OPTIMIZATIONS IN APPLIED MACHINE LEARNING
Research Article* | *Volume 4* | *Issue 4* | *Dec 2024
Received: 14 Nov 2024 | Revised: 19 Nov 2024
Accepted: 1 Dec 2024 | Published Online: 4 December 2024

Deep Learning-Based Optimization for Mobile Robotic Delivery Systems

Diwei Zhu[1], Yunxiang Gan[2], Xiaoyang Chen[3*]

[1]New York University, New York City, United States;

[2]Moloco, Redwood City, CA 94063, United States;

[3*]Radiawave Co., Ltd., Shen Zhen, China

*Corresponding Author, Email: chenxiaoyang@radiawave.com

Abstract: In today's logistics and delivery landscape, mobile robot delivery systems have attracted considerable attention due to their efficiency and adaptability. Nevertheless, current robotic delivery solutions encounter various obstacles in complex and dynamically changing environments. Traditional algorithms, for instance, struggle with processing high-dimensional and unstructured data, resulting in inefficient adaptation to real-time environmental changes, which compromises accuracy and efficiency in path planning and task execution. Moreover, the lack of robust perception and decision-making mechanisms limits the robots' ability to handle intricate scenarios and fluctuating delivery demands. To tackle these challenges, this paper proposes an optimization approach for mobile robot delivery systems that leverages deep learning. The study initially integrates a spatial attention mechanism within the model, enabling the robot to focus on critical environmental regions and dynamically adjust attention points, thus enhancing obstacle recognition and avoidance in complex settings, which improves navigation accuracy and path planning. Furthermore, the Deep Deterministic Policy Gradient (DDPG) algorithm is utilized to optimize policies, supporting efficient learning in high-dimensional continuous spaces and empowering robots to acquire effective delivery strategies in challenging environments. Finally, an end-to-end optimization approach allows the system to convert sensor inputs directly into control commands, reducing intermediate complexity and minimizing error accumulation, thereby streamlining the system's structure. Experimental results confirm that the proposed method substantially boosts delivery system performance, excelling in key metrics like path planning accuracy, task efficiency, and system robustness. The successful integration of the spatial attention mechanism with the deep policy gradient algorithm demonstrates a valuable new approach for advancing future robot delivery system optimizations.

Keywords: *Delivery system; deep learning; spatial attention mechanism; DDPG algorithm; end-to-end optimization; path planning*

1 Introduction

With the popularization of e-commerce and online shopping, there has been a rapid increase in logistics demand, driving the rapid development of delivery systems. These systems not only need

to efficiently manage and transport goods but also ensure timely and accurate delivery to consumers. Efficient logistics and delivery systems can not only reduce operating costs but also enhance customer satisfaction, thus improving the competitiveness of businesses (Gomes et al., 2023). As an innovative technology in the logistics industry, robot mobile delivery systems demonstrate tremendous application potential. Through automation and intelligence, robots can efficiently execute delivery tasks, reduce manual intervention, and improve work efficiency. At the same time, robots possess flexible path planning and navigation capabilities, enabling them to autonomously complete delivery tasks in different environments, further enhancing the adaptability and flexibility of the system. Especially in scenarios such as warehousing and urban delivery, robot delivery systems can significantly improve overall operational efficiency. Despite the significant advantages of robot delivery systems, they still face many challenges in complex and dynamic environments (Jiang & Huang, 2022). Traditional path planning algorithms exhibit increased computational complexity when dealing with high-dimensional and unstructured data. These algorithms may need to process a large number of nodes and edges in complex environments, resulting in significantly increased computation time, making it difficult to meet real-time requirements (Jones et al., 2023). In dynamically changing environments, it is difficult to adapt quickly to changes in the environment. Each time the environment changes, the path needs to be recalculated from scratch, leading to inefficiencies (Yan et al., 2020). For example, when new obstacles or blocked paths appear, it is necessary to recalculate from scratch, which cannot efficiently update existing paths. Furthermore, traditional algorithms typically rely on static maps and preset paths, lacking dynamic adjustment capabilities. Even if a feasible path is found, it is difficult to guarantee that it is the globally optimal path, especially in complex environments, where the algorithm may only find a local optimal solution and fail to discover a better global path (Chang et al., 2021). Additionally, these algorithms perform poorly in handling dynamic obstacles, typically based on predefined static maps, lacking real-time perception and processing capabilities for dynamic obstacles. When robots encounter moving obstacles in complex environments, they may not be able to adjust the path in time, leading to collisions or path planning failures (Wang et al., 2021).

Deep learning continues to evolve, and the application of robot mobile delivery systems in modern logistics is becoming increasingly widespread. Faced with the many challenges of existing delivery systems in complex and dynamically changing environments, researchers are constantly exploring new methods and technologies. Spatial attention mechanism, as an advanced technology, significantly enhances the perception ability of robots by focusing on key areas in the environment (Zhou et al., 2022). This mechanism can dynamically adjust the focus, enabling robots to better identify and avoid obstacles in complex environments, thereby improving navigation and path planning accuracy. However, the spatial attention mechanism also faces certain challenges in the implementation process, including how to efficiently calculate attention weights and its application in high-dimensional data. Deep Deterministic Policy Gradient (DDPG) algorithm is a combination of policy gradient methods and deep learning algorithms, suitable for reinforcement learning tasks in continuous action spaces (Wu & Li, 2020). The DDPG algorithm models policies through deep neural networks and optimizes them using policy gradient methods, enabling efficient learning in high-dimensional continuous spaces. Although DDPG performs well in policy optimization in complex environments, it also has some limitations, such as stability and convergence speed issues in high-noise environments (Wang et al., 2020). End-to-end optimization is a holistic optimization solution from input to output, aimed at reducing the complexity of intermediate links and error accumulation. The design of end-to-end optimization allows the system to directly input sensor data to output control commands, not only simplifying the system structure but also improving overall response speed and reliability. However, end-to-end optimization also faces some challenges, such as the complexity of model training and the demand for large-scale data (Chen et al., 2023). To address these issues, this paper proposes a robot mobile delivery system optimization

method combining spatial attention mechanism, Deep Deterministic Policy Gradient algorithm, and end-to-end optimization, aiming to solve the main problems faced by existing systems in complex and dynamic environments. By introducing the spatial attention mechanism, we enhance the system's perception ability to dynamically changing environments; using the DDPG algorithm for policy optimization improves the efficiency and accuracy of path selection and task execution; through an end-to-end optimization solution, overall performance improvement from input to output is achieved. The model combines advanced deep learning computations to provide new ideas and methods for future optimization of robot delivery systems.

The structure of this paper is arranged as follows: Part 1 introduces the background, motivation, and objectives of the research, emphasizing the importance of robot mobile delivery systems in modern logistics and the main challenges faced by existing systems in complex and dynamic environments. Part 2 introduces related work, including existing methods for optimizing robot delivery systems. A detailed review of path planning and navigation technology, as well as the application of deep reinforcement learning in robot control, is provided. Part 3 describes in detail the proposed method, including the implementation of spatial attention mechanism and DDPG algorithm. This section explains how to apply the spatial attention mechanism to robot perception, enhancing its adaptability to complex environments, and optimize delivery strategies through the DDPG algorithm to achieve efficient path planning and task execution. Part 4 describes the design process of the experiments, the selection of datasets, the setting of evaluation metrics, and the analysis of experimental results. The effectiveness of the proposed method in improving the overall performance of the delivery system is verified through experiments. Part 5 is the conclusion and future work, summarizing the main contributions and research results of this paper, and proposing future research directions and improvement suggestions, providing references for further optimization of robot mobile delivery systems.

2 Relevant work

Path planning and navigation are core components of robot mobile delivery systems. Traditional path planning algorithms include the A* algorithm (Erke et al., 2020), Dijkstra's algorithm (Mirahadi & McCabe, 2021), and the Rapidly-exploring Random Tree (RRT) algorithm (Wu et al., 2021). These algorithms perform well in static environments, capable of finding the shortest path from the starting point to the target point. However, they have limitations in complex and dynamic environments. For example, A* and Dijkstra's algorithms exhibit high computational complexity when handling high-dimensional and unstructured data, making real-time applications challenging. Additionally, these algorithms typically rely on predefined static maps, lacking adaptability to environmental changes. To overcome these issues, researchers have proposed various improved methods. For instance, the Real-Time A* (RTA*) algorithm achieves real-time performance by limiting search depth and computation time per decision, considering only a limited number of future steps at each stage, making it suitable for resource-constrained embedded systems and robot navigation (Zhang et al., 2020). Lifelong Planning A* (LPA*) can quickly update the shortest path when the graph structure changes, updating only the affected parts when the environment changes, thereby improving path update efficiency (Segato et al., 2021). Focused D*, a further optimization of the D* algorithm, enhances efficiency and dynamic adaptability by concentrating the search on regions most likely to affect the path during planning (Qadir et al., 2021). However, these methods still face challenges in handling dynamic obstacles and high-dimensional data. In recent years, deep learning-based path planning methods have emerged, significantly improving the efficiency and accuracy of path planning by learning strategies in complex environments. For example, Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG) reinforcement learning algorithms have been successfully applied to robot navigation tasks.

Deep Reinforcement Learning (DRL) combines the advantages of deep learning and reinforcement learning, suitable for control tasks in high-dimensional continuous spaces. In the field of robot control, DRL has been widely applied to path planning, navigation, and task execution. DQN combines Q-learning with deep neural networks to address high-dimensional state space problems by approximating the Q-value function with neural networks, enabling effective learning in discrete action spaces. In warehouse automation, mobile robots use the DQN algorithm to achieve autonomous navigation, avoiding collisions and efficiently completing tasks (Lee & Yusuf, 2022). However, DQN performs poorly in continuous action spaces, a limitation addressed by DDPG. By introducing policy and value networks, DDPG can optimize policies in high-dimensional continuous action spaces. In the autonomous driving field, deep learning technologies are widely applied in perception, decision-making, and control systems. Perception systems typically use Convolutional Neural Networks (CNNs) to process sensor data from cameras, LiDAR, and radar. Tesla's autonomous driving system uses deep learning models to recognize road signs, lane markings, and pedestrians, making driving decisions accordingly. Waymo utilizes deep learning models for environmental perception and dynamic obstacle detection, ensuring vehicle safety (Gupta et al., 2021). Significant progress has also been made in controlling humanoid robots with deep learning. Boston Dynamics' Atlas robot uses deep reinforcement learning algorithms to perform complex actions and behaviors such as running, jumping, and balancing. By learning from sensor data, Atlas can adjust its action strategies to cope with various terrains and environmental changes (Neri & Dinarama, 2024).

Additionally, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have notable advantages in handling time series data, making them suitable for path planning tasks in dynamic environments. By remembering and predicting environmental changes (Torres et al., 2021), RNNs and LSTMs can help mobile robots control paths more effectively in dynamic environments. LSTM networks are commonly used to predict the motion trajectories of dynamic obstacles and adjust the robot's path to avoid collisions. LSTM is also employed in multi-robot systems for task scheduling and path optimization, improving system coordination and efficiency. Despite addressing the short-term memory issues of RNNs to some extent, LSTM can still suffer from information loss or forgetting over particularly long-time sequences. Additionally, the network's complex structure with numerous parameters can result in high computational overhead during inference and prediction stages. For real-time path planning tasks, excessive computational complexity can lead to response delays, failing to meet real-time requirements. Generative Adversarial Networks (GANs) can generate realistic environmental simulation data through adversarial training between the generator and the discriminator, assisting in the training of path planning algorithms (Zhao et al., 2022). GANs can be used to create virtual training environments, allowing mobile robots to learn and optimize their path planning strategies in simulated settings, thus reducing training costs and risks in real environments. In the field of autonomous driving, GANs are employed to generate various driving scenarios, aiding the training and testing of autonomous driving systems in diverse complex situations. However, the training process of GANs is often unstable and prone to mode collapse, where the generator produces only a limited variety of samples instead of covering the entire data distribution. This phenomenon can lead to a lack of diversity in the generated data, affecting the model's generalization ability and practical application effectiveness. Spatial attention mechanisms, which can dynamically adjust focus areas, are widely used in computer vision and natural language processing. In robot perception and control, spatial attention mechanisms significantly enhance perception and decision-making capabilities by focusing on key areas of the environment. Models combining CNNs and attention mechanisms can better identify obstacles and navigation targets in complex environments, improving path planning accuracy. Although spatial attention mechanisms can enhance perception capabilities in static environments, their adaptability and real-time performance

may still be insufficient in highly dynamic and rapidly changing environments, potentially failing to respond in real time to quickly changing obstacles and paths in practical applications. Therefore, this paper proposes a novel robot mobile delivery system optimization scheme combining spatial attention mechanisms, Deep Deterministic Policy Gradient (DDPG) algorithms, and end-to-end optimization methods. By introducing spatial attention mechanisms, the perception capability and decision accuracy of the robot are enhanced through dynamically adjusting the environmental focus areas. Additionally, the DDPG algorithm is adopted for optimizing delivery strategies. DDPG, combining policy gradient methods and deep learning techniques, achieves efficient learning in high-dimensional continuous action spaces through the mutual optimization of policy and value networks. Finally, this paper implements an end-to-end optimization scheme, directly inputting sensor data into control command output, simplifying the system structure, reducing the complexity and error accumulation of intermediate links, not only improving the system's response speed and reliability but also enhancing overall performance.

3 Method

Figure 1 shows the overall algorithm architecture of the robot delivery system used in this article.

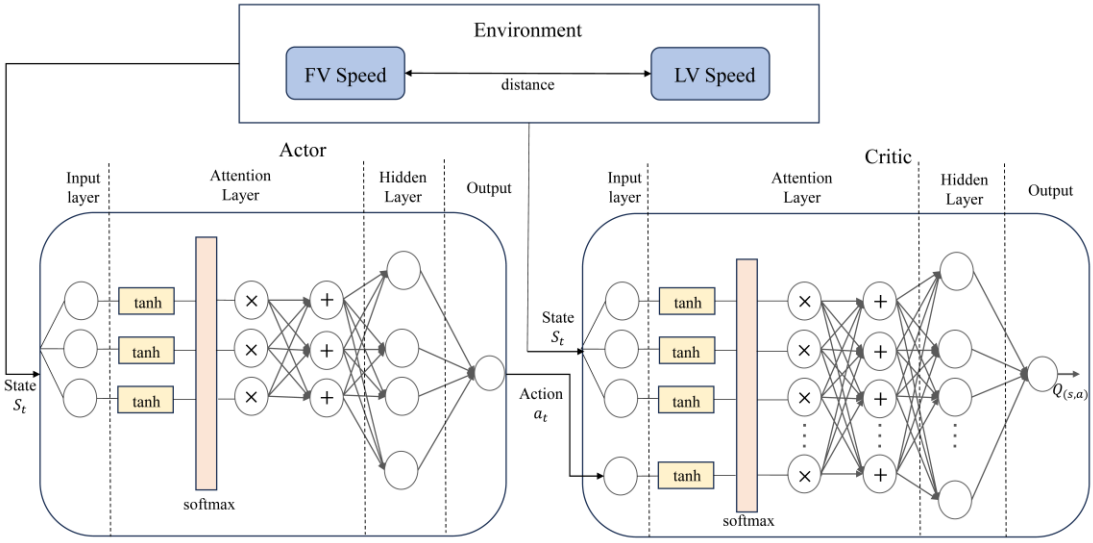

Figure 1. Overall algorithm architecture.

3.1 Spatial Attention Mechanism

The application of spatial attention mechanism in robot mobile delivery systems aims to enhance the perception capabilities of robots, enabling them to navigate and plan paths more accurately in complex environments. Its core lies in assigning different attention weights to different regions of the input feature map, focusing on key areas in the environment to improve the precision of perception and decision-making (Li et al., 2022). In this paper, the spatial attention mechanism is mainly divided into two steps: attention weight calculation and attention feature map generation. The architecture diagram of SAM is shown in Figure 2.

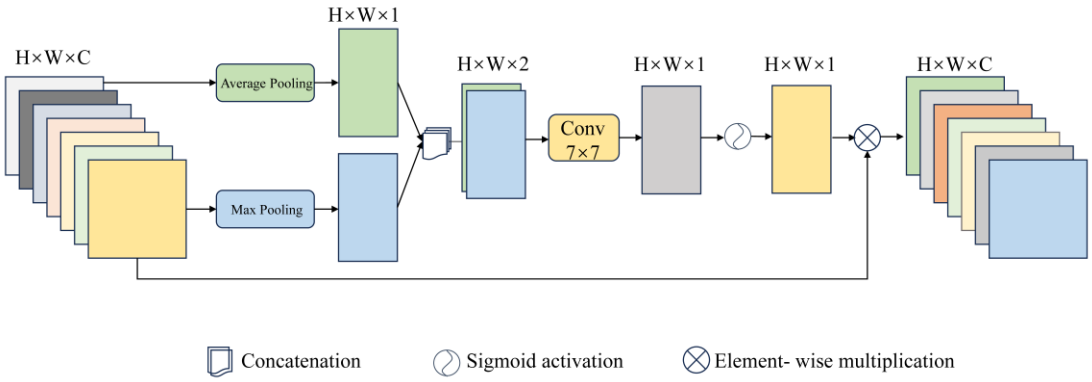

Figure 2. Structure diagram of SAM.

Firstly, for a given input feature map $F \in R^{C\times H\times W}$, where C, H, and W represent the number of channels, height, and width of the feature map, respectively, we need to calculate the attention weights for each spatial position. Representing the feature vector at each position of F as $f_{i,j} \in R^{C}$, where i and j represent the indices of height and width of the feature map, respectively, the calculation of attention weights can be achieved through a simple feedforward neural network, formalized as:

$$\alpha_{i,j} = \sigma(\mathbf{W}_{\alpha}\mathbf{f}_{i,j} + b_{\alpha}) \tag{1}$$

Here, $W_{\alpha} \in R^{\mathbb{1}\times C}$ and $b_{\alpha} \in R$ represent the weight matrix and bias term, respectively, and σ denotes the activation function (such as the sigmoid function). The computed $\alpha_{i,j}$ represents the attention weight at position (i,j). To ensure that the sum of all attention weights equals 1, normalization can be applied:

$$\tilde{\alpha}_{i,j} = \frac{\alpha_{i,j}}{\sum_{m=1}^{H}\sum_{n=1}^{W}\alpha_{m,n}} \tag{2}$$

With normalized attention weights, we can generate the attention feature map. The generation of the attention feature map is achieved by weighted summation of each position of the input feature map, formalized as:

$$\mathbf{F}_{\text{att}} = \sum_{i=1}^{H}\sum_{j=1}^{W}\tilde{\alpha}_{i,j}\mathbf{f}_{i,j} \tag{3}$$

Here, $F_{att} \in R^{C}$ represents the attention feature map, which integrates the feature representations with attention weights. The generated attention feature map F_{att} can be used for subsequent path planning and navigation decisions. In robot mobile delivery systems, the attention feature map serves as input to guide robots in making real-time decisions in complex environments. Specific applications include obstacle recognition, path selection, and adaptation to dynamic environments. The spatial attention mechanism significantly enhances the perception and decision-making capabilities of robots in complex environments, enabling them to perform mobile delivery tasks more efficiently.

3.2 DDPG Architecture

Deep Deterministic Policy Gradient (DDPG) is a reinforcement learning algorithm that combines policy gradient methods with deep learning, suitable for tasks in high-dimensional continuous action spaces. By using deep neural networks to approximate the policy and value functions, the DDPG algorithm achieves efficient learning in complex environments (Wang et al., 2022). DDPG integrates the advantages of Deep Q-Learning (DQN) and policy gradient methods, employing two deep neural networks: the policy network (Actor) and the value network (Critic) for decision making and evaluation, respectively. These networks are optimized jointly to continuously improve the policy in high-dimensional continuous action spaces. The architecture diagram of DDPG is shown in Figure 3

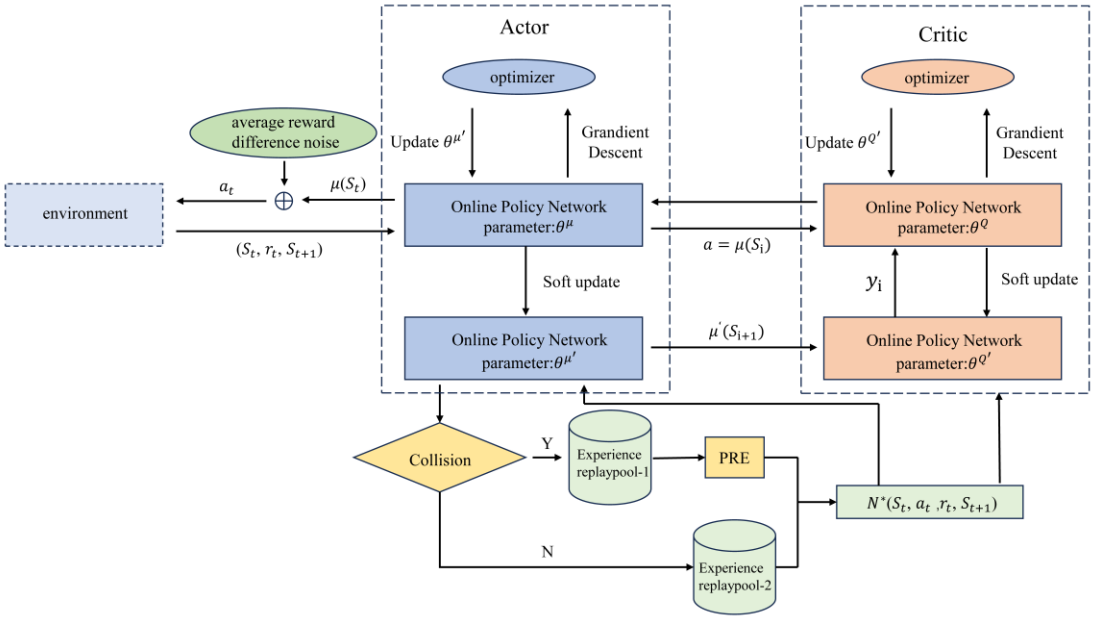

Figure 3. Structure diagram of DDPG.

The policy network $\mu(s|\theta^{\mu})$ takes the state s as input and outputs the corresponding action a. The parameters θ^{μ} of the policy network are optimized using policy gradients to maximize the expected cumulative reward for the actions chosen in a given state. The value network $Q(s,a|\theta^{Q})$ takes the state s and action a as inputs and outputs the corresponding state-action value (Q-value). The parameters θ^{Q} of the value network are optimized by minimizing the Temporal Difference (TD) error, which evaluates the effectiveness of the policy.

Initialization of Networks and Experience Replay Buffer: Initialize the policy network $\mu(s|\theta^{\mu})$ and the value network $Q(s,a|\theta^{Q})$, as well as their target networks $\mu'(s|\theta^{\mu'})$ and $Q'(s,a|\theta^{Q'})$. The target networks are used to stabilize the training process. Initialize the experience replay buffer $\mathcal{D}$ to store the agent's experiences. Experience Collection: Execute actions in the environment based on the current policy network, selecting actions $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$, where $\mathcal{N}_t$ is the exploration noise. After executing an action, observe the next state s_{t+1} and reward r_t, and store the experience (s_t, a_t, r_t, s_{t+1}) in the experience replay buffer $\mathcal{D}$.

Experience Replay: Compute the expected cumulative reward for the future state using the target value network and target policy network. Randomly sample a minibatch (s_i, a_i, r_i, s_{i+1}) from the experience replay buffer. Calculate the target Q-value y_i:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'}) \tag{4}$$

where γ is the discount factor representing the decay rate of future rewards. Update the Value Network: Minimize the Temporal Difference error to update the parameters θ^Q of the value network:

$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i \mid \theta^Q))^2 \tag{5}$$

where N is the size of the minibatch. This loss function measures the error between the current value network's predicted Q-values and the target Q-values, guiding the parameter updates of the value network.

Update the Policy Network: Using policy gradient methods, the update direction of the policy network parameters is determined by the gradient of the Q-values from the value network and the gradient of the actions from the policy network. Update the policy network parameters θ^μ via policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a \mid \theta^Q) \mid_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s \mid \theta^\mu) \mid_{s=s_i} \tag{6}$$

Soft Update of Target Networks: Soft update the parameters of the target policy network and target value network:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{7}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{8}$$

where $\tau \ll 1$ is the step size for the soft update.

3.3 End-to-End Optimization

End-to-end optimization reduces the complexity and error accumulation in intermediate stages. By directly learning the mapping from raw sensor inputs to final control commands, end-to-end optimization significantly improves response speed and reliability in robotic systems. This approach uses a unified neural network model to directly map sensor inputs to control commands, simplifying the system structure and enhancing overall performance (Zhao et al., 2023). The architecture diagram of end-to-end optimization is shown in Figure 4.

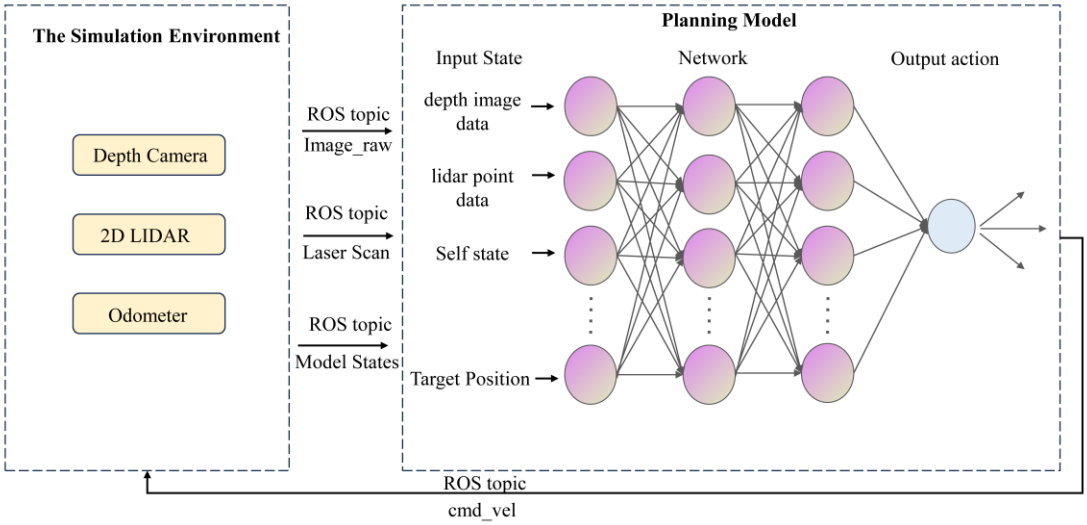

Figure 4. Structure diagram of end-to-end optimization.

In end-to-end optimization, neural network models typically include convolutional layers (for processing image data), recurrent layers (for handling time-series data), and fully connected layers (for generating control commands). The input layer receives data from sensors such as camera images and LiDAR point clouds. Convolutional layers extract high-level features from the input data, capturing key information from the environment. Recurrent layers handle time-series data, capturing changes in the dynamic environment. Fully connected layers map the extracted features to specific control commands, such as the robot's speed and direction.

The training process for end-to-end optimization is conducted through either reinforcement learning or supervised learning. The state s_t is defined as the robot's sensory information at time step t, such as camera images or LiDAR point clouds. The action a_t is the control command at time step t. The reward r_t is the reward obtained after the robot executes the action at time step t, such as the reduction in distance to the target point. The loss function measures the discrepancy between the predicted control commands and the actual desired commands. Common loss functions include Mean Squared Error (MSE) and Policy Gradient Loss. The MSE loss function is given by:

$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(a_i - \hat{a}_i)^2 \tag{9}$$

where N is the number of samples, a_i is the actual control command, and $\hat{a}_i$ is the predicted control command. The policy gradient loss is given by:

$$L_{\text{PG}} = -\mathbb{E}[R_t \log \pi(a_t \mid s_t)] \tag{10}$$

where R_t is the cumulative reward and $\pi(a_t|s_t)$ is the policy for selecting action a_t in state s_t. Gradient descent is used to optimize the neural network parameters by minimizing the loss function. The gradient descent update rule is:

$$\theta \leftarrow \theta - \alpha \nabla_\theta L \tag{11}$$

where θ represents the network parameters, α is the learning rate, and $\nabla_\theta L$ is the gradient of the loss function. End-to-end optimization enables overall optimization from input to output, significantly improving the response speed and reliability of robotic systems. By directly generating control commands from sensor inputs, neural networks simplify the system structure, reducing the complexity and error accumulation in intermediate stages.

4 Experiment

The experimental flow chart of this paper is shown in Figure 5.

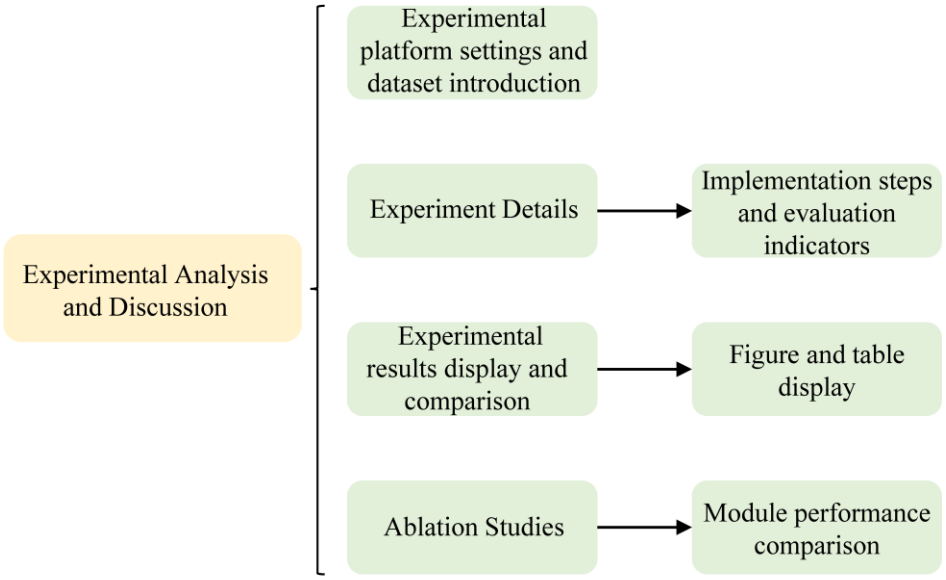

Figure 5. Experimental flowchart.

4.1 Experimental Environment

In terms of hardware environment, our computing platform is configured with an Intel Core i9-10900K CPU, suitable for parallel computing and handling complex tasks. The GPU is an NVIDIA GeForce RTX 3090, supporting accelerated training and inference of deep learning models. Additionally, it has 256GB of memory, supporting large-scale data processing and model training. As for the robot platform, we utilize the TurtleBot 3, an open-source platform designed for robot research and education, with support for ROS (Robot Operating System). Sensor configurations include the Intel RealSense D435i depth camera and Hokuyo URG-04LX laser rangefinder, which are high-precision sensors used for environment perception and navigation. This experiment is conducted on the Ubuntu 20.04 LTS operating system, known for its stability and wide range of applications, making it particularly suitable for machine learning and robotics development. Python 3.8 is chosen as the programming language, which is a mainstream language in the fields of machine learning and deep learning, with rich library and tool support.

4.2 Experimental Data

- KITTI Dataset

The KITTI dataset is a benchmark dataset widely used in autonomous driving and robotics research. It was jointly created by the Karlsruhe Institute of Technology and the Toyota Technological

Institute at Chicago, containing high-quality images and LiDAR data from real driving environments. The dataset is collected by mounting cameras and LiDAR sensors on the top of cars, covering various urban, rural, and highway scenes. Content of the KITTI dataset includes color and grayscale images, 3D point cloud data, GPS information, and IMU readings. Its diversity and richness make it an essential resource for evaluating the performance of visual deep learning and path planning algorithms. Researchers can utilize the KITTI dataset for tasks such as object detection, semantic segmentation, 3D reconstruction, path planning, and autonomous driving. By testing in complex and dynamic environments, the KITTI dataset provides a solid foundation for validating the robustness and effectiveness of algorithms.

- COCO Dataset

The COCO (Common Objects in Context) dataset is a widely used benchmark dataset for computer vision research, created by Microsoft. It consists of over 200,000 high-quality images, annotated with more than 2.5 million instance objects spanning 80 common object categories. Each image is annotated not only with bounding boxes for objects but also detailed segmentation masks, keypoints, and image-level labels. These annotations make the COCO dataset widely applicable in tasks such as object detection, semantic segmentation, instance segmentation, human pose estimation, and image captioning. The images in the COCO dataset are collected from various real-life scenarios, including indoor and outdoor environments, featuring rich background information and complex object layouts, providing an ideal resource for training and testing algorithms in diverse and complex scenes. Its diversity and high-quality annotations make the COCO dataset an important tool for evaluating and driving the development of computer vision algorithms.

- RobotCar Dataset

The RobotCar dataset is a benchmark dataset for autonomous driving and robotics research created by the Mobile Robotics Group at the University of Oxford. This dataset comprises rich data collected under various time, weather, and seasonal conditions in the city of Oxford, covering diverse urban driving environments. Data collection in the RobotCar dataset is facilitated through multiple sensors mounted on vehicles, including stereo cameras, LiDAR, GPS, and Inertial Measurement Units (IMU). These sensors provide high-resolution images, 3D point cloud data, precise location information, and vehicle motion data. The diversity and detailed annotations of the RobotCar dataset make it a crucial resource for evaluating and developing tasks such as autonomous driving systems, 3D reconstruction, path planning, and environment perception. Researchers can utilize this dataset for robustness testing across different weather and seasonal variations, validating algorithms' adaptability and stability under various environmental conditions.

- NuScenes Dataset

The NuScenes dataset, created by Motional, is an advanced benchmark dataset for autonomous driving research. It collects real-world data from complex urban environments in Boston and Singapore, covering various weather and lighting conditions. The NuScenes dataset comprises data from multiple sensors, including panoramic images from six cameras, point cloud data from five LiDARs, millimeter-wave radar, GPS, and Inertial Measurement Units (IMU). These sensors provide comprehensive environmental perception information, aiding researchers in studying tasks such as multimodal perception, 3D object detection, tracking, semantic segmentation, and scene understanding. In addition to high-resolution sensor data, the NuScenes dataset also includes detailed annotation information such as object bounding boxes, category labels, and trajectories. These rich annotations make NuScenes an essential resource for evaluating the robustness and performance of autonomous driving algorithms.

4.3 Evaluation Metrics

- Accuracy

Accuracy represents the proportion of correct predictions made by a model out of all predictions. It is an intuitive metric for assessing the overall correctness of a model, particularly useful for evaluating the performance of tasks such as robot perception and environmental understanding. The formula for accuracy is:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{12}$$

where TP is the number of instances in path planning where real obstacles are correctly detected. TN is the number of instances where non-existing obstacles are correctly recognized as non-existing. FP is the number of instances where non-existing obstacles are incorrectly detected as existing. FN is the number of instances where real obstacles are not detected.

- Precision：

Precision represents the proportion of samples predicted as positive that are actually positive. Precision reflects the accuracy of a model, and particularly in dealing with imbalanced datasets, precision is a crucial performance metric. The mathematical definition of precision is as follows:

$$Pr\,e\,cision = \frac{TP}{TP + FP} \tag{13}$$

- Recall：

Recall represents the proportion of all actual positive samples that are correctly predicted as positive. Recall reflects the detection capability of a model, particularly in cases where there are many missed detections. Recall evaluates the completeness of obstacle detection in the environment during robot path planning. High recall indicates that the robot can detect most of the actual obstacles, reducing missed detections and improving the safety of path planning. The mathematical definition of recall is as follows:

$$Re\,c\,all = \frac{TP}{TP + FN} \tag{14}$$

- F1-Score：

The F1 score is a comprehensive metric for evaluating the performance of classification models, combining both precision and recall. It provides a more complete assessment of classification problems in imbalanced datasets. In dynamic environments, the F1 score evaluates the robot's overall capability to handle real-time changes in environmental information. A high F1 score indicates that the robot can accurately identify newly appearing obstacles while minimizing missed detections, thereby improving navigation efficiency and safety. The formula is as follows:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{15}$$

4.4 Experimental Comparison and Analysis

In this section, we conduct a comprehensive comparison between six different path optimization algorithms and our proposed method. This evaluation utilizes four datasets: KITTI, COCO, RobotCar, and NuScenes. These datasets encompass various complex environments, including urban streets, indoor scenes, and challenging driving conditions. To thoroughly assess the

performance of each algorithm, we employ four key metrics: Accuracy, Precision, Recall, and F1 Score. We will comparatively analyze the strengths and weaknesses of each algorithm and discuss their applicability in different environments and tasks.

Table 1. Comparison of indicators of various models under KITTI Dataset and COCO Dataset.

Model	KITTI Dataset				COCO Dataset			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Zhang L et al. (Zhang et al., 2020a)	85.89	86.09	88.82	87.43	82.81	92.54	89.65	91.07
Aslan MF et al. (Aslan et al., 2022)	88.10	86.28	87.74	87.00	84.41	85.01	87.09	86.04
Lee DH et al. (Lee & Liu, 2023)	91.00	84.81	86.59	85.69	87.19	85.59	88.39	86.97
Gu Y et al. (Gu et al., 2023)	87.32	87.85	84.36	86.07	88.86	90.32	85.45	87.82
Huang R et al. (Huang et al., 2023)	90.60	85.38	85.39	85.38	87.12	88.26	87.89	88.07
Chen L et al. (Chen et al., 2022)	84.99	83.52	86.42	84.95	88.05	92.67	90.68	91.66
Ours	93.46	92.54	94.43	93.48	92.73	94.61	92.43	93.51

Table 1 presents the comparison results of six different path optimization algorithms and our proposed method on four key metrics (accuracy, precision, recall, and F1 score) across the KITTI and COCO datasets. It can be observed from the table that our proposed method performs excellently on both datasets, outperforming other methods across all metrics. Specifically, on the KITTI dataset, our method achieves an accuracy of 93.46%, precision of 92.54%, recall of 94.43%, and an F1 score of 93.48%; while on the COCO dataset, the accuracy is 92.73%, precision is 94.61%, recall is 92.43%, and F1 score is 93.51%. In comparison, other methods show varied performance across different metrics, but overall none surpasses our method, particularly in the comprehensive metric of F1 score. This result indicates that our proposed method holds significant performance advantages in path planning tasks across diverse and complex environments, particularly in enhancing the accuracy and completeness of detection. At the same time, we show the visualization of various indicator comparisons in Figure 6.

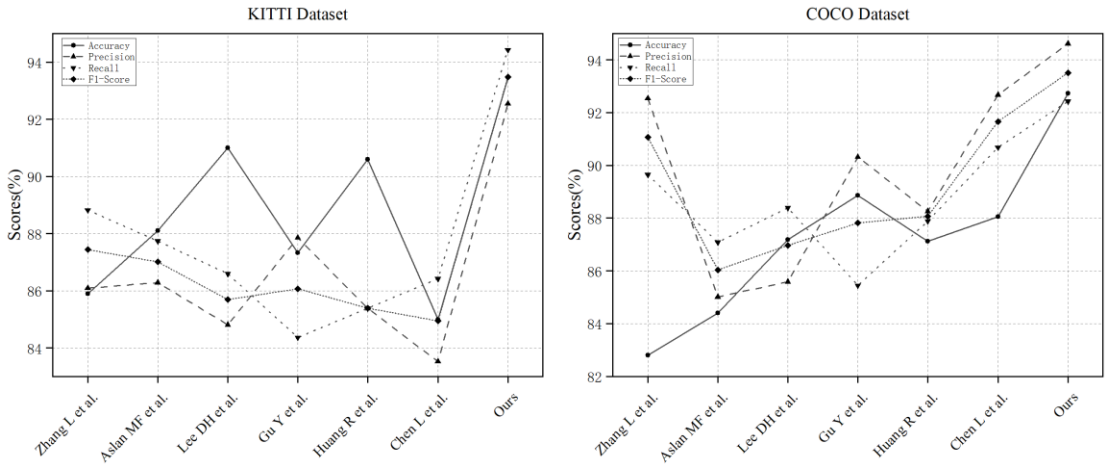

Figure 6. Comparative visualization of each model indicator under the KITTI Dataset and COCO Dataset.

Table 2. Comparison of indicators of various models under the RobotCar Dataset and NuScenes Dataset.

Model	RobotCar Dataset				NuScenes Dataset			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Zhang L et al. (Zhang et al., 2020a)	83.39	81.20	82.47	81.83	87.58	85.42	90.98	88.11
Aslan MF et al. (Aslan et al., 2022)	84.07	89.32	86.25	87.76	88.40	90.31	88.66	89.48
Lee DH et al. (Lee & Liu, 2023)	83.87	81.91	89.41	85.50	90.53	91.18	91.53	91.35
Gu Y et al. (Gu et al., 2023)	87.74	80.22	88.57	84.19	87.30	83.65	87.77	85.66
Huang R et al. (Huang et al., 2023)	88.89	81.35	82.15	81.75	90.17	82.41	88.76	85.47
Chen L et al. (Chen et al., 2022)	88.35	90.19	83.66	86.80	86.48	89.26	91.85	90.54
Ours	91.43	93.43	92.43	92.93	94.24	93.76	95.63	94.69

Table 2 presents the comparison results of algorithms on the RobotCar and NuScenes datasets. From the table, it is evident that our method significantly outperforms others in terms of accuracy (91.43%), precision (93.43%), recall (92.43%), and F1 score (92.93%) on the RobotCar dataset. Similarly, on the NuScenes dataset, our method demonstrates excellent performance in accuracy (94.24%), precision (93.76%), recall (95.63%), and F1 score (94.69%). In comparison, while some methods show better performance in certain metrics, such as Aslan MF et al.'s precision (89.32%) on the RobotCar dataset and Chen L et al.'s recall (91.85%) on the NuScenes dataset, none surpasses our method overall. This indicates that our proposed method holds significant performance advantages in path planning tasks across various complex driving environments. At the same time, we show the visualization of various indicator comparisons in Figure 7.

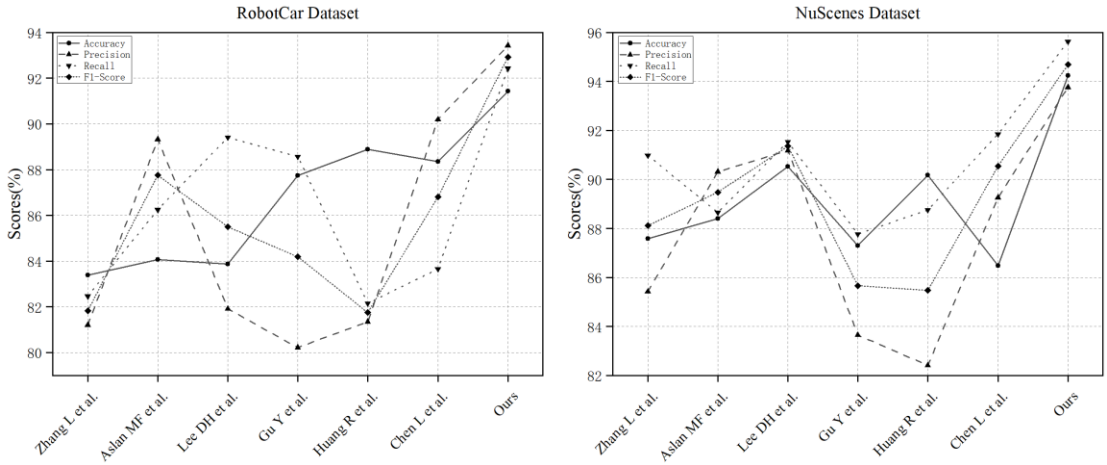

Figure 7. Comparative visualization of each model indicator under the RobotCar Dataset and NuScenes Dataset.

Table 3. Metrics of multiple models on four datasets.

Model	KITTI Dataset			COCO Dataset		
	Parameters (M)	Inference Time(ms)	Trainning Time(s)	Parameters (M)	Inference Time(ms)	Trainning Time(s)
Zhang L et al. (Zhang et al., 2020a)	441.09	391.13	214.57	426.47	283.61	260.81
Aslan MF et al. (Aslan et al., 2022)	525.52	339.68	260.42	497.47	385.35	278.70
Lee DH et al. (Lee & Liu, 2023)	388.58	346.01	234.46	433.20	312.43	280.25
Gu Y et al. (Gu et al., 2023)	492.10	398.87	229.25	397.64	274.91	211.57
Huang R et al. (Huang et al., 2023)	405.60	314.69	241.51	472.63	290.25	242.96
Chen L et al. (Chen et al., 2022)	422.81	295.16	220.05	442.68	324.70	204.36
Ours	367.24	264.34	161.45	374.73	257.94	182.43
Model	RobotCar Dataset			NuScenes Dataset		
	Parameters (M)	Inference Time(ms)	Trainning Time(s)	Parameters (M)	Inference Time(ms)	Trainning Time(s)
Zhang L et al. (Zhang et al., 2020a)	477.26	379.27	213.06	465.14	342.26	229.57
Aslan MF et al. (Aslan et al., 2022)	396.20	385.73	202.88	384.25	323.36	265.15
Lee DH et al. (Lee & Liu, 2023)	387.62	315.08	195.18	471.66	297.13	249.15
Gu Y et al. (Gu et al., 2023)	458.96	397.01	270.96	452.13	300.50	298.61
Huang R et al. (Huang et al., 2023)	504.18	385.52	268.73	378.21	307.37	228.53
Chen L et al. (Chen et al., 2022)	475.99	301.89	266.20	415.66	316.34	236.99
Ours	362.94	261.84	178.02	356.64	279.71	193.41

Table 3 presents a comparative analysis of the number of model parameters (Parameters), inference time (Inference Time), and training time (Training Time) for various path optimization algorithms across four datasets. In terms of the number of model parameters, our method consistently exhibits the lowest parameter count across all datasets (e.g., 367.24 M for the KITTI dataset and 374.73 M for the COCO dataset). This indicates that our method achieves efficient path optimization while maintaining a minimal parameter count. Regarding inference time, our method consistently demonstrates the fastest inference speed across all datasets (e.g., 264.34 ms for the

KITTI dataset and 257.94 ms for the COCO dataset). This suggests that our method has a significant speed advantage in real-time applications, enabling faster path planning and decision-making. In terms of training time, our method shows the shortest training time across all datasets (e.g., 161.45 s for the KITTI dataset and 182.43 s for the COCO dataset). This indicates that our method is more efficient during model training, achieving faster convergence to the optimal state. On the other hand, Aslan MF et al. have longer inference and training times on certain datasets (e.g., 323.36 ms inference time and 265.15 s training time on the NuScenes dataset), while Gu Y et al. have a larger number of model parameters (e.g., 492.10 M on the KITTI dataset). Overall, our proposed method performs excellently on all key metrics across the four datasets, indicating significant performance advantages in path optimization tasks, particularly in model simplicity, inference speed, and training efficiency. At the same time, we show the visualization of various indicator comparisons in Figure 8.

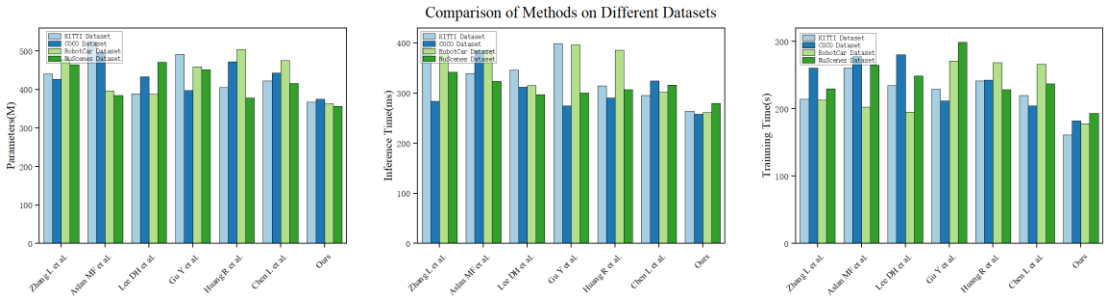

Figure 8. Visual comparison of indicators of multiple models on four datasets.

Table 4. Ablation experiments of this model on the KITTI Dataset and COCO Dataset.

Model	Dataset					
	KITTI Dataset			COCO Dataset		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
baseline	78.64	79.24	78.94	81.73	79.37	80.53
+SAM	83.03	84.73	83.87	86.27	86.24	86.25
＋DDPG	89.38	90.93	90.15	91.06	88.41	89.72
+SAM-DDPG	92.54	94.43	93.48	94.61	92.43	93.51

Table 4 presents the results of ablation experiments conducted on the KITTI dataset and COCO dataset. For the KITTI dataset, the baseline model achieves a precision of 78.64%, recall of 79.24%, and an F1-score of 78.94%. Introducing the SAM module significantly improves performance, with precision reaching 83.03%, recall at 84.73%, and an F1-score of 83.87%. Subsequently, incorporating the DDPG module on top of SAM further enhances performance, with precision, recall, and F1-score reaching 89.38%, 90.93%, and 90.15%, respectively. Finally, combining SAM and DDPG results in the best performance, with precision at 92.54%, recall at 94.43%, and an F1-score of 93.48%. Similar trends are observed for the COCO dataset: as modules are introduced, performance steadily improves, with the SAM-DDPG combination achieving the best performance, with precision, recall, and F1-score at 94.61%, 92.43%, and 93.51%, respectively. These results show that the introduction of the spatial attention mechanism and the deep deterministic policy gradient algorithm significantly enhances the performance of the model. At the same time, we show the visualization of various indicator comparisons in Figure 9.

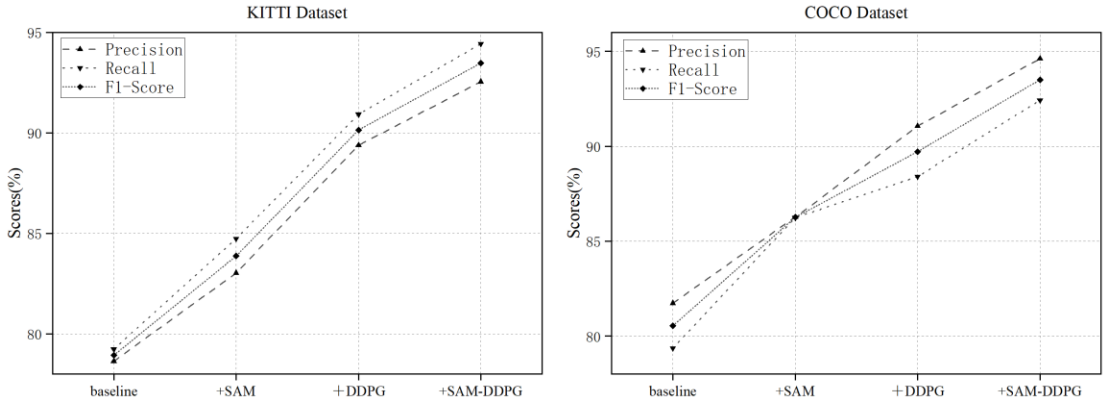

Figure 9. Comparative visualization of ablation experiments on KITTI Dataset and COCO Dataset.

Table 5. Ablation experiments of this model on the RobotCar Dataset and NuScenes Dataset.

Model	Dataset					
	RobotCar Dataset			NuScenes Dataset		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
baseline	79.24	80.61	79.92	81.68	82.46	82.07
+FPN	87.71	86.04	86.87	85.72	88.62	87.15
＋ViT	89.79	88.57	89.18	88.73	90.04	89.38
+FPN ViT	93.43	92.43	92.93	93.76	95.63	94.69

Table 5 illustrates the results of ablation experiments conducted on the RobotCar dataset and NuScenes dataset. For the RobotCar dataset, the baseline model achieves a precision of 79.24%, recall of 80.61%, and an F1-score of 79.92%. Introduction of the SAM module leads to significant performance enhancement, with precision reaching 87.71%, recall at 86.04%, and an F1-score of 86.87%. Further incorporation of the DDPG module on top of SAM results in improved performance, with precision, recall, and F1-score reaching 89.79%, 88.57%, and 89.18%, respectively. Finally, combining SAM and DDPG yields the highest performance, with precision at 93.43%, recall at 92.43%, and an F1-score of 92.93%. Similar trends are observed for the NuScenes dataset. At the same time, we show the visualization of various indicator comparisons in Figure 10.

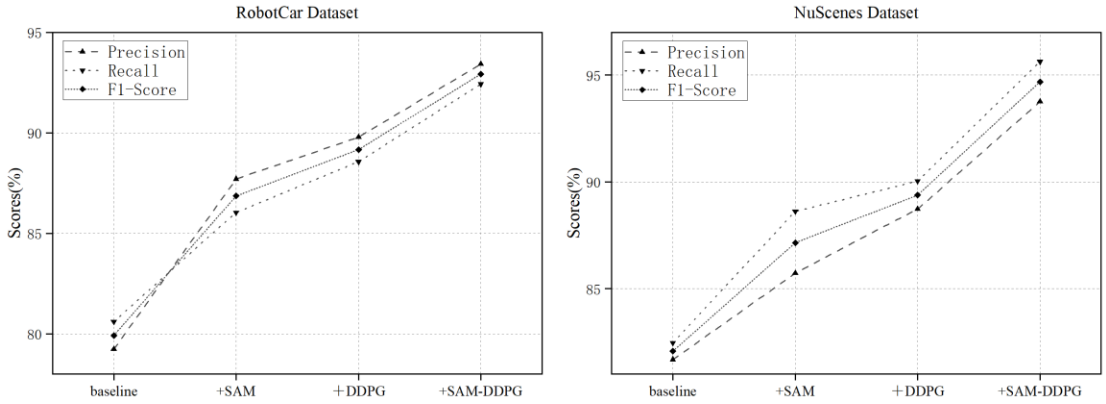

Figure 10. Comparative visualization of ablation experiments on RobotCar Dataset and NuScenes Dataset.

5 Conclusion

This article proposes a path optimization method that combines spatial attention mechanism with deep deterministic policy gradient algorithm. Its superior performance is validated through experiments on multiple complex datasets. In this path optimization model, a spatial attention mechanism is introduced to enhance the model's perception ability by dynamically adjusting the focus area. Experimental results show that after adding SAM, the precision, recall, and F1 score of the model significantly improve on KITTI and COCO datasets, verifying its effectiveness in complex environments. By integrating the DDPG algorithm to optimize the path planning strategy, the model can efficiently learn in high-dimensional continuous action spaces. The experiments demonstrate that adding DDPG leads to significant improvements in various metrics, particularly in real-time dynamic environments. Combining SAM and DDPG, a new path optimization method is proposed, which outperforms existing path optimization algorithms in key metrics such as accuracy, precision, recall, and F1 score, showcasing its significant performance advantages in various complex environments. Finally, our method not only excels in accuracy and recall but also demonstrates significant advantages in terms of model parameter count, inference time, and training time. Experimental results show that our method achieves efficient path optimization while maintaining fewer parameters, which is crucial for real-time performance and resource efficiency in practical applications. Although the proposed method demonstrates significant performance advantages in multiple complex environments, there are still many directions worth further exploration and improvement. Future research could focus on enhancing the model's generalization ability, optimizing computational efficiency, integrating multimodal perception, exploring adaptive strategies, and human-machine cooperative optimization. Through further research in these areas, we aim to enhance the performance and application value of path optimization methods, providing more solid technical support for the development of intelligent robots.

Funding
Not applicable

Author Contributions

Conceptualization, D. Z. and X. C.; writing—original draft preparation, D. Z. and Y. G.; writing—review and editing, Y. G. and X. C.; All of the authors read and agreed to the published the final manuscript.

Institutional Reviewer Board Statement
Not applicable

Informed Consent Statement
Not applicable

Data Availability Statement
Not applicable

Conflict of Interest

The authors declare no conflict of interest.

Reference

[1] Aslan, M. F., Durdu, A., and Sabanci, K. (2022). Visual-inertial image-odometry network (viionet): A gaussian process regression-based deep architecture proposal for uav pose estimation. Measurement 194, 111030

[2] Chang, L., Shan, L., Jiang, C., and Dai, Y. (2021). Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. Autonomous robots 45, 51–76

[3] Chen, L., Jiang, Z., Cheng, L., Knoll, A. C., and Zhou, M. (2022). Deep reinforcement learning based trajectory planning under uncertain constraints. Frontiers in Neurorobotics 16, 883562

[4] Chen, L., Wu, P., Chitta, K., Jaeger, B., Geiger, A., and Li, H. (2023). End-to-end autonomous driving: Challenges and frontiers. arXiv preprint arXiv:2306.16927

[5] Erke, S., Bin, D., Yiming, N., Qi, Z., Liang, X., and Dawei, Z. (2020). An improved a-star based path planning algorithm for autonomous land vehicles. International Journal of Advanced Robotic Systems 17, 1729881420962263

[6] Gomes, A. C., de Lima Junior, F. B., Soliani, R. D., de Souza Oliveira, P. R., de Oliveira, D. A., Siqueira, R. M., et al. (2023). Logistics management in e-commerce: challenges and opportunities. Revista de Gest˜ao e Secretariado 14, 7252–7272

[7] Gu, Y., Zhu, Z., Lv, J., Shi, L., Hou, Z., and Xu, S. (2023). Dm-dqn: Dueling munchausen deep q network for robot path planning. Complex & Intelligent Systems 9, 4287–4300

[8] Gupta, A., Anpalagan, A., Guan, L., and Khwaja, A. S. (2021). Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. Array 10, 100057

[9] Huang, R., Qin, C., Li, J. L., and Lan, X. (2023). Path planning of mobile robot in unknown dynamic continuous environment using reward-modified deep q-network. Optimal Control Applications and Methods 44, 1570–1587

[10] Jiang, M. and Huang, G. Q. (2022). Intralogistics synchronization in robotic forward-reserve warehouses for e-commerce last-mile delivery. Transportation Research Part E: Logistics and Transportation Review158, 102619

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

[12] Lee, D.-H. and Liu, J.-L. (2023). End-to-end deep learning of lane detection and path prediction for real-time autonomous driving. Signal, Image and Video Processing 17, 199–205

[13] Lee, M.-F. R. and Yusuf, S. H. (2022). Mobile robot navigation using deep reinforcement learning. Processes 10, 2748

[14] Li, J., Qiao, Y., Liu, S., Zhang, J., Yang, Z., and Wang, M. (2022). An improved yolov5-based vegetable disease detection method. Computers and Electronics in Agriculture 202, 107345

[15] Mirahadi, F. and McCabe, B. Y. (2021). Evacusafe: A real-time model for building evacuation based on dijkstra's algorithm. Journal of Building Engineering 34, 101687

[16] Neri, I. and Dinarama, E. (2024). Cities' match-making: Fostering international collaboration for climate-resilient twins. In The Routledge Handbook on Greening High-Density Cities (Routledge). 15–29

[17] Qadir, Z., Ullah, F., Munawar, H. S., and Al-Turjman, F. (2021). Addressing disasters in smart cities through uavs path planning and 5g communications: A systematic review. Computer Communications168, 114–135

[18] Segato, A., Di Marzo, M., Zucchelli, S., Galvan, S., Secoli, R., and De Momi, E. (2021). Inverse reinforcement learning intra-operative path planning for steerable needle. IEEE Transactions on Biomedical Engineering 69, 1995–2005

[19] Torres, J. F., Hadjout, D., Sebaa, A., Mart´ınez- ´Alvarez, F., and Troncoso, A. (2021). Deep learning for time series forecasting: a survey. Big Data 9, 3–21

[20] Wang, J., Liu, Y., and Li, B. (2020). Reinforcement learning with perturbed rewards. In Proceedings of the AAAI conference on artificial intelligence. vol. 34, 6202–6209

[21] Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., et al. (2022). Deep reinforcement learning: A survey. IEEE Transactions on Neural Networks and Learning Systems

[22] Wang, Y., Li, X., Zhang, J., Li, S., Xu, Z., and Zhou, X. (2021). Review of wheeled mobile robot collision avoidance under unknown environment. Science Progress 104, 00368504211037771

[23] Wu, J. and Li, H. (2020). Deep ensemble reinforcement learning with multiple deep deterministic policy gradient algorithm. Mathematical Problems in Engineering 2020, 1–12

[24] Wu, Z., Meng, Z., Zhao, W., and Wu, Z. (2021). Fast-rrt: A rrt-based optimal path finding method. Applied sciences 11, 11777

[25] Yan, B., Chen, T., Zhu, X., Yue, Y., Xu, B., and Shi, K. (2020). A comprehensive survey and analysis on path planning algorithms and heuristic functions. In Intelligent Computing: Proceedings of the 2020

[26] Computing Conference, Volume 1 (Springer), 581–598

[27] Zhang, L., Zhang, Y., and Li, Y. (2020a). Path planning for indoor mobile robot based on deep learning. Optik 219, 165096

[28] Zhang, Z., Wu, J., Dai, J., and He, C. (2020). A novel real-time penetration path planning algorithm for stealth uav in 3d complex dynamic environment. Ieee Access 8, 122757–122771

[29] Zhao, C., Zhu, Y., Du, Y., Liao, F., and Chan, C.-Y. (2022). A novel direct trajectory planning approach based on generative adversarial networks and rapidly-exploring random tree. IEEE Transactions on

[30] Intelligent Transportation Systems 23, 17910–17921

[31] Zhao, J., Zhao, W., Deng, B., Wang, Z., Zhang, F., Zheng, W., et al. (2023). Autonomous driving system: A comprehensive survey. Expert Systems with Applications , 122836

[32] Zhou, Y., Xiao, J., Zhou, Y., and Loianno, G. (2022). Multi-robot collaborative perception with graph neural networks. IEEE Robotics and Automation Letters 7, 2289–2296