Dynamic path planning of mobile robots using adaptive dynamic programming

Xin Li [a,b], Lei Wang [a,c], Yi An [d,e,b,*], Qi-Li Huang [f], Yun-Hao Cui [g], Huo-Sheng Hu [h]

[a] School of Mathematical Sciences, Dalian University of Technology, Dalian 116023, China
[b] Key Laboratory of Intelligent Control and Optimization for Industrial Equipment of Ministry of Education, Dalian University of Technology, Dalian 116023, China
[c] Key Laboratory of Computational Mathematics and Data Intelligence of Liaoning Province, Dalian University of Technology, Dalian 116023, China
[d] School of Control Science and Engineering, Dalian University of Technology, Dalian 116023, China
[e] School of Electrical Engineering, Xinjiang University, Urumqi 830046, China
[f] Zhongxing Telecom Equipment, Shenzhen 518000, China
[g] School of Mechanical Engineering, Dalian University of Technology, Dalian 116023, China
[h] School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK

A B S T R A C T

Dynamic path planning has gained increasing popularity in mobile robot navigation. Some of the current path planning methods require a priori information about the motion space and are easily affected by the distribution of obstacles. To address the above limitation, this paper presents a novel dynamic method that transforms path planning into an optimal control problem and solves it dynamically through adaptive dynamic programming and artificial potential field. The proposed method can obtain optimal paths for a differentially-driven mobile robot model in an unknown environment with many irregular obstacles. First, by combining path optimization and kinematical constraints of the mobile robot, the original problem is transformed into a new problem. Second, the total distance traveled, the effect of heading angle, the distance from the target to the robot, and the resultant force of the artificial potential field are included in the new performance index function. Third, the method based on adaptive dynamic programming is developed to avoid obstacles and guarantee the safety of autonomous navigation. The convergence analysis provides theoretical guarantees for our method, and the iterative control sequence will converge to the optimal control. Furthermore, simulation results and analyses under different complexity levels demonstrate that our method has promising performance in exploring and exploiting dynamic path planning problems.

Keywords:
Path planning
Differentially-driven mobile robot
Optimal control
Adaptive dynamic programming
Artificial potential field

1. Introduction

With rapid advances in artificial intelligence and control engineering, mobile robots are beginning to have a major impact in both the military and civilian sectors. For example, self-driving cars do not need drivers but only passengers, unmanned aerial vehicles (UAVs) can replace humans in delivering goods, unmanned surface vehicles can perform precise autonomous docking, and smart wheelchairs can improve the quality of life for disabled people. (Liu et al., 2023; Li et al., 2023; Wang et al., 2023; Low et al., 2022). Autonomous navigation is a crucial technical issue that needs further research for mobile robots. Autonomous navigation can be broadly classified into four categories: environment sensing, path planning, motion control, and decision making (Ntakolia et al., 2023; Wang et al., 2020; Pietrzykowski et al., 2022). Path planning is an essential bridge between environment sensing and motion control, and it is a key part of the mobile robot (Dian et al., 2022; González et al., 2016). The path planning of the mobile robot is to find a feasible route in the motion space from the initial location to the given target location, and avoid some irregular obstacles such as rocks, trees, and cars. Advanced path planning methods can significantly improve motion performance and reduce wear and tear of mobile robots.

Generally, traditional path planning methods for mobile robots are divided into two types, depending on the environmental space available to the mobile robot: static and dynamic path planning (Jones et al., 2023). The static methods generate feasible paths in an off-line manner, which can obtain complete environmental information and the mobile robot can arrive at the target location by following the "first planning then tracking" framework, such as curve fitting, grid method,

intelligent optimization algorithm, etc. (Sathiya et al., 2022; Wahab et al., 2020; Sedighi et al., 2019; Ammar et al., 2016). Static path planning methods have served the scientific and industrial communities for quite some time. Chen et al. (2022b) proposed an interval path planning method for patrol robots in risk areas, which improved the multi-objective particle swarm optimization to search for safe paths with objective functions of both optimal distance and riskiness. Li et al. (2023) developed an off-line trajectory planner with high-quality warm-started strategy for the fixed-wing UAV formation. Kyaw et al. (2022) designed a novel batch-informed trees* method with the energy-based objectives for reconfigurable robots in complex environments. However, the planning process of the static method relies on complete environmental information, the data collection is space consuming and expensive, and the mobile robot needs to construct a global map model before planning. Once unknown obstacles or moving obstacles appear in the motion space, it may be questionable whether the mobile robot can safely track the planned path to the target. These static path planning methods are limited in their application because they are unable to obtain a viable motion path for the mobile robot in real-time.

Compared with the static methods, the dynamic path planning methods generate paths in an on-line manner, which obtain the partial environment information by sensors in real-time, such as the artificial potential field (APF), dynamic window approach (DWA) and reinforcement learning (Fan et al., 2023; Fox et al., 1997; Kiran et al., 2022). Dynamic path planning has gained popularity in recent years. Wang et al. (2022) proposed a simultaneous planning and control framework to solve the real-time planning problem for unmanned surface vehicles in unknown motion spaces. Sangiovanni et al. (2021) presented a hybrid control method to obtain a collision-free path in anthropomorphic robots, which could be deployed in real-time, requiring only the sensor data of the robot and the surroundings. Chen et al. (2022a) developed a reinforcement learning-based path planning method with dynamic obstacle avoidance, which can avoid moving obstacles in the environment and achieve real-time planning. These methods significantly reduce the cost of resources and improve the dynamic obstacle avoidance capability. The dynamic methods guide the mobile robots to dynamically obtain the solutions in the sampling range and adjust the path on-line to achieve obstacle avoidance (Wu et al., 2021). As the study progresses, the above dynamic path planning methods encounter many challenges. Real-time capability is a major problem for such methods. Dynamic methods can only be effective if the computing time is less than the sampling interval, so the computational burden cannot be too high. In addition, many dynamic methods tend to get stuck in a situation where they fail to converge. The APF uses force fields to guide the mobile robot to complete the path, but this method often falls into a local optimum in obstacle-dense spaces and ultimately fails to reach its target (Rosas et al., 2019). The DWA realizes a path planning scheme by rolling computation of the window containing local information, but it suffers from the infinite loop problem (Kowsar et al., 2022). The reinforcement learning methods have excellent performance, but designing an appropriate reward function can be difficult and the training process requires significant computational power (Ladosz et al., 2022). More importantly, most of the above static and dynamic path planning methods consider the mobile robot as a mass and the mobile robot can move unconstrained. However, mobile robots are essentially nonlinear systems with complicated motion constraints. For example, differentially-driven models must adjust the direction by the difference in velocity between two wheels. And fixed-wing UAVs cannot remain stationary in the sky. Planning the path of a mass may be impractical for the autonomous navigation of mobile robots in some scenes.

In recent years, some researchers have been working on optimal control based dynamic path planning methods. In order to obtain a better solution, the researchers consider the motion equation of the mobile robot when designing the path planner. Optimal control methods are the intersection of mathematical optimization and control theory, which calculates the optimal performance index function under dynamic constraints (Teng et al., 2022; Wang et al., 2017). The original problems are formulated as optimal control problems (OCPs), which can be solved by pseudospectral or other optimal control methods. Hansen and Wang (2020) solved the path accuracy problem for autonomous parking systems by applying Pontryagin's maximum principle. Zhang et al. (2018) modeled the path problem of a mobile robot as an OCP and designed an approximation strategy to deal with the non-convex part. Ji et al. (2017) proposed a planning and tracking algorithm through the virtual potential field and the model predictive control for intelligent vehicles, which can minimize collision occurrence. When there are many mobile robots or obstacles in the motion space, the traditional optimal control method will encounter the "curse of dimensionality". This is due to the nonlinear motion equation constraint and other state constraints of mobile robots, which causes great inconvenience to the solution. Based on the above analysis, it would be likely to facilitate the dynamic path planning for mobile robots if an optimal control method with a low computational burden could be developed.

To investigate how planners generate collision-free solutions in real-time, this paper builds a new dynamic path planning method for mobile robots in complex unknown environments. The method has advantages in on-line planning, high solving efficiency, and can obtain the optimal path. With the proposed method, we attempt to solve the "curse of dimensionality", and improve the computational efficiency in OCPs. The main contributions of our study are

- This paper developed an ADP-based method to solve the dynamic path planning problem, the original problem is transformed into an OCP.
- Our method designs a new performance index function, which can optimize the path length of the mobile robot, and avoid obstacles appearing in the unknown environment.
- The convergence analysis proves that the proposed method can theoretically find the optimal solution of the OCP under appropriate constraints.
- The simulation results based on dynamic and static obstacles demonstrate that the proposed method is effective for the path planning of the mobile robot, and that robustness and real-time capability can be guaranteed.

The rest of this paper is organized as follows. Section 2 formulates the path planning problem and gives the mathematical description. In Section 3, we present the specific implementation of the proposed method based on the optimal control. In Section 4, we prove the convergence of our method. Simulation results and related analysis are presented in Section 5. At last, Section 6 concludes this paper and gives future prospects.

2. Problem formulation

The original path planning problem will be converted into an OCP. We will give a mathematical description of the problem in this section.

2.1. Motion equation

There are many different types of mobile robots in the world, such as differentially-driven mobile robots, legged robots, car-like models, UAVs and so on. And they have different driving principles and kinematical equations to make mobile robots move. Differentially-driven mobile robots are a typical class of mobile robot models whose motion comes from their driven wheels placed on both sides. The configuration of a differentially-driven model is shown in Fig. 1. This mobile robot model has three wheels, of which wheels A and B are driven wheels, and wheel C is the passive wheel.

Differentially-driven mobile robots, such as smart wheelchairs and sweeping robots, often operate in complex unknown environments,

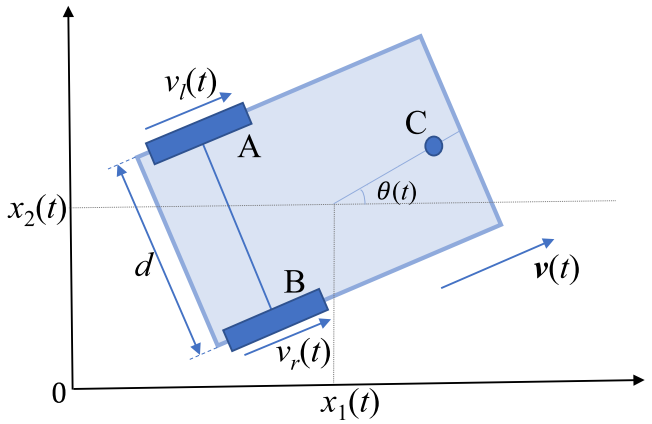

Fig. 1. The configuration of a differentially-driven mobile robot.

dynamic path planning methods with real-time obstacle avoidance performance are very important for such mobile robots. Therefore, this paper considers a differentially-driven model as the controlled plant and is devoted to developing the dynamic methods to successfully drive the model to the target while satisfying maneuverability. The kinematical equation of a differentially-driven mobile robot (Almomani et al., 2021; Wang et al., 2015) is written as

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \begin{pmatrix} v(t)\cos(x_3(t)) \\ v(t)\sin(x_3(t)) \\ \omega(t) \end{pmatrix} \tag{1}$$

where the state vector $\boldsymbol{x}(t) = [x_1(t), x_2(t), x_3(t)] \in \mathbb{R}^3$, $(x_1(t), x_2(t))$ are the position coordinates in the 2D environment, and $x_3(t) = \theta(t)$ is the heading angle. The control vector $\boldsymbol{u}(t) = [u_1(t), u_2(t)] \in \mathbb{R}^2$, $u_1(t) = v_r(t)$ and $u_2(t) = v_l(t)$ presents the linear velocity of the right and left driven wheel, respectively. For the model, d indicates the distance between the two driven wheels, the linear velocity is $v(t) = (u_1(t) + u_2(t))/2$, and the angular velocity is $\omega(t) = (u_1(t) - u_2(t))/d$.

To solve the path planning problem using the numerical method, the kinematical equation in continuous-time can be transformed into a discrete-time equation as follows

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{pmatrix} = \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{pmatrix} + \Delta t \begin{pmatrix} 0.5 * (u_1(k) + u_2(k))\cos(x_3(k)) \\ 0.5 * (u_1(k) + u_2(k))\sin(x_3(k)) \\ (u_1(k) - u_2(k))/d \end{pmatrix} \tag{2}$$

where the variables $\boldsymbol{x}(k) = [x_1(k), x_2(k), x_3(k)]$ and $\boldsymbol{u}(k) = [u_1(k), u_2(k)]$ are introduced to present $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$, respectively. The constant Δt is the uniform sampling interval in our study.

For the mobile robot, the kinematical Eq. (2) can be simplified to a compact form as follows

$$\boldsymbol{x}(k+1) = F(\boldsymbol{x}(k), \boldsymbol{u}(k)), k = 0, 1, 2, \ldots, \infty \tag{3}$$

where $k = \{0, 1, 2, \ldots, \infty\}$ is defined as the sampling point in this work, $F(\cdot) : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}^3$ is the mapping of the system function. The moving of the mobile robot must strictly obey this kinematical equation, and we assume in this paper that the robot system is controlled.

2.2. Physical constraints

For path planning problems, the mobile robot will obtain the corresponding sampling state values $\boldsymbol{x}(k)$ at each time step k, and the initial state $\boldsymbol{x}_0$ and the terminal state $\boldsymbol{x}_f$ of the mobile robot should be satisfied. Both the state $\boldsymbol{x}(k)$ and the control $\boldsymbol{u}(k)$ are constrained due to practical physical properties of mobile robots, leading to the following inequality constraints

$$\boldsymbol{x}_{\min} \le \boldsymbol{x}(k) \le \boldsymbol{x}_{\max} \tag{4}$$

$$\boldsymbol{u}_{\min} \le \boldsymbol{u}(k) \le \boldsymbol{u}_{\max} \tag{5}$$

where $\boldsymbol{x}_{\min}$ and $\boldsymbol{u}_{\min}$ are the minimum values of the state and control, and $\boldsymbol{x}_{\max}$ and $\boldsymbol{u}_{\max}$ are the maximum values of the state and control, respectively.

In addition, the distance between the mobile robot and these obstacles should be kept at a safe distance in case of collisions. In a dynamic path planning problem, the mobile robot moves in an unknown environment without a priori knowledge and only calculates the distance to obstacles within its perception range. Thus, the obstacle constraint is expressed as

$$d_m(k) \ge \beta \tag{6}$$

where $m = 1, 2, \ldots, M_k$ presents that there are M_k obstacles $\{O_1, O_2, \ldots, O_{M_k}\}$ within the perception range. β is a predefined safety distance between the mobile robot and O_m; $d_m(k)$ is the Euclidean distance between the mobile robot and the outer surface of O_m.

2.3. Performance index function

For OCPs, the performance index function $J(\cdot)$ (Liu et al., 2021; Mu et al., 2017) under the control law $\{\boldsymbol{u}(k)\}$ can be described as follows

$$J(\boldsymbol{x}(0)) = \sum_{k=0}^{\infty} U(\boldsymbol{x}(k), \boldsymbol{u}(k)) = U(\boldsymbol{x}(0), \boldsymbol{u}(0)) + J(\boldsymbol{x}(1)) \tag{7}$$

where the utility function $U(\boldsymbol{x}(k), \boldsymbol{u}(k)) > 0$ represents the stage value based on $\boldsymbol{u}(k)$, and transfers the system equation from $\boldsymbol{x}(k)$ to $\boldsymbol{x}(k+1)$.

It is necessary to design an appropriate performance index function for our problem, which can represent the expectation of minimizing energy consumption, terminal time, distance traveled by mobile robots, or other factors to minimize the mission risk. In general, the shortest path is advantageous to improve the performance of a mobile robot. From the path-optimal perspective, we expect to solve the problem in this study as follows

$$\min J_1 = \min \sum_{k=0}^{\infty} \Delta S^2(k) = \min \sum_{k=0}^{\infty} \sum_{i=1}^{2} (x_i(k+1) - x_i(k))^2 \tag{8}$$

where the notation $\sum \Delta S^2(k)$ is the square of the total distance traveled, $\Delta S(k) = \|\boldsymbol{s}(k+1) - \boldsymbol{s}(k)\|_2$ is the movement distance from the time step k to $k+1$, and $\boldsymbol{s}(k) = [x_1(k), x_2(k)]$ is the position coordinate of the differentially-driven model.

2.4. Formulation

By combining (1) to (8), the dynamic path planning problem of the differentially-driven mobile robot is established as an OCP with a number of constraints

$$\begin{aligned} \min J_1 &= \min \sum_{k=0}^{\infty} \Delta S^2(k) \\ \text{s.t.} \quad & \boldsymbol{x}(k+1) = F(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\ & G(\boldsymbol{x}(k), \boldsymbol{u}(k), k) \le 0 \\ & k = 0, 1, 2, \ldots \end{aligned} \tag{9}$$

where, for simplicity, the inequality $G(\boldsymbol{x}(k), \boldsymbol{u}(k), k) \le 0$ denotes the set of all physical constraints. The OCP is subject to the kinematical equation and these constraints on state variables, control variables and time.

3. Proposed path planning method

Adaptive dynamic programming (ADP) and APF methods are used to obtain the solution of dynamic path planning problems. The advantages of our method are to obtain dynamic paths that avoid all obstacles in unknown environments.

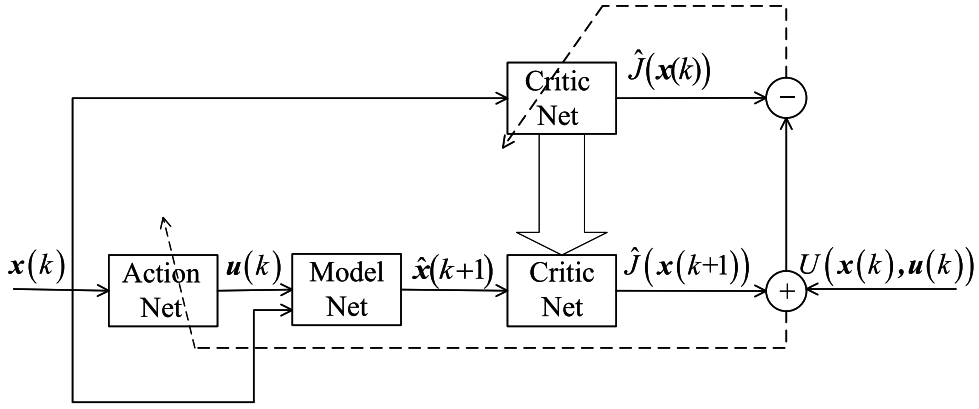

Fig. 2. The structure of ADP.

3.1. Iterative optimal control method

This paper formulates an OCP to solve the dynamic path planning problem. The final solution is to achieve an admissible control strategy that stabilizes the system and minimizes the function $J(\boldsymbol{x}(k))$ under constraint conditions. Namely, we want the planning scheme to obtain a collision-avoidance path in complex unknown environments with many irregular obstacles. A new performance index function is designed to obtain the optimal or quasi-optimal path and avoid obstacles. The optimal value $J^*(\boldsymbol{x}(k))$ derived from (7) (Wei et al., 2022) is defined as

$$\begin{aligned} J^*(\boldsymbol{x}(k)) &= \min_{\boldsymbol{u}(k)}\{J(\boldsymbol{x}(k),\boldsymbol{u}(k)) : \boldsymbol{u}(k) \in \mathcal{U}\} \\ &= U(\boldsymbol{x}(k),\boldsymbol{u}^*(k)) + J^*(\boldsymbol{x}(k+1)) \end{aligned} \tag{10}$$

where $\mathcal{U}$ is the admissible control set.

The optimal control $\boldsymbol{u}^*(k)$ is

$$\boldsymbol{u}^*(\boldsymbol{x}(k)) = \arg\min_{\boldsymbol{u}(k)}\{U(\boldsymbol{x}(k),\boldsymbol{u}(k))\} + J^*(\boldsymbol{x}(k+1)) \tag{11}$$

Commonly, it is difficult to know $J^*(\boldsymbol{x}(k))$ before the sequence $\{\boldsymbol{u}^*(k)\}$ is determined. If optimal control methods have to calculate $J^*(\boldsymbol{x}(k))$ at every time step k, this will impose a huge computational burden on the processing units. This phenomenon makes it difficult to solve the Hamilton–Jacobi–Bellman (HJB) equation directly.

To overcome these challenges, ADP (Wang et al., 2009; Werbos, 1977) is developed to use a function to approximate the actual structure of the system function. The ADP method is derived from Bellman's principle of optimality and is a kind of novel iterative optimal control method to approximate functions. It combines the ideas of dynamic programming, neural networks, and reinforcement learning (Liu et al., 2021). Neural networks and some basis functions can be used as approximation functions to replace some parts of the ADP. Since its proposal, the theoretical research of ADP has received much attention, and ADP will have a bright future in solving dynamic path planning problems.

The structure of ADP is shown in Fig. 2. In general, the ADP consists of three parts: model net, action net, and critic net. In this paper, we use neural networks to approximate these modules. The model net approximates the system equation and predicts the next state. The input is $\boldsymbol{x}(k)$ and $\boldsymbol{u}(k)$ at the time step k, and $\hat{\boldsymbol{x}}(k+1)$ is an approximation of the actual state $\boldsymbol{x}(k+1)$. The training process is realized by minimizing the following error function

$$\|\boldsymbol{E}_m(k+1)\| = \frac{1}{2}\sum_{i=1}^{n}[x_i(k+1) - \hat{x}_i(k+1)]^2 \tag{12}$$

where $\|\boldsymbol{E}_m(k+1)\|$ is the sum of the squared error vector, n is the dimension of the state vector $\boldsymbol{x}(k+1)$.

The critic net takes the state $\boldsymbol{x}(k)$ as the input value, and the corresponding output value $\hat{J}(\cdot)$ is the approximation of $J(\cdot)$ given in (7), realized as follows

$$\|E_c(k)\| = \frac{1}{2}\{\hat{J}(\boldsymbol{x}(k)) - [U(\boldsymbol{x}(k),\boldsymbol{u}(k)) + \hat{J}(\boldsymbol{x}(k+1))]\}^2 \tag{13}$$

where $\|E_c(k)\|$ is the sum of the squared errors. If $\|E_c(k)\| = 0$ for all time steps, then we have $\hat{J}(\boldsymbol{x}(k)) = J(\boldsymbol{x}(k))$. Therefore, a trained critic net could be obtained by optimizing the function defined in (13).

The action net is trained by minimizing the output $\hat{J}(k)$ of the critic net to generate the optimal or suboptimal control variable. The output depends on the utility function associated with the controlled system. The common form of the $U(\boldsymbol{x}(k),\boldsymbol{u}(k))$ is a quadratic form.

$$U(\boldsymbol{x}(k),\boldsymbol{u}(k)) = \boldsymbol{x}(k)\boldsymbol{A}\boldsymbol{x}^T(k) + \boldsymbol{u}(k)\boldsymbol{B}\boldsymbol{u}^T(k) \tag{14}$$

where matrix $\boldsymbol{A}$ is an n-dimensional positive semidefinite matrix, and matrix $\boldsymbol{B}$ is an m-dimensional positive-definite matrix.

3.2. Artificial potential field method

The APF method is derived from potential field methods (Khatib, 1990) and has been widely used to solve path planning problems because of its low complexity, good real-time performance, and easy controlling operation. The core idea of the classical APF is to consider the mobile robot as a free mass, and the mass is affected by the resultant force of potential fields. It is assumed that the robot can be guided to reach the newly generated position and keep it away from obstacles at the next time step. However, the motion of a differentially-driven model is subject to its kinematical equation rather than a free mass. In practice, free motion could pose a risk to the internal mechanics of the differentially-driven mobile robot. Therefore, we construct the novel potential field for the dynamic path planning problem.

The attractive potential field $U_{att}(k)$ in this study is

$$U_{att}(k) = \frac{1}{2}K_{att} * \rho_1^2(k) \tag{15}$$

where the parameter K_{att} is the attractive potential coefficient, $\rho_1(k) = \|\boldsymbol{s}(k) - \boldsymbol{s}_f\|$ is the distance from $\boldsymbol{s}(k)$ to the desired target position $\boldsymbol{s}_f$, and we have no requirements for the terminal heading angle. The mobile robot is always affected by the field $U_{att}(k)$ until it reaches the target $\boldsymbol{s}_f$.

The attractive force $F_{att}(k)$ is the negative gradient of $U_{att}(k)$, when the mobile robot reaches the desired point, then $F_{att}(k) = 0$.

$$F_{att}(k) = -\nabla U_{att}(k) = -K_{att} * \rho_1(k) \tag{16}$$

The challenge for mobile robots is to move in unknown environments without a priori knowledge. When calculating the repulsive potential field, only obstacles within the mobile robot's perception range can be counted. And classical APF tends to treat the obstacle

as a particle, using a point to represent the whole obstacle. However, the obstacles encountered in practice are irregular, and sometimes the mobile robot can only perceive part of the obstacle.

In our method, we calculate the repulsive potential field with the outer surface of the obstacle. Once an obstacle is detected, its outer surface is discretized into many points. Then, the repulsive potential field $U^j_{rep}(k)$ of the mobile robot and the outer surface of the obstacle is written as

$$U^j_{rep}(k) = \begin{cases} \frac{1}{2}K_{rep}[\frac{1}{\rho^j_2(k)} - \frac{1}{p}]^2\rho^{\iota}_1(k), & \rho^j_2(k) < p \\ 0, & \rho^j_2(k) \geq p \end{cases} \tag{17}$$

where the parameter K_{rep} is the repulsive potential coefficient, $\rho^j_2(k) = \|\boldsymbol{s}(k) - \boldsymbol{s}^j_{obs}\|$ is the length from the $\boldsymbol{s}(k)$ to the position $\boldsymbol{s}^j_{obs}$ of the jth point on the outer surface of obstacle-m, $j = \{1, 2, \ldots, J_m\}$ denotes the outer surface of obstacle-m is discretized into J_m points, p is the perception range of the mobile robot, ι is the repulsive decay factor of the target. If $\rho^j_2(k) < p$, the mobile robot will be affected by the repulsive potential field $U^j_{rep}(k)$. Otherwise the field $U^j_{rep}(k) = 0$.

The repulsive force $F^j_{rep}(k)$ is the negative gradient of the potential field $U^j_{rep}(k)$. $F^j_{rep}(k)$ tends to infinity as the mobile robot approaches the jth point on the outer surface of the obstacle-m.

$$\begin{aligned} F^j_{rep}(k) &= -\nabla U^j_{rep}(k) \\ &= \begin{cases} K_{rep}[\frac{1}{\rho^j_2(k)} - \frac{\rho^2_1(k)}{p}]^2\frac{\rho^{\iota}_1(k)}{(\rho^j_2)^2(k)}, & \rho^j_2(k) < p \\ 0, & \rho^j_2(k) \geq p \end{cases} \end{aligned} \tag{18}$$

The total repulsive force $F_{rep}(k)$ is the vectorial sum of repulsive forces acting on the mobile robot, as follows

$$F_{rep}(k) = \sum_{j=1}^{a} F^j_{rep}(k) \tag{19}$$

where $a = \sum_{m=1}^{M_k} J_m$ is the number of points on the outer surface of obstacles within the perception range p at the time step k.

The APF assumes that the mobile robot moves in an abstract virtual potential field. It can reflect the structure of the motion space. Before planning, the sensors provide the mobile robot with environmental information about the distribution of obstacles and the target point. The area of the attractive potential field is the whole motion space. The area of the repulsive potential field acts in an area close to obstacles. The resultant force of the APF is expressed as

$$F_{apf}(k) = F_{att}(k) + F_{rep}(k) \tag{20}$$

To make the path generated by our method match the actual motion situation, we integrate the resultant force as part of the OCP (9). The motion is then restained by the kinematic Eq. (2).

3.3. Construction of the performance index function

This paper develops a novel iterative optimal control method to solve dynamic path planning problems, which incorporates many advantages of ADP and APF. In order to obtain the optimal path and avoid obstacles, we have designed a new performance index function $J_2(\boldsymbol{x}(k))$ to replace J_1 in the OCP (9). The proposed function $J_2(\boldsymbol{x}(k))$ includes the total distance traveled $J_{path}(k)$, the change in heading angle $J_{angle}(k)$, the distance $J_{goal}(k)$ between the target and the robot, and the resultant force of the APF method $J_{apf}(k)$.

To ensure optimality, we take into account distance traveled $J_{path}(k) = \sum_{i=k}^{\infty} j^2_{path}(i)$, where $j_{path}(i) = \Delta S(i) = \|\boldsymbol{s}(i+1) - \boldsymbol{s}(i)\|_2$ is the distance traveled in adjacent time steps, $\boldsymbol{s}(i)$ is the position of the mobile robot at time step i, and the terminal variable $\boldsymbol{s}(\infty)$ is the final position of the mobile robot. And we want the change in heading angle $x_3(k)$ between adjacent time steps to be as small as possible, i.e. $\min \|x_3(k+1) - x_3(k)\|_2$, so that we can make the directional changes of a differentially-driven mobile robot more realistic. We therefore define the function $J_{angle} = \sum_{i=k}^{\infty} \|x_3(k+1) - x_3(k)\|^2_2$ as part of the designed function $J_2(\boldsymbol{x}(k))$.

To reach the desired target position, one of the fundamental indices to consider is the distance between the desired position and our mobile robot. The index is written as $J_{goal}(k) = \sum_{i=k}^{\infty}(\boldsymbol{s}(i) - \boldsymbol{s}_f)^T\boldsymbol{R}(\boldsymbol{s}(i) - \boldsymbol{s}_f)$, where $\boldsymbol{s}_t$ is the desired position, and $\boldsymbol{R}$ is a positive-definite matrix. And if $\|\boldsymbol{s}(k) - \boldsymbol{s}_f\| \leq \chi$, it means that the mobile robot has reached the desired target position.

To realize the function of dynamic obstacle avoidance in the unknown environment, this paper makes the resultant force $J_{apf}(k)$ of the APF method part of the function $J_2(\boldsymbol{x}(k))$.

Correspondingly, the total function $J_2(\boldsymbol{x}(k))$ of the OCP is defined in this form

$$\begin{aligned} J_2(\boldsymbol{x}(k)) &= J_{path}(k) + J_{angle}(k) + J_{goal}(k) + J_{apf}(k) \\ &= \sum_{i=k}^{\infty}\left\{\begin{array}{l} \sigma_1(\|\boldsymbol{s}(i+1) - \boldsymbol{s}(i)\|^2_2 \\ +\|x_3(i+1) - x_3(i)\|^2_2) \\ +(\boldsymbol{s}(i) - \boldsymbol{s}_f)^T\boldsymbol{R}(\boldsymbol{s}(i) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(i) + F_{rep}(i)) \end{array}\right\} \\ &= \sum_{i=k}^{\infty}\left\{\begin{array}{l} \sigma_1(\sum_{j=1}^{3}\|x_j(i+1) - x_j(i)\|^2_2) \\ +(\boldsymbol{s}(i) - \boldsymbol{s}_f)^T\boldsymbol{R}(\boldsymbol{s}(i) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(i) + F_{rep}(i)) \end{array}\right\} \\ &= \sum_{i=k}^{\infty}\left\{\begin{array}{l} \sigma_1(\|\boldsymbol{x}(i+1) - \boldsymbol{x}(i)\|^2_2) \\ +(\boldsymbol{s}(i) - \boldsymbol{s}_f)^T\boldsymbol{R}(\boldsymbol{s}(i) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(i) + F_{rep}(i)) \end{array}\right\} \end{aligned} \tag{21}$$

where σ_1 and σ_2 are the weighted factors, $\boldsymbol{R}$ is the weight matrix.

3.4. Dynamic path planning method

We expect the mobile robot to move from the initial point to the desired terminal point, avoiding all obstacles in the dynamic environment without a priori information. The mobile robot is constrained by the kinematical equation and many complex constraints. According to our designed performance index function, the OCP (9) is rewritten as follows.

$$\begin{aligned} \min \sum_{i=k}^{\infty}&\left(\begin{array}{l} \sigma_1(\|\boldsymbol{x}(i+1) - \boldsymbol{x}(i)\|^2_2) \\ +(\boldsymbol{s}(i) - \boldsymbol{s}_f)^T\boldsymbol{R}(\boldsymbol{s}(i) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(i) + F_{rep}(i)) \end{array}\right) \\ \text{s.t.}\quad & \boldsymbol{x}(k+1) = F(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\ & G(\boldsymbol{x}(k), \boldsymbol{u}(k), k) \leq 0 \\ & k = 0, 1, 2, \ldots \end{aligned} \tag{22}$$

The exact penalty function (Bertsekas, 1975) eliminates constraints of the optimization problem, and the transformed problem becomes

$$\begin{aligned} &\min\{J_2(\boldsymbol{x}(k)) + \gamma\sum_{i=k}^{\infty}\max[0, G(\boldsymbol{x}(i), \boldsymbol{u}(i), i)]\} \\ &\text{s.t.}\quad \boldsymbol{x}(k+1) = F(\boldsymbol{x}(k), \boldsymbol{u}(k)) \end{aligned} \tag{23}$$

where γ is the penalty term of the exact penalty function.

To maximize the non-smooth operator (Nesterov, 2004), the smoothing approximation function (Lian, 2012) is used to replace the maximum function

$$p_{\varepsilon,\gamma,G}(k) = \begin{cases} 0, & G < -\frac{\varepsilon}{\gamma m} \\ \frac{\gamma m}{2\varepsilon}G^2 + G + \frac{\varepsilon}{2\gamma m}, & -\frac{\varepsilon}{\gamma m} \leq G < 0 \\ G + \frac{\varepsilon}{2\gamma m}, & 0 \leq G \end{cases} \tag{24}$$

where the operator $p_{\epsilon,\gamma,G}$ is the first-order differentiable of the constraint set $G(\boldsymbol{x}(k), \boldsymbol{u}(k), k)$, the notation $G = G(\boldsymbol{x}(k), \boldsymbol{u}(k), k)$ represents

the constraint set, the parameter ε is the smoothing coefficient, and the limitation is given by $\lim_{\varepsilon\to 0^+} p_{\varepsilon,\gamma,G} = \max\{0, G(\boldsymbol{x}(k), \boldsymbol{u}(k), k)\}$.

In other words, a smooth optimization problem (Beck and Teboulle, 2012) is obtained, which can replace the penalty function based problem (14), it is shown as

$$\begin{aligned} &\min \tilde{J}(\boldsymbol{x}(k)) = J_2(\boldsymbol{x}(k)) + \gamma \sum_{i=k}^{\infty} p_{\varepsilon,\gamma,G}(i) \\ &\text{s.t.} \quad \boldsymbol{x}(k+1) = F(\boldsymbol{x}(k), \boldsymbol{u}(k)) \end{aligned} \tag{25}$$

where $\tilde{J}(\boldsymbol{x}(k))$ is a new function. If the factor ε is small enough, then the solution can approximate the original problem. And if the parameter γ is large enough, then the solution is also approximately equal to the original problem.

The ADP searches for the solution to the dynamic path planning problem. The iterative approximation strategy is performed to get the optimal value and the optimal control. We assume that the $V_0(\cdot)$ is equal to 0. Then the corresponding initial control policy $v_0(\cdot)$ is obtained as follows

$$v_0(\boldsymbol{x}(k)) = \arg\min \begin{Bmatrix} \sigma_1(\|\boldsymbol{x}(k+1) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k) + F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_0(\boldsymbol{x}(k+1)) \end{Bmatrix} \tag{26}$$

Further iteration, performance index functions of the iteration $V_1(\cdot)$ can be obtained as follows

$$\begin{aligned} V_1(\boldsymbol{x}(k)) &= \min \begin{Bmatrix} \sigma_1(\|\boldsymbol{x}(k+1) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k) + F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_0(\boldsymbol{x}(k+1)) \end{Bmatrix} \\ &= \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k), v_0(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k) + F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_0(F(\boldsymbol{x}(k), v_0(\boldsymbol{x}(k)))) \end{Bmatrix} \end{aligned} \tag{27}$$

For the index $\{i = 1, 2, \ldots, \infty\}$, we can get iterative equations

$$v_i(\boldsymbol{x}(k)) = \arg\min \begin{Bmatrix} \sigma_1(\|\boldsymbol{x}(k+1) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k) + F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_i(\boldsymbol{x}(k+1)) \end{Bmatrix} \tag{28}$$

$$\begin{aligned} V_{i+1}(\boldsymbol{x}(k)) &= \min \begin{Bmatrix} \sigma_1(\|\boldsymbol{x}(k+1) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k) + F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_i(\boldsymbol{x}(k+1)) \end{Bmatrix} \\ &= \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k), v_i(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ +\sigma_2(F_{att}(k) + F_{rep}(k)) \\ +(\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_i(F(\boldsymbol{x}(k), v_i(\boldsymbol{x}(k)))) \end{Bmatrix} \end{aligned} \tag{29}$$

In the process of finding the approximate optimal solution at time k, cycles between strategy improvement and value determination are used until the value approaches optimal results.

The $V_i(\boldsymbol{x}(k)) \to J^*(\boldsymbol{x}(k))$, and $v_i(\boldsymbol{x}(k)) \to u^*(\boldsymbol{x}(k))$ as $i \to \infty$. The control policy at time k can be obtained in time forward, and the optimal path for the mobile robot can be generated. To clarify the problem solving procedure of our proposed method, the pseudocode is given in Algorithm 1, and the framework of the developed path planner is shown in Fig. 3.

Algorithm 1 Pseudocode of the proposed method

Input: Parameters of neural network: learning-speed factor, weights, discount factor, maximum iterative step
Output: Optimal path; Optimal control

```
Initialize the parameters of ADP
k=0
while true do
    Receive environment information from sensors
    Calculate the repulsive force and the attractive force
    Calculate the resultant force of the mobile robot
    Train the model net according to the kinematical equation (2)
    if Reach the terminal condition of ADP then
        Input x(k) to the action net to calculate u(k)
        Input x(k) and u(k) to the model net to calculate x(k+1)
        Construct the utility function U(x(k), u(k))
        Input x(k+1) to the critic net to calculate Ĵ(k+1)
        Input x(k) to the critic net to calculate Ĵ(k)
        Update the weight of the critic net and action net
    end if
    Obtain the control signal u*(k)
    Drive the mobile robot to the next state
    if the mobile robot reach the target location then
        break
    end if
    k=k+1
end while
return J*, x*, u*
```

4. Convergence of the iterative method

This section will prove that the control sequence $\boldsymbol{u}(k)$ and function $J(\boldsymbol{x}(k))$ got by the proposed method can approximate the $\boldsymbol{u}^*(k)$ and optimum $J^*(\boldsymbol{x}(k))$, respectively. Thus, the proposed method will not fall into the local optimum, can generate the optimal path, and avoid obstacles in solving the OCPs.

First, a theorem is given to show that the new performance index function achieved by our proposed iterative method is bounded. This theorem plays a key role in proving the following theorems.

Theorem 1. *Let the $\{v_i(\boldsymbol{x}(k))\}$ be a control sequence obtained by the proposed iterative algorithm at time k, and the sequence $\{V_i(\cdot)\}$ be the corresponding performance index functions gained by the iteration. Let $\{\mu_i(\boldsymbol{x}(k))\}$ be the control for the controlled dynamic system. Similar to the iterative formula, the following formula is holding*

$$\begin{aligned} \Lambda_{i+1}(\boldsymbol{x}(k)) &= \sigma_1(\|F(\boldsymbol{x}(k), \mu_i(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ &+ (\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ &+ \sigma_2(F_{att}(\boldsymbol{x}(k)) + F_{rep}(\boldsymbol{x}(k))) \\ &+ \gamma p_{\varepsilon,\gamma,G}(\boldsymbol{x}(k)) \\ &+ \Lambda_i(F(\boldsymbol{x}(k), \mu_i(\boldsymbol{x}(k)))) \end{aligned} \tag{30}$$

Let $\Lambda_0(\cdot) = V_0(\cdot) = 0$, there holds $V_i(\boldsymbol{x}(k)) \le \Lambda_i(\boldsymbol{x}(k))$ for any index i.

The demonstration of Theorem 1 is obvious. Theorem 2 will show the sequence of $V_i(\boldsymbol{x}(k))$ is upper bounded.

Theorem 2. *Let the sequence $\{V_i(\cdot)\}$ be the corresponding performance index functions got by the proposed algorithm. Let the dynamic system be*

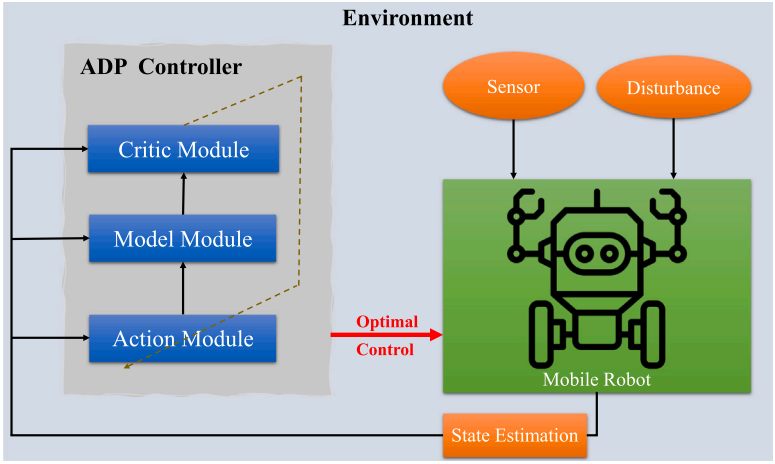

Fig. 3. Framework of the proposed method.

controllable. There is an upper bound Y, and the following formula holds for each i

$$0 \le V_i(\boldsymbol{x}(k)) \le Y \tag{31}$$

Proof. Let the sequence $\{\eta_i(\boldsymbol{x}(k))\}$ be an admissible control sequence. As time k towards infinity, the corresponding state value $\boldsymbol{x}(k)$ approaches zero, and the performance index function is a bounded value. Let $Z_0(\cdot) = V_0(\cdot) = 0$. Let the $\{\upsilon_i(\boldsymbol{x}(k))\}$ be a control achieved by the iteration. The $Z_i(\cdot)$ is calculated as follows

$$\begin{aligned} Z_{i+1}(\boldsymbol{x}(k)) = {} & \sigma_1(\|F(\boldsymbol{x}(k), \eta_i(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ & + (\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ & + \sigma_2(F_{att}(k) + F_{rep}(k)) \\ & + \gamma p_{\varepsilon,\gamma,G}(k) \\ & + Z_i(\boldsymbol{x}(k+1)) \end{aligned} \tag{32}$$

The function $U(\boldsymbol{x}(k))$ is written as follows

$$\begin{aligned} U(\boldsymbol{x}(k)) = {} & \sigma_1(\|F(\boldsymbol{x}(k), \eta_i(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ & + (\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ & + \sigma_2(F_{att}(k) + F_{rep}(k)) \\ & + \gamma p_{\varepsilon,\gamma,G}(k) \end{aligned} \tag{33}$$

Then we have the following equation

$$\begin{aligned} Z_{i+1}(\boldsymbol{x}(k)) = {} & U(\boldsymbol{x}(k)) + U(\boldsymbol{x}(k+1)) + \cdots \\ & + U(\boldsymbol{x}(k+i)) + Z_0(\boldsymbol{x}(k+i+1)) \end{aligned} \tag{34}$$

The function $U(\cdot)$ is positive definite. There exists a finite value Y for which the following formula holds

$$Z_{i+1}(\boldsymbol{x}(k)) \le \lim_{i\to\infty} \sum_{j=0}^{i} U(\boldsymbol{x}(k+j)) \le Y \tag{35}$$

Thus, according to the Theorem 1, the following conclusion can be drawn

$$0 \le V_{i+1}(\boldsymbol{x}(k)) \le Z_{i+1}(\boldsymbol{x}(k)) \le Y \quad \square \tag{36}$$

Theorems 1 and 2 illustrate that the function $V_i(\boldsymbol{x}(k))$ of our proposed method is upper bounded.

Theorem 3. *Let $\{\upsilon_i(\boldsymbol{x}(k))\}$ be the control which is obtained by iterations at time k, and $\{V_i(\cdot)\}$ be the corresponding performance index functions. Let $V_0(\cdot) = 0$, there holds $V_i(\boldsymbol{x}(k)) \le V_{i+1}(\boldsymbol{x}(k))$ for each i and $\lim_{i\to\infty} V_i(\boldsymbol{x}(k)) = J^*(\boldsymbol{x}(k))$.*

Proof. The sequence $\{\Phi_i(\boldsymbol{x}(k))\}$ is defined by

$$\begin{aligned} \Phi_{i+1}(\boldsymbol{x}(k)) = {} & \sigma_1(\|F(\boldsymbol{x}(k), \upsilon_{i+1}(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ & + (\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ & + \sigma_2(F_{att}(k) + F_{rep}(k)) \\ & + \gamma p_{\varepsilon,\gamma,G}(k) \\ & + \Phi_i(F(\boldsymbol{x}(k), \upsilon_{i+1}(\boldsymbol{x}(k)))) \end{aligned} \tag{37}$$

Next, let $\Phi_0(\cdot) = 0$, and the mathematical induction (Bussey, 1917) demonstrates the following inequality equation

$$\Phi_i(\boldsymbol{x}(k)) \le V_{i+1}(\boldsymbol{x}(k)) \tag{38}$$

We have

$$\begin{aligned} V_1(\boldsymbol{x}(k)) - \Phi_0(\boldsymbol{x}(k)) = {} & \sigma_1(\|F(\boldsymbol{x}(k), \upsilon_0(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ & + (\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ & + \sigma_2(F_{att}(k) + F_{rep}(k)) \\ & + \gamma p_{\varepsilon,\gamma,G}(k) \\ & \ge 0 \end{aligned} \tag{39}$$

Thus, $\Phi_0(\boldsymbol{x}(k)) \le V_1(\boldsymbol{x}(k))$.

Next, assuming that $\Phi_{i-1}(\boldsymbol{x}(k)) \le V_i(\boldsymbol{x}(k))$, then

$$\begin{aligned} \Phi_i(\boldsymbol{x}(k)) = {} & \sigma_1(\|F(\boldsymbol{x}(k), \upsilon_i(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ & + (\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ & + \sigma_2(F_{att}(k) + F_{rep}(k)) \\ & + \gamma p_{\varepsilon,\gamma,G}(k) \\ & + \Phi_{i-1}(F(\boldsymbol{x}(k), \upsilon_i(\boldsymbol{x}(k)))) \end{aligned} \tag{40}$$

and

$$\begin{aligned} V_{i+1}(\boldsymbol{x}(k)) = {} & \sigma_1(\|F(\boldsymbol{x}(k), \upsilon_i(\boldsymbol{x}(k))) - \boldsymbol{x}(k)\|_2^2) \\ & + (\boldsymbol{s}(k) - \boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k) - \boldsymbol{s}_f) \\ & + \sigma_2(F_{att}(k) + F_{rep}(k)) \\ & + \gamma p_{\varepsilon,\gamma,G}(k) \\ & + V_i(F(\boldsymbol{x}(k), \upsilon_i(\boldsymbol{x}(k)))) \end{aligned} \tag{41}$$

This gives us the following equation

$$V_{i+1}(\boldsymbol{x}(k)) - \Phi_i(\boldsymbol{x}(k)) = V_i(\boldsymbol{x}(k)) - \Phi_{i-1}(\boldsymbol{x}(k)) \ge 0 \tag{42}$$

Then, we have $\Phi_i(\boldsymbol{x}(k)) \le V_{i+1}(\boldsymbol{x}(k))$.

Combined with the Theorem 2, there holds $V_i(\boldsymbol{x}(k)) \le \Phi_i(\boldsymbol{x}(k)) \le V_{i+1}(\boldsymbol{x}(k))$, $\{V_i(\boldsymbol{x}(k))\}$ has an upper bound. $\square$

Theorem 4. *Let $V_\infty(\boldsymbol{x}(k)) = \lim_{i\to\infty} V_i(\boldsymbol{x}(k))$ and $\boldsymbol{x}(k)$ be the state value of dynamic systems that can be fetched. The $V_\infty(x_k)$ satisfies the HJB equation*

$$V_\infty(\boldsymbol{x}(k)) = \min_{\boldsymbol{u}(k)} \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k),\boldsymbol{u}(k)) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k)-\boldsymbol{s}_f)^T \boldsymbol{R}(\boldsymbol{s}(k)-\boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k)+F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_\infty(F(\boldsymbol{x}(k),\boldsymbol{u}(k))) \end{Bmatrix} \tag{43}$$

Proof. Combined with Theorem 3, there holds the following inequality equation for arbitrarily chosen control v and i

$$\begin{aligned} V_i(\boldsymbol{x}(k)) \le & \sigma_1(\|F(\boldsymbol{x}(k),\boldsymbol{v}(k)) - \boldsymbol{x}(k)\|_2^2) \\ & +(\boldsymbol{s}(k)-\boldsymbol{s}_f)^T R(\boldsymbol{s}(k)-\boldsymbol{s}_f) \\ & +\sigma_2(F_{att}(k)+F_{rep}(k)) \\ & +\gamma p_{\varepsilon,\gamma,G}(k) \\ & +V_\infty(\boldsymbol{x}(k+1)) \end{aligned} \tag{44}$$

Therefore, the following equation holds

$$V_i(\boldsymbol{x}(k)) \le \min_{v} \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k),\boldsymbol{v}(k)) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k)-\boldsymbol{s}_f)^T R(\boldsymbol{s}(k)-\boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k)+F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_\infty(\boldsymbol{x}(k+1)) \end{Bmatrix} \tag{45}$$

When the iterative times $i \to \infty$, we get the following inequality equation

$$V_\infty(\boldsymbol{x}(k)) \le \min_{v} \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k),\boldsymbol{v}(k)) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k)-\boldsymbol{s}_f)^T R(\boldsymbol{s}(k)-\boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k)+F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_\infty(\boldsymbol{x}(k+1)) \end{Bmatrix} \tag{46}$$

In addition, since

$$V_i(\boldsymbol{x}(k)) = \min_{v} \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k),\boldsymbol{v}(k)) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k)-\boldsymbol{s}_f)^T R(\boldsymbol{s}(k)-\boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k)+F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_{i-1}(\boldsymbol{x}(k+1)) \end{Bmatrix} \tag{47}$$

for each i,

$$V_\infty(\boldsymbol{x}(k)) \ge \min_{v} \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k),\boldsymbol{v}(k)) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k)-\boldsymbol{s}_f)^T R(\boldsymbol{s}(k)-\boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k)+F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_{i-1}(\boldsymbol{x}(k+1)) \end{Bmatrix} \tag{48}$$

Let i be the iteration times. When $i \to \infty$, we can get the following inequality equation

$$V_\infty(\boldsymbol{x}(k)) \ge \min_{v} \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k),\boldsymbol{v}(k)) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k)-\boldsymbol{s}_f)^T R(\boldsymbol{s}(k)-\boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k)+F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_\infty(\boldsymbol{x}(k+1)) \end{Bmatrix} \tag{49}$$

Table 1
Parameters of the APF in our method.

Parameters	Values
Attractive coefficient K_{att}	3
Repulsive coefficient K_{rep}	10
Perception range p/m	1
Repulsive decay factor n	2

Finally, the following equation is proved

$$V_\infty(\boldsymbol{x}(k)) = \min_{v} \begin{Bmatrix} \sigma_1(\|F(\boldsymbol{x}(k),\boldsymbol{v}(k)) - \boldsymbol{x}(k)\|_2^2) \\ +(\boldsymbol{s}(k)-\boldsymbol{s}_f)^T R(\boldsymbol{s}(k)-\boldsymbol{s}_f) \\ +\sigma_2(F_{att}(k)+F_{rep}(k)) \\ +\gamma p_{\varepsilon,\gamma,G}(k) \\ +V_\infty(\boldsymbol{x}(k+1)) \end{Bmatrix} \quad \square \tag{50}$$

Theorem 3 states that the function $V_i(\boldsymbol{x}(k))$ obtained by the proposed method converges to a bounded value. Theorem 4 shows that the sequence $\{\boldsymbol{u}(k)\}$ can converge to the $\boldsymbol{u}^*(k)$. Based on the above theorems, our method can generate a feasible and safe path from the initial position to the target position theoretically.

5. Simulation results

To evaluate of our proposed method, several simulations under different complexities are tested on the Matlab R2021b platform for dynamic path planning problems of differentially-driven mobile robots. The software environment for all simulations is based on the Windows 10 system, and the hardware environment is based on a 2.90 GHz, Intel(R) Core i5-10700M CPU with 16 GB RAM.

For all simulations in this paper, the initial position of the robot is set as the origin of the global coordinate system, the state $x_1(k)$ is from -1 m to 21 m, $x_2(k)$ is from -1 m to 21 m, $x_3(k)$ is from $-\pi$ to π. For the differentially-driven model (Almomani et al., 2021), the distance between two driven wheels $d = 0.37$ m, the control $u_1(k) \in [0\ \text{m/s}, 1\ \text{m/s}]$ and $u_2(k) \in [0\ \text{m/s}, 1\ \text{m/s}]$, the safety distance $\beta = 0.3$ m, the sampling interval $\Delta t = 0.1$ s. The weighted factors $\sigma_1 = \sigma_2 = 0.5$, and the weight matrix $\boldsymbol{R}$ is an identity matrix. The parameters of the APF in our method are displayed in Table 1.

In this section, our proposed method is compared with some state-of-the-art path planning methods, such as APF (Wang et al., 2022), DWA (Lee et al., 2021), and dynamic A-star algorithm (Yan et al., 2022). In each simulation experiment, the parameters for all methods are the same.

5.1. Case 1: Unknown environment with static obstacles

In this case, the differentially-driven mobile robot is tested in an unknown environment with many static obstacles. We set the mission environment as a 2D plane with a ground environment, whose region is 22 m × 22 m, as illustrated in Fig. 4. There are three scenes to test the efficiency and robustness of our method, with the number of static obstacles increasing from 9 to 15. The initial position is randomly generated in the range $x_1(0) \in [-0.5\ \text{m}, 0.5\ \text{m}]$, $x_2(0) \in [-0.25\ \text{m}, 0.25\ \text{m}]$, and the initial heading angle $x_3(0) = 0$ rad. There are three targets in Scenes I-III and their positions are displayed in Table 2. When the distance between the mobile robot and the target position is less than 0.045 m, the robot is considered to have reached the target. In this part, the number of repetitions for each simulated group is 50.

In Scene I, the paths generated by APF, DWA, dynamic A-star, and our method are plotted in Fig. 5. We can see that all four methods can arrive at the target and generate the feasible path in an unknown environment. We also compare the average distance traveled, the average computing time, and the success rates for the proposed method

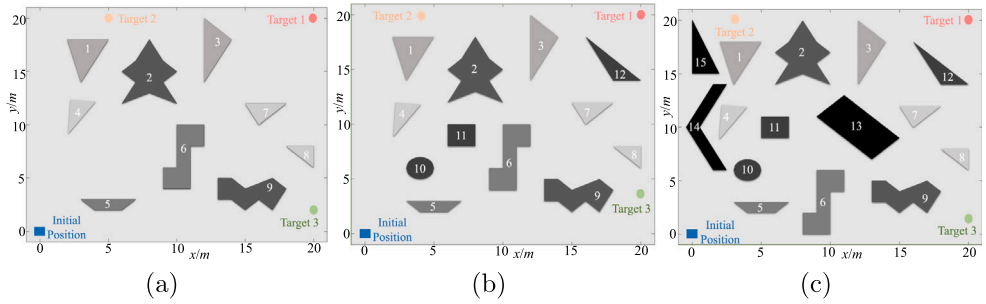

Fig. 4. The map information of (a) Scene I, (b)Scene II, and (c) Scene III.

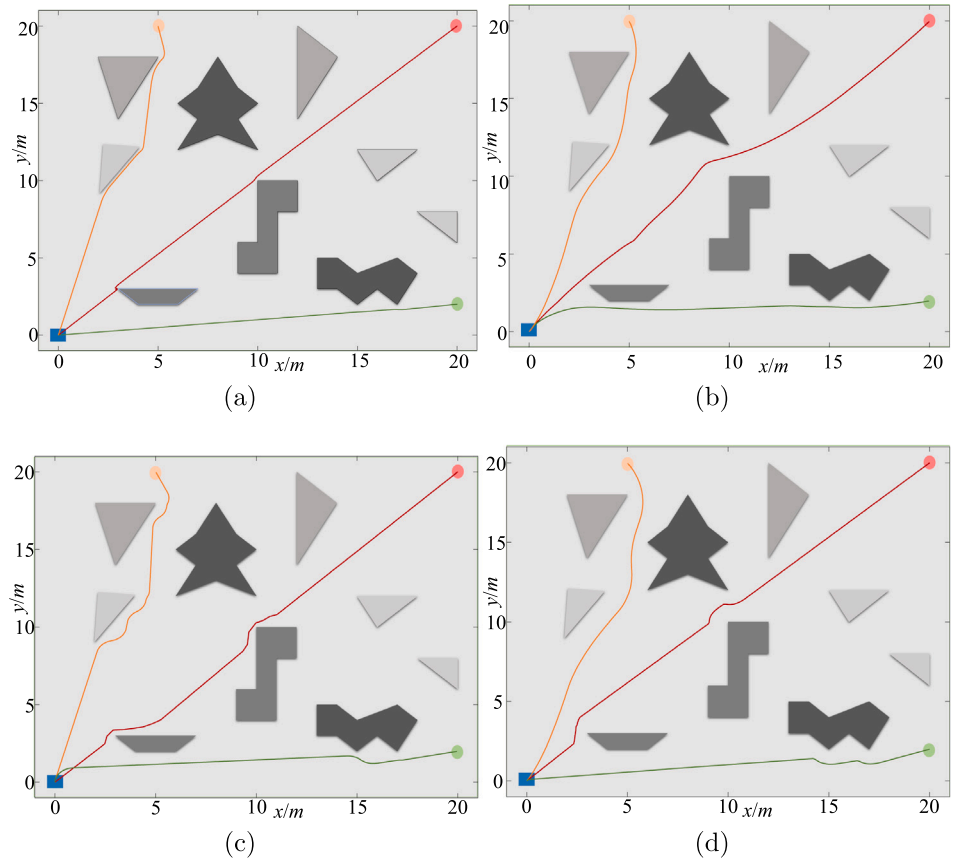

Fig. 5. Planning results of Scene I with (a) APF, (b) DWA, (c) Dynamic A-star, and (d) our method.

Table 2
The target information in Scenes I-III.

Target	Scene I	Scene II	Scene III	Color
Target 1/m	(20,20)	(20,20)	(20,20)	Red
Target 2/m	(20,02)	(20,04)	(20,02)	Orange
Target 3/m	(05,20)	(04,20)	(03,20)	Green

and these comparative methods in this case, as shown in Table 3. From this table, it can be found that all methods can achieve similar traveled lengths in this relatively simple environment. At the time level, the DWA will consume the most computational resources. The other three methods consume approximately the same amount of time in Scene I. The success rates for all four methods are close to 100% when the target position changes. The two simulations where APF fails are due to the fact that this method is sensitive to obstacles and may fall into a local optimum. In simple environments, our method can achieve excellent performance comparable to that of state-of-the-art path planning methods. Next, we will discuss the performance of these three path planning methods in scenes with more complex obstacles.

In Scene II, we add obstacle-10, obstacle-11, and obstacle-12 to the environment to validate these path planning methods, as plotted in Fig. 4(b). Obstacle-12 is located close to target 1 and can impede the motion of the mobile robot. The paths generated by these methods are plotted in Fig. 6. The dynamic A-star, and our method are able to reach the target and obtain a successful path in this environment. In all fifty repetitions, obstacle-12 successfully prevented the APF-driven mobile robot from reaching target 1, and the main reason for the failure of the DWA to reach target 3 is obstacle-9, and suffers the infinite loop problem. To reach target 2 and target 3, the APF and the dynamic A-star method produce paths with severe oscillations, which may cause damage to the structure of the mobile robot. From Table 3, we can see that the average distance traveled has increased due to a more complex environment, and the DWA also consumes the most computing time. The success rate of all four methods decreases in this scene, but our proposed method is the least affected, with only one simulation

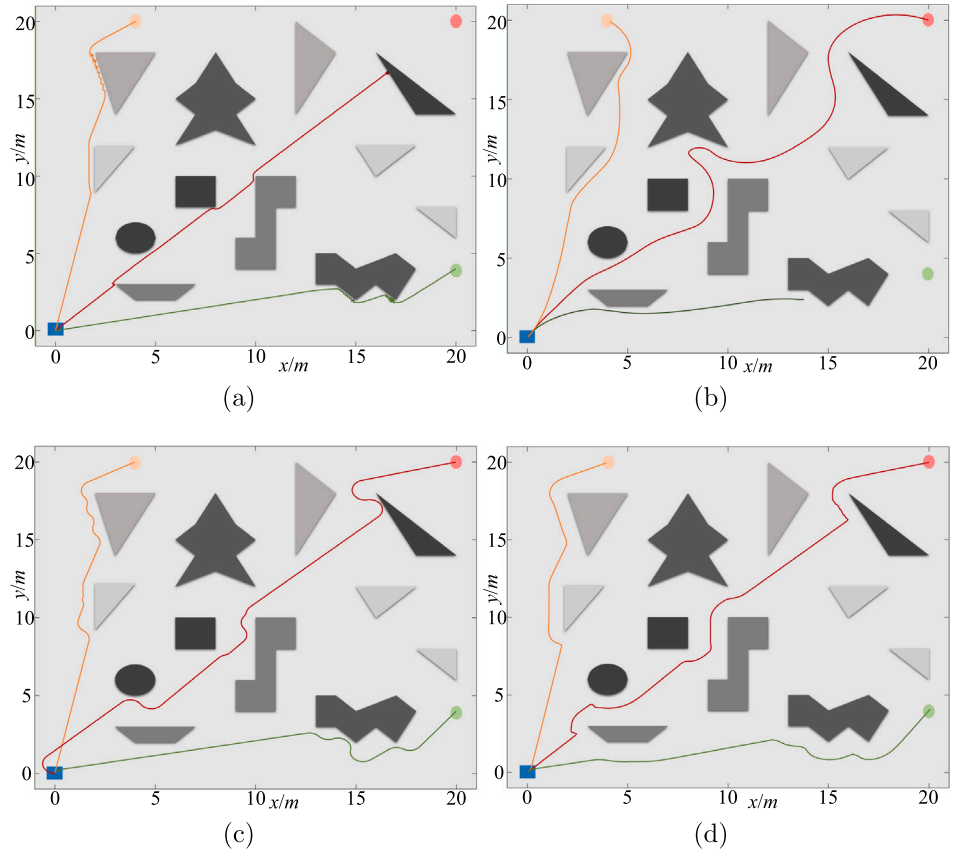

Fig. 6. Planning results of Scene II with (a) APF, (b) DWA, (c) dynamic A-star, and (d) our method.

to target 1 failing. In this simulation, the ADP was not well trained and the control signals generated by the ADP did not help the robot to achieve the correct state when the mobile robot approached obstacle-12. When the APF is applied to the autonomous navigation of target 1, the success rate is only 0%, the same failure occurred in the situation where the DWA drives the mobile robot to target 3. In this scene, the proposed method outperforms the APF and the DWA methods in mission completion and shows a better performance than the dynamic A-star method on the path oscillation phenomenon.

In order to test the proposed method in the unknown narrow environment, we change the position of the original obstacles and add obstacle-13, obstacle-14, and obstacle-15 in Scene III. The distribution of these obstacles is shown in Fig. 4(c). Obstacle-13 is located on the line between the initial position area and target 1, which can further increase the difficulty, obstacle-14 and obstacle-15 will provide a great challenge for the mobile robot to reach the target 2, and the new position of obstacle-6 blocks the movement of the mobile robot to reach target 3. The space in which the mobile robot can move in this scene becomes narrow and crowded, and the performance of the path planning methods is greatly tested. The experimental results for avoiding these obstacles are shown in Fig. 7 and Table 3. In Fig. 7, only our method can successfully guide the mobile robot to target 1, target 2, and target 3, respectively. When the destinations are target 1 and target 3, the APF falls into a local optimum, and the path to successfully reach target 2 is highly oscillating. All simulations of DWA have failed. The method stops iterating on the way to reach target 1 and target 2, and the mobile robot drives out of bounds when searching for target 3. The dynamic A-star suffers a similar situation to the APF. From Table 3, we can see that only our method has shown outstanding performance in the unknown narrow environment.

From the above simulations, it can be seen that our proposed method has the advantage of good real-time planning and can avoid static obstacles in an unknown environment. And the kinematical equation and constraints of the differentially-driven mobile robot can be satisfied. Figs. 5 to 7 show the path generated by our method and three comparative methods. Table 3 gives the average distance traveled, the average computing time and the success rate for all methods in Scenes I-III. All methods can generate a feasible path in a simple environment. However, as the number of obstacles increases, the performance of these comparative methods decreases. Compared to the APF method, our method can avoid the local minimum, and always keep an appropriate distance from obstacles, which overcomes the main problems of the APF method. Although the APF method generates shorter paths and consumes less time, it can only successfully generate paths in a certain simple environment, and the oscillation phenomenon is dangerous. The DWA method is time consuming even in simple spaces, and the narrow environment will limit its application in autonomous navigation. In Scenes I-III, the dynamic A-star method outperforms the APF and the DWA, but is still heavily influenced by obstacles. Our method has demonstrated similar performance to state-of-the-art path planning methods in simple spaces. When the environment becomes narrow and crowded, the performance of our method does not drop significantly. Hence, the proposed method is suitable for the autonomous navigation of differentially-driven mobile robots in unknown environments with static obstacles.

5.2. Case 2: Unknown environment with dynamic obstacles

In this case, our method guides the differentially-driven mobile robot in an unknown environment with some dynamic obstacles. The distribution of static obstacles in this case is consistent with Scene II in Case 1, as illustrated in Fig. 4(b). The initial state is set as $\boldsymbol{x}(0) = (0\ \text{m}, 0\ \text{m}, 0\ \text{rad})$, and the target position is target 1 in Scene II, i.e. $(20\ \text{m}, 20\ \text{m})$. In Case 1, we have verified the effectiveness of

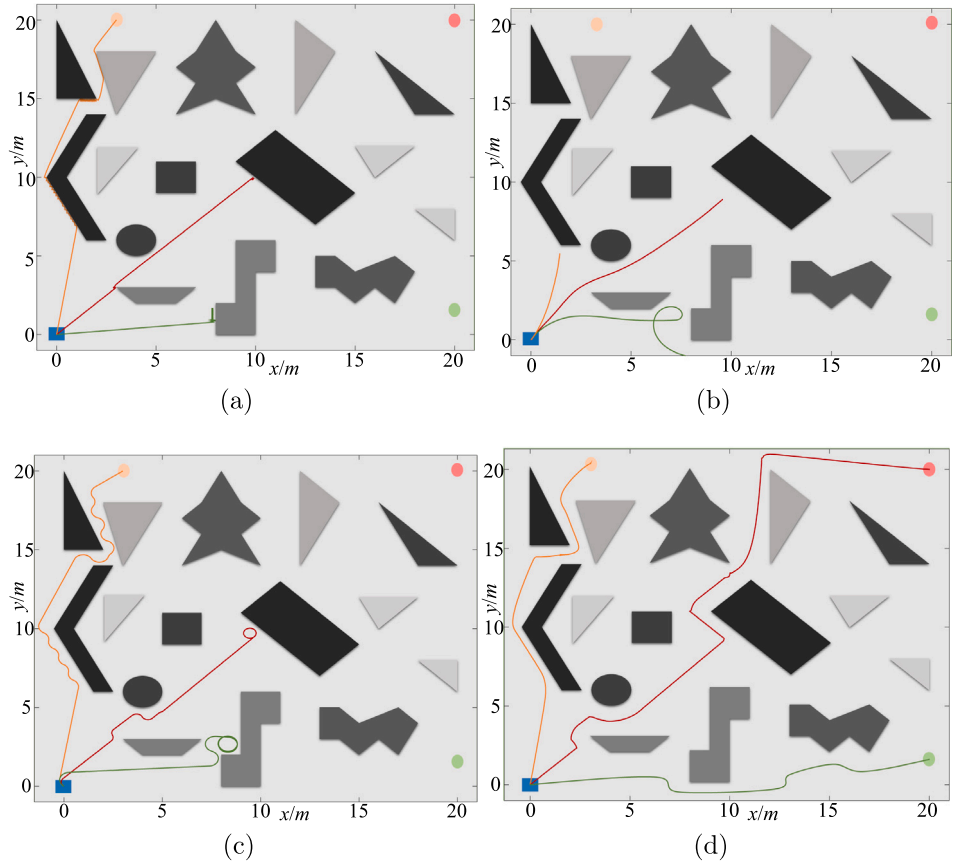

Fig. 7. Planning results of Scene III with (a) APF, (b) DWA, (c) dynamic A-star, and (d) our method.

Table 3
Simulation result from different methods of Scenes I-III.

S-T	APF			DWA			Dynamic A-star			Our method		
	Length /m	Time /s	Success rate	Length /m	Time /s	Success rate	Length /m	Time /s	Success rate	Length /m	Time /s	Success rate
I-1	28.59	18.03	100%	29.57	47.26	100%	29.56	20.77	100%	28.89	19.21	100%
I-2	21.25	11.93	98%	20.97	31.76	100%	21.65	13.70	100%	21.44	15.17	100%
I-3	20.13	10.55	98%	20.56	30.93	100%	20.71	12.16	100%	20.51	12.10	100%
II-1	–	–	0%	35.74	62.67	92%	33.85	38.56	90%	31.11	30.91	98%
II-2	22.90	15.63	84%	21.36	44.64	94%	22.35	18.77	94%	22.40	24.87	100%
II-3	21.24	15.47	88%	–	–	0%	23.15	19.98	94%	23.39	17.17	100%
III-1	–	–	0%	–	–	0%	–	–	0%	37.76	38.17	94%
III-2	24.87	23.41	76%	–	–	0%	25.90	26.55	88%	24.95	25.71	100%
III-3	–	–	0%	–	–	0%	–	–	0%	20.71	12.87	98%

Table 4
Detail information of two dynamic obstacles in Case 2.

Number	Initial position	Moving direction
Pedestrian-1	(10 m,13 m)	Rightwards
Pedestrian-2	(15 m,13 m)	Upwards

our method in this environment without dynamic obstacles. Now, two pedestrians are introduced into the environment as dynamic obstacles, and their detailed information is displayed in Table 4. To better illustrate the problem solving process, only the path results after two pedestrian appearances are discussed, as plotted in Fig. 8. The red line represents the path of the mobile robot, the blue dashed line is the path of pedestrian-1, and the green dotted line is the path of pedestrian-2.

Fig. 8 shows the position of the differentially-driven mobile robot at different time steps (k=160, 170, 190,210, 225, 260, 270, 285, and 325). Before $k = 160$, the real-time path is almost the same as that of Scene II in Case 1. When $k = 160$, two pedestrians suddenly appear in the environment and start moving, and the mobile robot's information about these pedestrians is unknown, as in Fig. 8(a). When $k = 170$, pedestrian-1 moves to the right and pedestrian-2 moves upwards, as in Fig. 8(b). When $k = 190$, the mobile robot perceives pedestrian-1 and starts to avoid the dynamic obstacle by turning downwards, as in Fig. 8(c). When $k = 210$, the mobile robot is detouring pedestrian-1, as in Fig. 8(d). When $k = 225$, the mobile robot has already avoided pedestrian-1 and perceives pedestrian-2, as in Fig. 8(e). Under the influence of pedestrian-2 and static obstacle-3, our method guides the mobile robot upwards, as plotted in Fig. 8(f). Then the mobile robot decides to avoid the obstacles by moving to the right when $k = 270$, as in Fig. 8(g). The mobile robot essentially achieves the avoidance of pedestrian-2 when $k = 285$, as in Fig. 8(h). After that, the navigation path is influenced only by static obstacles and the target and reaches the target position at $k = 325$, as in Fig. 8(i).

From Case 1 and Case 2, we have demonstrated that the proposed method can guide the differentially-driven mobile robot to the target by avoiding dynamic and static obstacles in the unknown environment. Compared with three state-of-the-art path planning methods, our method has advantages in obstacle-dense environments. Case 2 shows

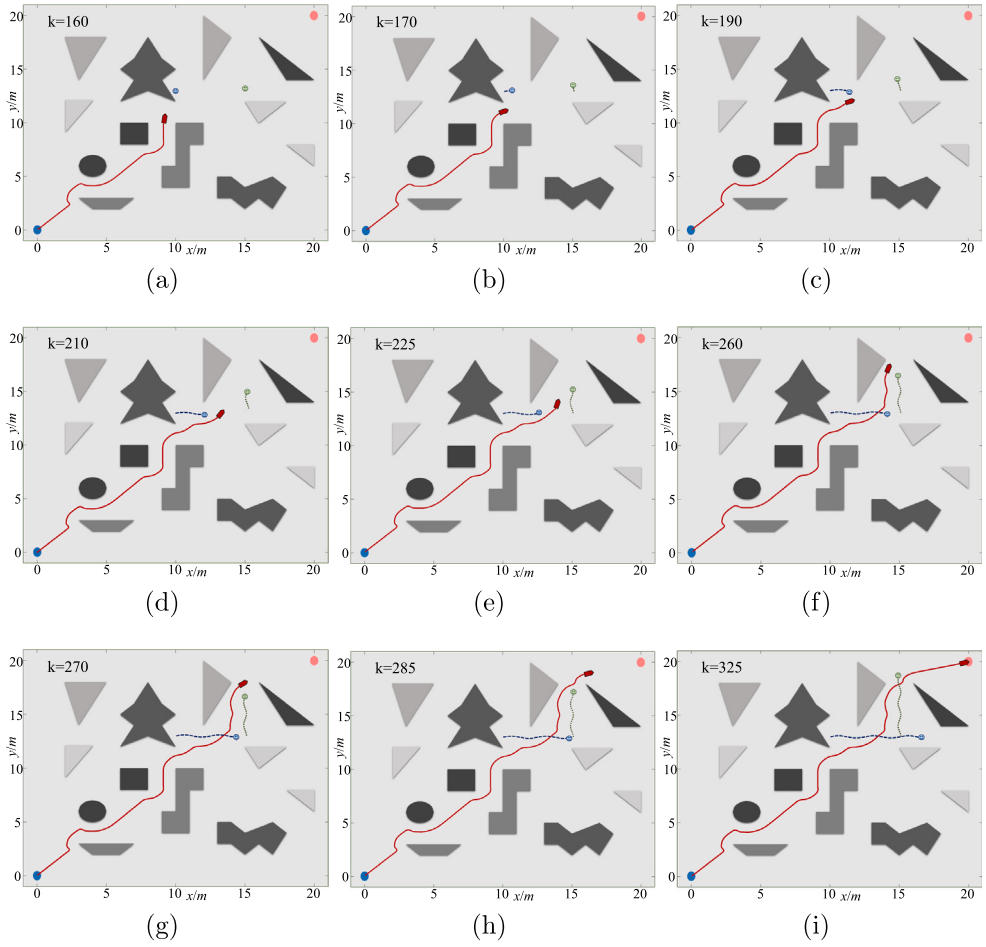

Fig. 8. Path history at different time steps with (a) k=160, (b)k=170, (c) k=190, (d) k=210, (e)k=225, (f) k=260,(g) k=270, (h) k=285, and (i) k=325.

that our method can control the mobile robot to continuously avoid dynamic obstacles. The above simulations highlight that our method is robust and efficient in dynamic path planning for differentially-driven mobile robots.

6. Conclusions

In this paper, we develop a novel dynamic path planning method for the differentially-driven mobile robot, which combines the idea of ADP and APF. The main technical contributions include the transformation of the original dynamic problem into an OCP, the design of a new performance index function, and the proof of convergence. The proposed method can combine the advantages of path optimization and the kinematical constraints of the differentially-driven mobile robot. The newly designed performance index function includes the total distance traveled, the effect of the heading angle, the distance from the target to the robot, and the resultant force of the APF. The proposed method can generate a feasible path and avoid obstacles in an unknown environment. The theoretical analysis proves the convergence of our method. The sequence $\{\boldsymbol{u}(k)\}$ and function $J(\boldsymbol{x}(k))$ can converge to the $\{\boldsymbol{u}^*(k)\}$ and $J^*(\boldsymbol{x}(k))$. These features demonstrate that our proposed method is accurate, reliable, and suitable for autonomous navigation applications. Compared with some state-of-the-art path planning methods, simulations exhibit the effectiveness and robustness of the proposed method. Our method can successfully drive the differentially-driven model to the target position in unknown environments with static and dynamic obstacles. We will focus on exploring dynamic path planning techniques on some other mobile robot models, such as car-like robot models and UAVs, and the physical experiments on mobile robots through embedded development will also be conducted.

CRediT authorship contribution statement

Xin Li: Conceptualization, Methodology, Software, Visualization, Writing – original draft. **Lei Wang:** Supervision, Resources, Formal analysis. **Yi An:** Resources, Writing – review & editing. **Qi-Li Huang:** Methodology, Software. **Yun-Hao Cui:** Visualization, Validation. **Huo-Sheng Hu:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Grants 12161076, 62173055 and 61673083, in part by the Science and Technology Major Project of Shanxi Province under Grant 20191101014, and in part by the Fundamental Research Funds for the Central Universities, China under Grant DUT21LAB301.

References

Almomani, F., Al-Tamimi, A., Al-Jarrah, A., & Salah, M. (2021). Discrete optimal tracking control for a two-wheel mobile robot driven by DC motors. In *2021 IEEE Jordan International joint conference on electrical engineering and information technology* (pp. 25–30).

Ammar, A., Bennaceur, H., Châari, I., Koubâa, A., & Alajlan, M. (2016). Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments. *Soft Computing*, *20*(10), 4149–4171.

Beck, A., & Teboulle, M. (2012). Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, *22*(2), 557–580.

Bertsekas, D. P. (1975). Necessary and sufficient conditions for a penalty method to be exact. *Mathematical Programming*, *9*, 87–99.

Bussey, W. H. (1917). The origin of mathematical induction. *American Mathematical Monthly*, *24*(5), 199–207.

Chen, P., Pei, J., Lu, W., & Li, M. (2022). A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing*, *497*, 64–75.

Chen, Z., Wu, H., Chen, Y., Cheng, L., & Zhang, B. (2022). Patrol robot path planning in nuclear power plant using an interval multi-objective particle swarm optimization algorithm. *Applied Soft Computing*, *116*, Article 108192.

Dian, S., Zhong, J., Guo, B., Liu, J., & Guo, R. (2022). A smooth path planning method for mobile robot using a BES-incorporated modified QPSO algorithm. *Expert Systems with Applications*, *208*, Article 118256.

Fan, J., Chen, X., & Liang, X. (2023). UAV trajectory planning based on bi-directional APF-RRT* algorithm with goal-biased. *Expert Systems with Applications*, *213*, Article 119137.

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, *4*(1), 23–33.

González, D., Pérez, J., Milanés, V., & Nashashibi, F. (2016). A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *17*(4), 1135–1145.

Hansen, E., & Wang, Y. (2020). Improving path accuracy for autonomous parking systems: An optimal control approach. In *2020 American control conference* (pp. 5243–5249).

Ji, J., Khajepour, A., Melek, W. W., & Huang, Y. (2017). Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. *IEEE Transactions on Vehicular Technology*, *66*(2), 952–964.

Jones, M., Djahel, S., & Welsh, K. (2023). Path-planning for unmanned aerial vehicles with environment complexity considerations: A survey. *ACM Computing Surveys*, *55*(11), 1–39.

Khatib, O. (1990). Real-time obstacle avoidance for manipulators and mobile robots. In I. J. Cox, & G. T. Wilfong (Eds.), *Autonomous robot vehicles* (pp. 396–404). Springer New York.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Pérez, P. (2022). Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *23*(6), 4909–4926.

Kowsar, Y., Moshtaghi, M., Velloso, E., Leckie, C., & Kulik, L. (2022). An online unsupervised dynamic window method to track repeating patterns from sensor data. *IEEE Transactions on Cybernetics*, *52*(6), 5148–5160.

Kyaw, P. T., Le, A. V., Veerajagadheswar, P., Elara, M. R., Thu, T. T., Nhan, N. H. K., Van Duc, P., & Vu, M. B. (2022). Energy-efficient path planning of reconfigurable robots in complex environments. *IEEE Transactions on Robotics*, *38*(4), 2481–2494.

Ladosz, P., Weng, L., Kim, M., & Oh, H. (2022). Exploration in deep reinforcement learning: A survey. *Information Fusion*, *85*, 1–22.

Lee, D. H., Lee, S. S., Ahn, C. K., Shi, P., & Lim, C.-C. (2021). Finite distribution estimation-based dynamic window approach to reliable obstacle avoidance of mobile robot. *IEEE Transactions on Industrial Electronics*, *68*(10), 9998–10006.

Li, X., Wang, L., Wang, H., Tao, L., & Wang, X. (2023). A warm-started trajectory planner for fixed-wing unmanned aerial vehicle formation. *Applied Mathematical Modelling*, *122*, 200–219.

Lian, S. (2012). Smoothing approximation to L1 exact penalty function for inequality constrained optimization. *Applied Mathematics and Computation*, *219*(6), 3113–3121.

Liu, L., Wang, X., Yang, X., Liu, H., Li, J., & Wang, P. (2023). Path planning techniques for mobile robots: Review and prospect. *Expert Systems with Applications*, *227*, Article 120254.

Liu, D., Xue, S., Zhao, B., Luo, B., & Wei, Q. (2021). Adaptive dynamic programming for control: A survey and recent advances. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *51*(1), 142–160.

Low, E. S., Ong, P., Low, C. Y., & Omar, R. (2022). Modified Q-learning with distance metric and virtual target on path planning of mobile robot. *Expert Systems with Applications*, *199*, Article 117191.

Mu, C., Ni, Z., Sun, C., & He, H. (2017). Air-breathing hypersonic vehicle tracking control based on adaptive dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(3), 584–598.

Nesterov, Y. (2004). Smooth minimization of non-smooth functions. *Mathematical Programming*, *103*(1), 127–152.

Ntakolia, C., Moustakidis, S., & Siouras, A. (2023). Autonomous path planning with obstacle avoidance for smart assistive systems. *Expert Systems with Applications*, *213*, Article 119049.

Pietrzykowski, Z., Wolejsza, P., Nozdrzykowski, L., Borkowski, P., Banas, P., Magaj, J., Chomski, J., Maka, M., Mielniczuk, S., Panka, A., Hatlas-Sowinska, P., Kulbiej, E., & Nozdrzykowska, M. (2022). The autonomous navigation system of a sea-going vessel. *Ocean Engineering*, *261*, Article 112104.

Rosas, U. O., Montiel, O., & Sepúlveda, R. (2019). Mobile robot path planning using membrane evolutionary artificial potential field. *Applied Soft Computing*, *77*, 236–251.

Sangiovanni, B., Incremona, G. P., Piastra, M., & Ferrara, A. (2021). Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning. *IEEE Control Systems Letters*, *5*(2), 397–402.

Sathiya, V., Chinnadurai, M., & Ramabalan, S. (2022). Mobile robot path planning using fuzzy enhanced improved Multi-Objective particle swarm optimization (FIMOPSO). *Expert Systems with Applications*, *198*, Article 116875.

Sedighi, S., Nguyen, D. V., & Kuhnert, K. D. (2019). Guided hybrid A-star path planning algorithm for valet parking applications. In *Proceedings of 5th International conference on control, automation and robotics* (pp. 570–575).

Teng, J., An, Y., & Wang, L. (2022). Time-optimal control problem for a linear parameter varying system with nonlinear item. *Journal of the Franklin Institute*, *359*(2), 859–869.

Wahab, M. N. A., Nefti-Meziani, S., & Atyabi, A. (2020). A comparative review on mobile robot path planning: Classical or meta-heuristic methods? *Annual Reviews in Control*, *50*, 233–252.

Wang, X., Deng, Z., Peng, H., Wang, L., Wang, Y., Tao, L., Lu, C., & Peng, Z. (2023). Autonomous docking trajectory optimization for unmanned surface vehicle: A hierarchical method. *Ocean Engineering*, *279*, Article 114156.

Wang, X., Liu, J., Peng, H., Qie, X., Zhao, X., & Lu, C. (2022). A simultaneous planning and control method integrating APF and MPC to solve autonomous navigation for USVs in unknown environments. *Journal of Intelligent and Robotic Systems*, *105*(36), 1–16.

Wang, X., Liu, J., Su, X., Peng, H., Zhao, X., & Lu, C. (2020). A review on carrier aircraft dispatch path planning and control on deck. *Chinese Journal of Aeronautics*, *33*(12), 3039–3057.

Wang, Y., Miao, Z., Zhong, H., & Pan, Q. (2015). Simultaneous stabilization and tracking of nonholonomic mobile robots: A Lyapunov-based approach. *IEEE Transactions on Control Systems Technology*, *23*(4), 1440–1450.

Wang, X., Peng, H., Zhang, S., Chen, B., & Zhong, W. (2017). A symplectic pseudospectral method for nonlinear optimal control problems with inequality constraints. *ISA Transactions*, *68*, 335–352.

Wang, F.-Y., Zhang, H., & Liu, D. (2009). Adaptive dynamic programming: An introduction. *IEEE Computational Intelligence Magazine*, *4*(2), 39–47.

Wei, Q., Han, L., & Zhang, T. (2022). Spiking adaptive dynamic programming based on Poisson process for discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(5), 1846–1856.

Werbos, P. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems*, *22*, 25–38.

Wu, J., Wang, H., Zhang, M., & Yu, Y. (2021). On obstacle avoidance path planning in unknown 3D environments: A fluid-based framework. *ISA Transactions*, *111*, 249–264.

Yan, H., Zhu, Q., Zhang, Y., Li, Z., & Du, X. (2022). An obstacle avoidance algorithm for unmanned surface vehicle based on a star and velocity-obstacle algorithms. In *2022 IEEE 6th Information technology and mechatronics engineering conference, vol. 6* (pp. 77–82).

Zhang, Z., Li, J., & Wang, J. (2018). Sequential convex programming for nonlinear optimal control problems in UAV path planning. *Aerospace Science and Technology*, *76*, 280–290.