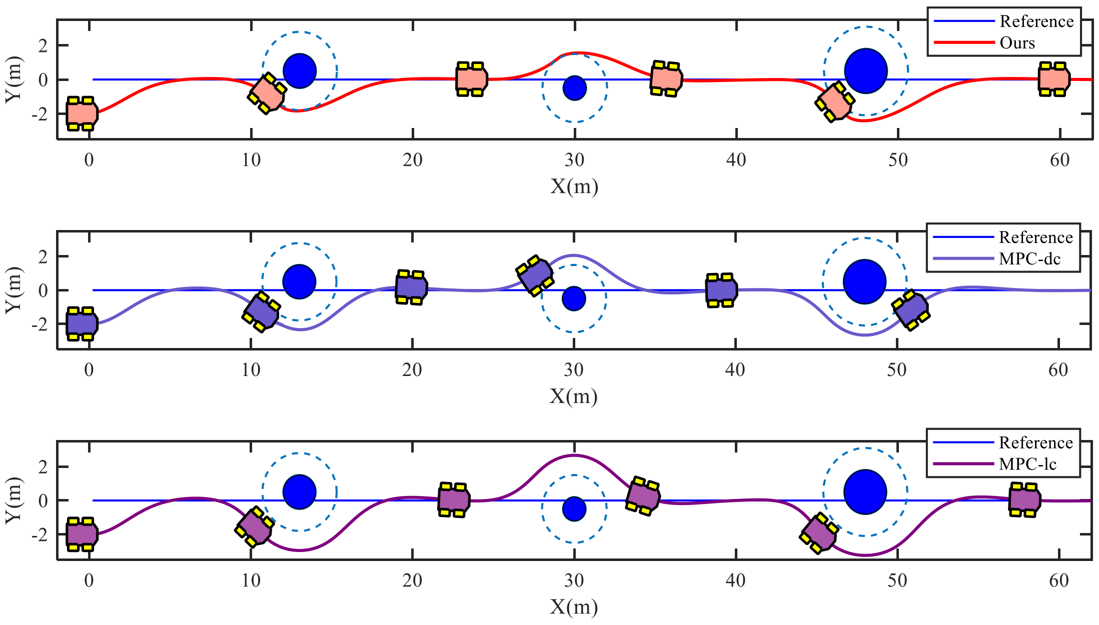

Fig. 9. *Simulation results of our approach, MPC-lc [29] and MPC-dc [28] with $v_r = 1.6$ m/s: the filled blue circular area represents the obstacles and the light blue dotted line indicates the influence range of the obstacle potential field. The simulation scene contains three sizes of obstacles, corresponding to $r_o = 0.7$ m, $r_o = 1$ m and $r_o = 1.3$ m respectively. The radius of the robot is $r = 1.3$ m.*

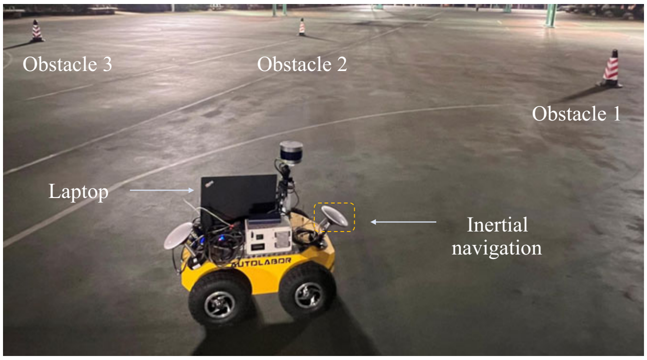

Fig. 10. *Experimental platform of the differential-drive wheeled robot.*

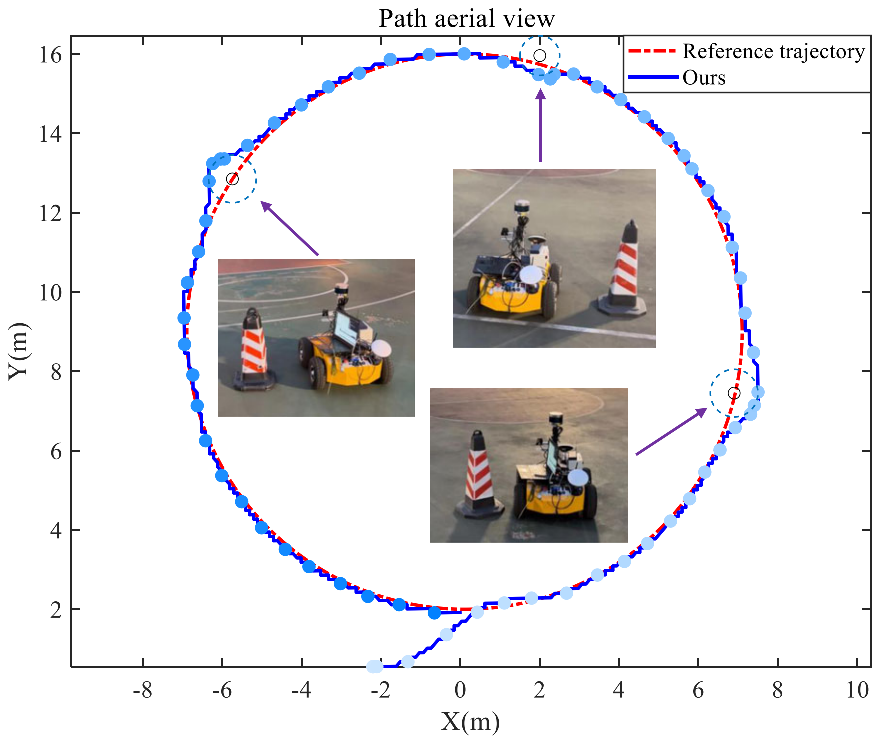

Fig. 11. *Experiment results of robot's path aerial view in a real-world scenario.*

The real-world experimental results are shown in Figs. 11 and 12, which illustrate that the robots can follow the desired path from an initial state, meanwhile avoiding all encountered obstacles on the path and recovering path following after collision avoidance. Moreover, the above-reported results show that our approach can resolve the weight divergence problem caused by the conflict between path following and obstacle avoidance.

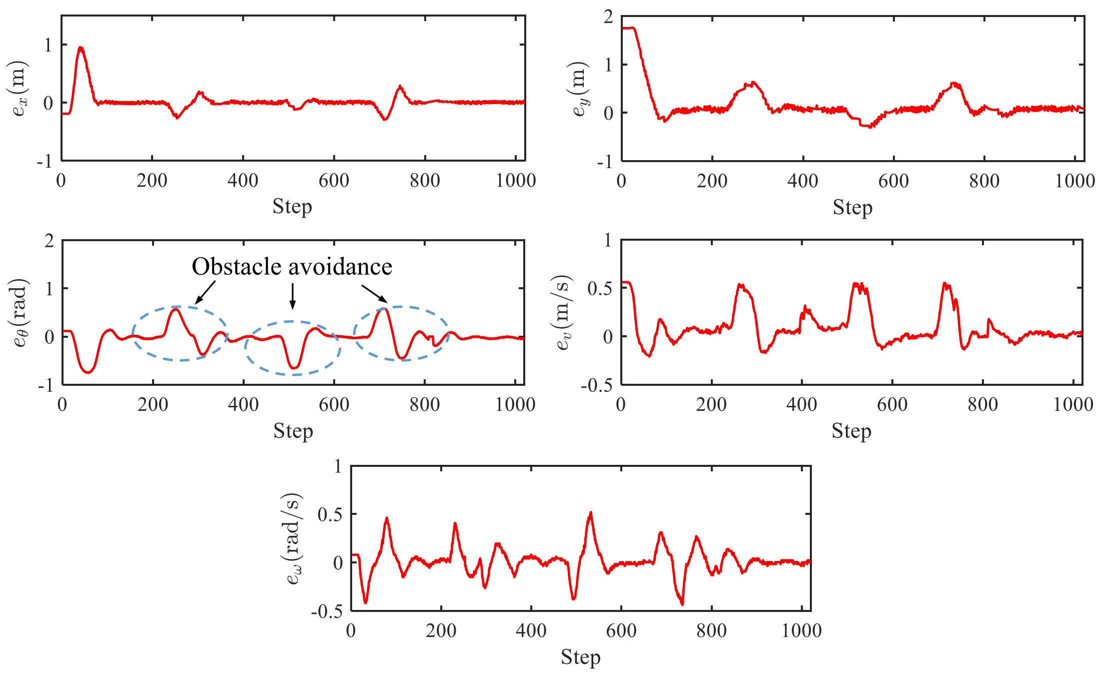

Fig. 12. *Experiment results of robot's state errors in real-world scenario.*

5. Conclusion

This paper proposed a P-DHP algorithm for path-following control of wheeled robots with obstacle avoidance. As for the main characteristics, the proposed P-DHP utilizes two coupled AC networks to resolve the conflicting goals, developing a near-optimal and safe control policy for path-following and obstacle avoidance. The convergence and safety of our method were proven. Our approach has been evaluated in simulation and tested on a real-world differential-drive mobile robot. The simulated and experimental results illustrate that our approach can realize path-following control with collision avoidance under conflicting learning objectives, showing the advantages in computational efficiency and smaller cost values compared with state-of-the-art approaches. Future work will focus on the application of model-free RL on wheeled mobile robots.

Author contributions. All authors have made great contributions to this paper.

Financial support. The work was supported by the National Natural Science Foundation of China under Grant U21A20518, 62003361, and 61825305, China Postdoctoral Science Foundation under Grant 47680, and the National Key R&D Program of China 2018YFB1305105.

Conflicts of interest. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] T. P. Nascimento, C. E. Dórea and L. M. G. Gonçalves, "Nonholonomic mobile robots' trajectory tracking model predictive control: A survey," *Robotica.* **36**(5), 676–696 (2018).

[2] Z. Li, K. Zhao, L. Zhang, X. Wu, T. Zhang, Q. Li, X. Li and C. Su, "Human-in-the-loop control of a wearable lower limb exoskeleton for stable dynamic walking," *IEEE/ASME Trans. Mechatron.* **26**(5), 2700–2711 (2021).

[3] H. Su, Y. Hu, H. R. Karimi, A. Knoll, G. Ferrigno and E. De Momi, "Improved recurrent neural network-based manipulator control with remote center of motion constraints: Experimental results," *Neural Netw.* **131**, 291–299 (2020).

[4] Z. Li, Z. Ren, K. Zhao, C. Deng and Y. Feng, "Human-cooperative control design of a walking exoskeleton for body weight support," *IEEE Trans. Ind. Inform.* **16**(5), 2985–2996 (2020).
[5] Z. Chen, Y. Liu, W. He, H. Qiao and H. Ji, "Adaptive-neural-network-based trajectory tracking control for a nonholonomic wheeled mobile robot with velocity constraints," *IEEE Trans. Ind. Electron.* **68**(6), 5057–5067 (2021).
[6] M. S. Wiig, K. Y. Pettersen and T. R. Krogstad, "Collision avoidance for underactuated marine vehicles using the constant avoidance angle algorithm," *IEEE Trans. Control. Syst. Technol.* **28**(3), 951–966 (2019).
[7] J. Alonso-Mora, P. Beardsley and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Trans. Robot.* **34**(2), 404–420 (2018).
[8] M. Hoy, A. S. Matveev and A. V. S. Al, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica.* **33**(3), 463–497 (2015).
[9] Khatib, Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.* **5**(1), 500–505 (1986).
[10] J. Funke, M. Brown, S. M. Erlien and J. C. Gerdes, "Collision avoidance and stabilization for autonomous vehicles in emergency scenarios," *IEEE Trans. Control. Syst. Technol.* **25**(4), 1204–1216 (2017).
[11] I. B. Hagen, D. K. M. Kufoalor, E. F. Brekke and M. T.A.Johansen, "MPC-Based Collision Avoidance Strategy for Existing Marine Vessel Guidance Systems," **In:** *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018) pp. 7618–7623.
[12] B. Brito, B. Floor, L. Ferranti and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robot. Autom. Lett.* **4**(4), 4459–4466 (2019).
[13] P. Leica, M. Herrera, C. Rosales, F. Roberti, J. Toibero and R. Carelli, "Dynamic Obstacle Avoidance Based on Time-Variation of a Potential Field for Robots Formations," **In:** *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)* (IEEE, 2017) pp. 1–6.
[14] P. Sudhakara, V. Ganapathy, B. Priyadharshini and K. Sundaran, "Obstacle avoidance and navigation planning of a wheeled mobile robot using amended artificial potential field method," *Proc. Comput. Sci.* **133**, 998–1004 (2018).
[15] B. Hu, Z.-H. Guan, F. L. Lewis and C. P. Chen, "Adaptive tracking control of cooperative robot manipulators with markovian switched couplings," *IEEE Trans. Ind. Electron.* **68**(3), 2427–2436 (2021).
[16] H. Dong, X. Zhao and H. Yang, "Reinforcement learning-based approximate optimal control for attitude reorientation under state constraints," *IEEE Trans. Control. Syst. Technol.* **29**(4), 1664–1673 (2020).
[17] Z. Zhang, S. Chen, X. Zhu and T. Z.Yan, "Two hybrid end-effector posture-maintaining and obstacle-limits avoidance schemes for redundant robot manipulators," *IEEE Trans. Ind. Inform.* **16**(2), 754–763 (2020).
[18] P. Deptula, H.-Y. Chen, R. A. Licitra, J. A. Rosenfeld and W. E. Dixon, "Approximate optimal motion planning to avoid unknown moving avoidance regions," *IEEE Trans. Robot.* **36**(2), 414–430 (2020).
[19] X. Zhang, Y. Peng, B. Luo, W. Pan, X. Xu and M. H.Xie, "Model-Based Safe Reinforcement Learning with Time-Varying State and Control Constraints: An Application to Intelligent Vehicles," *arXiv preprint*, arXiv:2112.11217 (2021).
[20] Y. Dong, X. Tang and Y. Yuan, "Principled reward shaping for reinforcement learning via Lyapunov stability theory," *Neurocomputing.* **393**, 83–90 (2020).
[21] M. Ohnishi, L. Wang, G. Notomista and M. Egerstedt, "Barrier-certified adaptive reinforcement learning with applications to brushbot navigation," *IEEE Trans. Robot.* **35**(5), 1186–1205 (2019).
[22] M. Wen and U. Topcu, "Constrained cross-entropy method for safe reinforcement learning," *IEEE Trans. Autom. Control* **66**(7), 3123–3137 (2021).
[23] M. H. Cohen and C. Belta, "Approximate Optimal Control for Safety-Critical Systems with Control Barrier Functions," **In:** *2020 59th IEEE Conference on Decision and Control (CDC)* (IEEE, 2020) pp. 2062–2067.
[24] Y. Y. Yang, W. He, K. G. Vamvoudakis, H. M. Modares and D. C. Wunsch, "Safety-Aware Reinforcement Learning Framework with an Actor-Critic-Barrier Structure," **In:** *2019 American Control Conference (ACC)* (IEEE, 2019) pp. 2352–2358.
[25] A. Ames, X. Xu, J. Grizzle and C. P.Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Robot. Automat. Lett.* **62**(8), 3861–3876 (2017).
[26] B. Luo, D. Liu, T. Huang and J. Liu, "Output tracking control based on adaptive dynamic programming with multistep policy evaluation," *IEEE Trans. Syst. Man Cybernet. Syst.* **49**(10), 2155–2165 (2019).
[27] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *J. Mach. Learn. Res.* **3**(12), 803–832 (2002).
[28] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez and H. Voos, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *IEEE Robot. Automat. Lett.* **5**(2), 3620–3625 (2020).
[29] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robot. Automat. Lett.* **4**(2), 776–783 (2019).
[30] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.* **1**(1), 1–36 (2019).

Cite this article: Y. Peng, X. Zhang, H. Xie and X. Xu (2023). "Potential field-based dual heuristic programming for path-following and obstacle avoidance of wheeled mobile robots", Robotica **41**, 1931–1946. https://doi.org/10.1017/S0263574723000127