Theory and Practice of Science and Technology
2025, VOL. 6, NO. 12, 165-171
DOI: 10.47297/taposatWSP2633-456935.20250612

Path Planning and Motion Control of Unmanned Vehicles Based on Reinforcement Learning

Hongxiang Chen

University of Science and Technology Liaoning, Anshan, Liaoning, 114051, China

ABSTRACT
Path planning and motion control are core technologies for autonomous navigation of unmanned vehicles (UAVs). Addressing the problems of poor adaptability and reliance on precise models in dynamic environments, traditional methods propose a reinforcement learning-based path planning and motion control method for UAVs, focusing on solving key technical issues in practical deployment from an application perspective. First, a collaborative architecture integrating A* global planning and reinforcement learning local control is constructed. The A* algorithm generates the global guidance path, while a deep reinforcement learning agent is responsible for dynamic obstacle avoidance and trajectory tracking. Based on this, the engineering implementation details, including state space design, reward function construction, and network structure configuration, are systematically described. Experiments in three typical scenarios are conducted in the Gazebo simulation environment. The results show that in static obstacle environments, the proposed method achieves a navigation success rate of 96%, with a path length reduction of 6.2% compared to traditional methods; in dynamic obstacle environments, the success rate remains above 92%, with an average collision distance increase of 17.4%; and in moving target point scenarios, task completion time is reduced by 15.3%. Finally, key issues and solutions in practical applications, such as Sim2Real transfer and multi-machine collaboration, are discussed.

KEYWORDS
Reinforcement learning; Unmanned vehicle; Path planning; Motion control; Engineering application

1 Introduction

1.1 Research Background and Significance

Mobile robots have become indispensable intelligent equipment in modern society, and are widely used in logistics distribution, security inspection, medical assistance, agricultural production and other fields. As an important branch of mobile robots, the core capability of unmanned vehicles lies in their ability to autonomously complete environmental perception, path planning and motion control without human intervention, and achieve safe and reliable navigation from the starting point to the target point. A complete autonomous navigation system usually includes subsystems such as environmental perception, localization and mapping, path planning and motion control. Among them, path planning solves the problem of "where to go", and motion control solves the problem of "how to go". The two are coupled with each other and jointly determine the quality of navigation performance.

As application scenarios become more dynamic and complex, unmanned vehicles need to operate reliably in environments such as densely populated warehouses, bustling parks, and complex terrain. Traditional path planning and motion control methods, such as the A* algorithm, dynamic window method, and pure tracking control, while stable in structured environments, have inherent limitations such as poor adaptability and reliance on precise models when facing complex situations like dynamic obstacles and uncertain disturbances.

1.2 Current Status of Domestic and International Research

Mobile robot path planning methods can be summarized into five categories: graph search-based methods (such as Dijkstra's algorithm and A*), sampling-based methods (such as RRT), gradient-based methods (such as artificial potential field method), biomimetic intelligent algorithms (such as genetic algorithms and particle swarm optimization), and learning-based methods. Traditional methods such as the A* algorithm guarantee path optimality through heuristic search and are widely used in static environments, but cannot cope with dynamic obstacles; local planning methods such as DWA have strong real-time performance, but are prone to getting trapped in local optima.

In recent years, breakthroughs in artificial intelligence technology have opened up new avenues for autonomous navigation of unmanned vehicles. Deep reinforcement learning combines the perception capabilities of deep learning with the decision-making capabilities of reinforcement learning, enabling intelligent agents to learn complex control strategies through interaction with the environment. Aradi's review points out that DRL has been successfully applied in tasks such as car following, lane keeping, trajectory tracking, lane changing, and driving in dense traffic. Haider et al.'s review systematically summarizes DRL-based mobile robot navigation and control algorithms, covering subsystems such as perception, mapping, localization, and motion planning.

At the engineering application level, existing research has combined traditional planning methods with reinforcement learning. Salehi et al. proposed a framework integrating A* path planning and deep reinforcement learning, achieving

precise control of microrobots in environments containing both static and dynamic obstacles. A research team at Nanyang Technological University employed a multi-stage reinforcement learning method to achieve target search and navigation for a heterogeneous robot system in a maze-like mine environment, achieving a mission success rate of 89.1% for unmanned ground vehicles.

1.3 Existing Problems and Positioning of This Paper

Although deep reinforcement learning has shown great potential in unmanned vehicle navigation, from an engineering application perspective, it still faces the following challenges: (1) low training sample efficiency and high cost of training physical robots; (2) reward function design relies on experience, and unreasonable rewards lead to slow policy convergence; (3) Sim2Real transfer is difficult, and the performance of models trained in simulations degrades in real environments; (4) lack of engineering system design methods, and existing research focuses more on algorithm innovation and pays insufficient attention to application implementation.

This paper, from an engineering application perspective, focuses on path planning and motion control of unmanned vehicles based on reinforcement learning. It elaborates on the system design method, key points of engineering implementation, and analysis of application effects, aiming to provide a reference technical solution for engineering technicians in related fields.

2 System Overall Design

2.1 System Architecture

The unmanned vehicle navigation system proposed in this paper adopts a modular design, and the overall architecture is shown in Figure 1. The system consists of four parts: a global planning layer, a local planning and control layer, an environmental perception layer, and a bottom-level execution layer.

Figure 1 Overall Architecture Diagram of Unmanned Vehicle Navigation System

The environmental perception layer integrates multiple sensors such as LiDAR, odometer, and IMU. LiDAR is used for obstacle detection (10Hz, detection range 5m), odometer provides pose estimation, and IMU supplements angular velocity information. The global planning layer runs the A* algorithm based on the prior map to generate a global path point sequence from the starting point to the target point. The local planning and control layer is the core decision-making unit, employing a deep reinforcement learning agent to integrate real-time perception information and global path guidance, outputting linear velocity and angular velocity control commands. The bottom execution layer parses the control commands into motor drive signals to realize the movement of the vehicle.

2.2 Hardware and Software Platform Configuration

To facilitate engineering implementation and reproduction, Table 1 lists the main hardware and software configuration parameters of the system.

Table 1 System Hardware and Software Configuration

Components	Model / Configuration	Parameter Description
Car Platform	Differential Drive Wheel Chassis	Wheelbase 0.3m, wheel diameter 0.15m
LiDAR	2D LiDAR	10Hz, 360° scan, ranging range 0.1-5m
Odometer	Photoelectric Encoder	Resolution 1024 lines/revolutions
Main Control Unit	NVIDIA Jetson Xavier NX	384-core GPU, 8GB memory
Operating System	Ubuntu 20.04 + ROS Noetic	Robot Operating System
Simulation Environment	Gazebo 11	Supports Physics Engine and Sensor Simulation
Reinforcement Learning Framework	Stable-Baselines3	Provides SAC, PPO, and other algorithm implementations

2.3 Workflow

The system workflow is divided into two stages: offline training and online deployment:

Offline training phase: Construct various maps in the Gazebo simulation environment and randomly generate starting and target points. The reinforcement learning agent collects experience data by interacting with the simulation environment and stores it in the experience pool in the form of "state-action-reward-next state" quadruples, and periodically updates the network parameters. Save the policy model after training.

Online Deployment Stage: Load the pre-trained model, receive the guide points output by the global planning layer, combine with the real-time data of the LiDAR, obtain the control commands by the policy network forward calculation, and send them to the underlying actuators. Simultaneously, the operating status is continuously monitored, and a protection mechanism is triggered in case of abnormal situations (such as prolonged stagnation).

3 Reinforcement Learning System Design and Engineering Implementation

3.1 State Space Construction

State space design directly affects the perception capability of the policy. From an engineering application perspective, it is necessary to balance information completeness and computational efficiency. The state space designed in this paper contains three types of information, totaling 25 dimensions:

(1) The car's own state (10-dimensional): including the current position (x,y), orientation angle θ (normalized to [-π,π]), linear velocity v (normalized to [0,1]), angular velocity ω (normalized to [-1,1]), and the velocity changes Δv and Δω at previous and next time moments, used to perceive the motion trend.

(2) LiDAR perception (10-dimensional): Divide the 180° range in front evenly into 10 sectors, take the minimum distance value of each sector, and normalize it to [0,1] (1 represents no obstacle, 0 represents distance less than 0.2m). This dimensionality reduction process reduces the input dimension while ensuring obstacle perception capability.

(3) Navigation guidance information (5-dimensional): including the relative position of the target point (x_goal, y_goal) and the orientation difference Δθ, as well as the relative position of the global path guide point (x_guide, y_guide). The guide point is selected from the point 3m ahead of the car on the path generated by A*.

All state components are normalized before being input into the network, which is beneficial to network convergence.

3.2 Action Space Definition

The action space is defined as a two-dimensional continuous quantity: linear velocity $v \in [0, 1.0]$ m/s, angular velocity $\omega \in [-1.5, 1.5]$ rad/s. The network output layer uses the tanh activation function to map the output to the interval [-1,1], and then linearly transforms it to the actual control range.

In engineering implementation, the output instructions are smoothed and subjected to first-order low-pass processing to reduce drastic changes in control variables and protect the actuators: this can be adjusted according to actual debugging conditions.

3.3 Engineering Design of Reward Function

The reward function is the key to guiding the agent to learn the desired behavior. From an engineering application perspective, reward design should follow the principle of "sparse as the main approach, dense as the secondary approach, and easy to difficult from the beginning". The composite reward function designed in this paper includes the following components:

Table 2 Reward Function Component Design

Reward Components	Calculation Formula	Weights	Triggering Conditions
Target Achieved	R_arrival = 100	1.0	Distance to target < 0.3m
Collision Penalty	R_collision = -50	1.0	LiDAR reading < 0.2m
Distance Progress	R_dist = 5×(d_prev d_curr)	0.5	Calculation per Step
Path Following	R_guide = -0.2×d_guide	0.3	Calculation per Step
Smoothness Penalty	R_smooth = -0.1×|Δv| -0.05×|Δω|	0.1	Calculation per Step
Direction Guidance	R_heading = -0.1×|Δθ|	0.2	Calculation per Step

Where d_curr is the current distance from the target, d_prev is the distance at the previous moment, d_guide is the distance to the guide point, and Δθ is the deviation of the orientation from the target direction. The weight parameters have been determined through multiple rounds of debugging and can be adjusted according to specific application scenarios.

In engineering practice, the reward function needs to be adjusted progressively: initially, exploration rewards can be increased; in the middle stage, path following can be strengthened; and in the later stage, smoothness can be optimized. This paper adopts a segmented training strategy, adjusting the reward weight every 200,000 steps.

3.4 Network Structure and Training Parameters

The Soft Actor-Critic algorithm is adopted, whose maximum entropy framework encourages exploration and is suitable for continuous control tasks. The network structure is designed as follows:

Policy Network (Actor): Input layer 25-dimensional → Fully connected layer 256-dimensional → ReLU → Fully connected layer 128-dimensional → ReLU → Output layer 2-dimensional (mean) + 2-dimensional (log-standard deviation)

Value Network (Critic): Input layer 25+2=27-dimensional → Fully connected layer 256-dimensional → ReLU → Fully connected layer 128-dimensional → ReLU → Output layer 1-dimensional (Q-value)

To reduce overestimation, a dual-Q network structure is adopted. The network weights are initialized using Xavier, the optimizer is Adam, and the learning rate is 3e-4. The experience pool size is 1e6, the batch size is 256, the discount factor γ =0.99, and the target network update rate τ=0.005.

Training was performed on a single NVIDIA RTX 3080 GPU, with the model saved and evaluated every 1000 steps. The total training time steps were 2 million, taking approximately 8 hours.

4 Experiment and Result Analysis

4.1 Experimental Scenario Design

To comprehensively evaluate the method's performance, three typical application scenarios were built in Gazebo, each scenario containing 10 randomly generated maps.

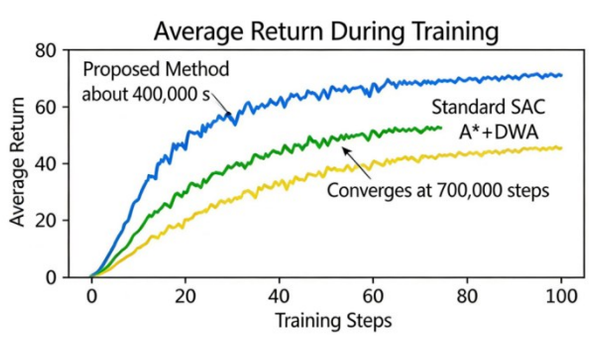
Scenario A (Static Obstacles): Simulates a warehouse environment, with 10-15 static obstacles (shelves, boxes) randomly placed in a 20m×20m area. The starting and target points are located at the area boundary, and the path must traverse the obstacle area.

Scenario B (Dynamic Obstacles): Adds 2-3 moving obstacles (simulating other AGVs or personnel) to Scenario A, with a moving speed of 0.3-0.5 m/s and random trajectories.

Scenario C (Moving Target): The target point moves randomly at a speed of 0.2 m/s, simulating dynamic tasks such as following a moving person.

Comparison methods selected: (1) Traditional A* +DWA combination (A* global planning, DWA local obstacle avoidance); (2) Standard SAC algorithm (without A* guidance); (3) The A*-SAC fusion method proposed in this paper.

4.2 Training Process Analysis



Algorithm	Convergence Steps	Final Average Return	Collision Rate During Training	Success Rate
Standard SAC	700,000	312.5	21.3%	73.5%
A*-SAC (Proposed)	400,000	365.8	12.6%	89.2%

Figure 2 Comparison of Average Reward Curves During Training

Figure 2 shows the average reward changes of the three methods during training. The proposed method converges at about 400,000 steps, while the standard SAC requires about 700,000 steps. A*+DWA is a traditional method that does not involve training. The convergence speed is improved by 42.9%, indicating that A* global guidance effectively compresses the exploration space.

As can be seen from Table 3, the proposed method is superior to the standard SAC in both training efficiency and learning effect. The collision rate during training is reduced by 8.7 percentage points, and the achievement power is increased by 15.7 percentage points. This means a shorter training cycle and a safer exploration process in practical

Table 3 Performance Comparison During Training

Algorithm	Number of Convergence Steps	Final Average Reward	Collision Rate During Training	Success Rate
Standard SAC	700,000	312.5	21.3%	73.5%
A*-SAC (proposed)	400,000	365.8	12.6%	89.2%

engineering.

4.3 Navigation Performance Comparison

Run 50 experiments in each of the three test scenarios and statistically analyze key performance indicators.

Table 4 Comparison of navigation performance in different scenarios

Scenario	Algorithm	Success rate (%)	Average path length (m)	Average time (s)	Minimum obstacle distance (m)
Scenario A	A*+DWA	94.0	24.8	35.2	0.28
(Static obstacles)	Standard SAC	78.0	25.3	38.6	0.21
Method in this paper	96.0	24.2	31.5	0.32	
Scene B	A*+DWA	82.0	26.1	41.3	0.23
(Dynamic obstacles)	Standard SAC	74.0	27.5	44.8	0.19
Method in this paper	92.0	25.4	36.7	0.27	
Scene C	A*+DWA	70.0	28.3	48.6	0.20
(Moving target)	Standard SAC	66.0	29.7	51.2	0.17
Method in this paper	86.0	27.2	42.3	0.24	

Figure 3 Comparison of typical running trajectories in Scene B

The following conclusions can be drawn from the data in Table 4:

(1) In static obstacle scenarios, both the proposed method and A*+DWA achieve a success rate of over 94%, but the proposed method has a shorter path (6.2% shorter), less time consumption (10.5% lower), and maintains a larger safety distance (14.3% larger), demonstrating the optimization capability of the learning strategy.

(2) In dynamic obstacle scenarios, the performance of A*+DWA drops significantly (success rate 82%), mainly because DWA's local planning relies on the current window information, making it difficult to predict the obstacle movement trend. The proposed method still maintains a success rate of 92%, indicating that the reinforcement learning strategy can learn dynamic obstacle avoidance rules from historical interactions.

(3) In the moving target scenario, the performance of all methods decreased, but the advantages of the method in this paper are more significant. The success rate is 16 percentage points higher than A*+DWA and 20 percentage points higher than the standard SAC, which shows the importance of global path guidance in complex dynamic tasks.

4.4 Analysis of Typical Failure Cases

Analyzing failure cases helps to improve system design. Statistics show that the failure cases of the method in this paper are mainly distributed in the following situations:

(1) Stuck in narrow passages (accounting for 38%): When the width of the passage is slightly greater than the width of the vehicle body, the strategy tends to be overly conservative, resulting in stagnation.

(2) Fast-moving obstacles (accounting for 32%): When the obstacle speed exceeds 0.8m/s, the obstacle avoidance reaction time is insufficient.

(3) Frequent turning back of the target point (accounting for 30%): When the trajectory of the moving target makes a sharp turn, the tracking lag leads to the loss of the target.

To address the above problems, the following engineering optimization measures can be taken: increase training samples in narrow passages; introduce an obstacle velocity estimation module; and adopt a target motion prediction model.

5 Key Issues in Engineering Applications

5.1 Sim2Real Migration

Models trained in simulation environments often face performance degradation when directly deployed to real platforms. The main reasons include: (1) differences between simulation and reality in dynamics; (2) inaccurate sensor noise models; and (3) visual differences in ambient lighting, texture, etc.

Commonly used transfer strategies in engineering practice include:

Domain randomization: Randomize physical parameters (mass, friction coefficient), sensor noise, lighting conditions, etc. in the simulation to allow the model to see more changes and enhance its generalization ability. In this paper, ±5% random noise is added to the lidar ranging during training, and the control response delay is randomized from 0 to 100ms.

System identification: collect motion data from the real platform, identify dynamic parameters, and calibrate the simulation model. By comparing the step response curves of reality and simulation, adjust the tire friction model and motor response characteristics in the simulation.

Progressive fine-tuning: The model is pre-trained in the simulation and then fine-tuned with small batch data on the real platform. Fine-tuning employs a small learning rate (1e-4) and safety constraints (limiting maximum speed) to ensure safe exploration.

5.2 Multi-machine Collaboration and Communication Constraints

In warehouse multi-AGV collaborative scenarios, reinforcement learning faces challenges such as limited communication bandwidth and localized observation. Engineering solutions include:

Distributed independent learning: each robot maintains an independent policy network and only shares global reward signals, reducing communication requirements.

Communication triggering mechanism: State information is exchanged only when the robot spacing is less than a threshold (e. g., 3m) or when there is a possibility of conflict in the predicted trajectory, reducing communication overhead by more than 60%.

Centralized training and distributed execution: global information is used during training, and only local observations are relied upon during execution. This approach ensures both training effectiveness and real-time performance.

5.3 Security and Reliability Assurance

The black-box nature of reinforcement learning strategies poses a challenge to security. Multi-layered security mechanisms need to be established in engineering deployment:

Level 1 Protection (Hardware Layer): Installing physical protection devices such as emergency stop switches, mechanical limit switches, and anti-collision strips as the last line of defense.

Secondary protection (monitoring layer): Real-time monitoring of state variables; switching to safety mode when the following conditions are triggered: minimum LiDAR distance < 0.15m; continuous stagnation time > 5s; attitude angle exceeding the normal range.

Tertiary protection (decision layer): A safety verification module is added after the policy output, adopting a "teacher-student" architecture: the traditional controller acts as the "teacher" to generate safe actions, and the RL policy acts as the "student" to optimize performance under the teacher's constraints. The actual output is a weighted average of the RL action and the teacher's action, with the weights dynamically adjusted based on safety risk assessment.

6 Conclusions and Outlook

6.1 Work Summary

This paper systematically studies the path planning and motion control methods for unmanned vehicles based on reinforcement learning from an engineering application perspective. The main work includes: (1) designing a system architecture that integrates A* global planning and reinforcement learning local control, and clarifying the functions and interfaces of each module; (2) explaining the key points of engineering implementation such as state space construction, reward function design, and network structure configuration; (3) conducting experiments on three typical scenarios in the Gazebo simulation environment to verify the effectiveness of the method; and (4) exploring key issues and solutions in engineering applications such as Sim2Real migration, multi-machine collaboration, and security assurance.

Experimental results show that the proposed method outperforms traditional methods and standard reinforcement learning methods in three scenarios: static environment, dynamic obstacles, and moving targets, achieving navigation success rates of 96%, 92%, and 86%, respectively, demonstrating good application potential.

6.2 Future Prospects

Further research can be deepened in the following directions:

(1) Multimodal perception fusion: Combine visual information to improve the understanding of complex environments, such as identifying obstacle types and predicting motion intentions.

(2) Lifelong learning and incremental update: Build a continuously evolving control system to adapt to long-term operation needs such as equipment aging and environmental changes.

(3) Hierarchical reinforcement learning: Decompose complex tasks into sub-tasks such as navigation, obstacle avoidance, and interaction, train strategies separately and then combine them to improve training efficiency.

(4) Empowering Large Language Models: Explore the application of LLM in natural language instruction

understanding, task planning, and exception handling to improve the naturalness of human-computer interaction.

For engineering and technical personnel, it is recommended to start with the simulation environment and prioritize the verification of algorithm feasibility; select appropriate algorithm variants according to specific scenarios; pay attention to the progressive debugging of the reward function and the design of security mechanisms; and fully evaluate the Sim2Real gap when migrating to the real platform and adopt corresponding migration strategies.

About the Author

Hongxiang Chen, Undergraduate, Research Direction: Applied Physics.

References

[1] Katona et al. A survey on autonomous navigation for mobile robots: From traditional techniques to deep learning and large language models. Springer, 2025.

[2] Reinforcement learning drives the future: Technological breakthroughs and application practices of intelligent robot control. Baidu Developer Center, 2025.

[3] Ben Elallid B, et al. A Reinforcement Learning Based Approach for Controlling Autonomous Vehicles in Complex Scenarios. CARLA simulator, 2025.

[4] Haider Z, et al. Exploring reinforcement learning techniques in the realm of mobile robotics. International Journal of Automation and Control, 2024, 18(6): 655-697.

[5] Aradi S. Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles. IEEE Transactions, 2025.

[6] Chen Y, Xiao J. Target Search and Navigation in Heterogeneous Robot Systems with Deep Reinforcement Learning. Machine Intelligence Research, 2025, 22(1): 79-90.

[7] High-precision deep reinforcement learning trajectory tracking control of unmanned vehicles. Journal of Engineering Science, 2026.

[8] Haider Z, et al. Exploring reinforcement learning techniques in the realm of mobile robotics. International Journal of Automation and Control, 2024.