Proceedings of the 8th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/2026.38719

A Review of Path Planning Methods for Mobile Robots

Liyuan Wang

School of International Education, Beijing University of Chemical Technology, Beijing, China
15820070718@126.com

Abstract. Path planning is an indispensable core problem in the autonomous navigation of mobile robots. Whether a safe and efficient trajectory can be planned directly determines the usability of robots in scenarios such as warehousing, delivery, medical assistance and autonomous driving. Scholars at home and abroad have accumulated abundant research results on this problem. Nevertheless, existing methods have respective trade-offs in application scope and performance limits, and no universally recognized optimal solution has been reached so far. This paper sorts out relevant methods. These methods are divided into two categories: traditional methods and intelligent methods. The traditional category includes graph search algorithms (Dijkstra, A*, D*), artificial potential field methods and sampling-based planning algorithms (PRM, RRT, RRT*). The intelligent category covers intelligent optimization methods represented by genetic algorithms, particle swarm optimization and ant colony optimization, as well as two data-driven approaches: reinforcement learning and deep learning. The core idea, typical application scenarios and limitations of each type of method are elaborated one by one. Finally, a horizontal comparison table is provided, and the author puts forward observations on several noteworthy research directions including hybrid planning frameworks, multi-robot coordination and uncertainty modelling.

Keywords: mobile robot, path planning, optimization algorithm, reinforcement learning, development trend

1. Introduction

Path planning serves as the foundation for mobile robots to achieve autonomous navigation. Its core objective is to find a collision-free motion trajectory from the start point to the target point for the robot within a given environment, which satisfies optimization criteria such as the shortest path, minimum time or lowest energy consumption [1]. This problem seems straightforward, yet it involves multiple modules including environmental perception, spatial representation, search strategy and real-time decision-making, bringing considerable complexity to algorithm design.

Over the past decades, the research focus of planning problems has been shifting gradually. Early algorithms concentrated on "how to find the shortest path in a known map". This problem has an elegant mathematical formulation, and *Dijkstra* and *A** have basically provided a sound solution. However, practical application scenarios at present are entirely different. Warehouse robots need to avoid moving workers nearby; delivery robots may encounter pedestrians suddenly appearing at

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.38719

intersections; autonomous vehicles must make real-time decisions under hundreds of thousands of traffic situations. These scenarios collectively reveal that the research emphasis of path planning has shifted from "optimal solutions" to "reasonable decision-making under imperfect information". Against this backdrop, traditional methods and intelligent methods have evolved into two parallel technical routes.

Fundamentally, the distinction between traditional methods and intelligent methods lies in whether explicit modelling is required. Traditional methods demand prior explicit description of the environment, such as grid maps, visibility graphs or potential fields, before searching on this mathematical structure. Intelligent methods try to skip the modelling step and enable algorithms to learn rules through interaction with the environment. Traditional methods feature controllability and provable properties, while intelligent methods suffer from poor interpretability and high training costs [2]. Figure 1 summarizes representative methods under these two routes.

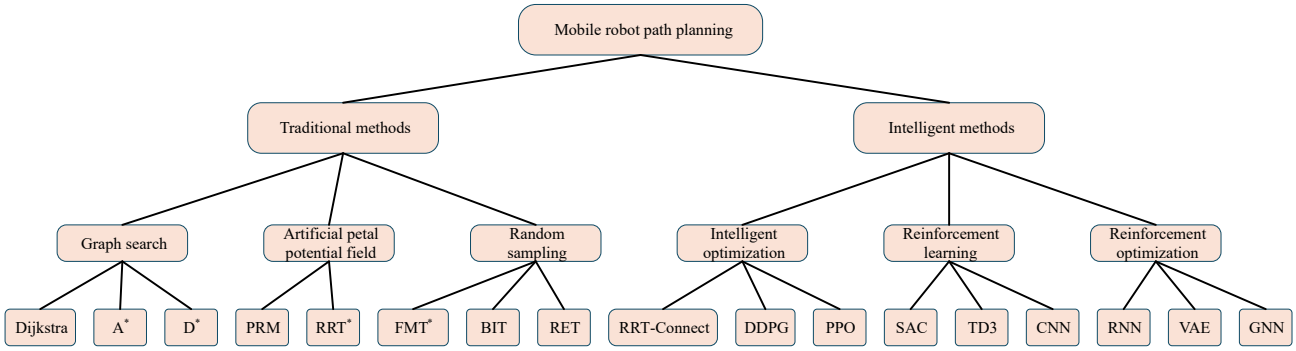


Figure 1. Classification of path planning methods for mobile robots

The work of this paper mainly covers three parts. First, it explains the principles and boundary conditions of key algorithms in the two major categories. Second, it visually presents their differences through figures and tables. Third, it discusses the author's opinions on existing challenges and potential research directions. The rest of this paper is organized as follows. Section 2 focuses on traditional methods. Section 3 introduces intelligent methods. Section 4 carries out horizontal comparison and prospect of research trends. Section 5 draws the conclusion.

2. Traditional path planning methods for mobile robots

2.1. Overview

Traditional methods are grouped together because they share a common premise: the environment is known and can be fully described by mathematical expressions. Whether graph search algorithms discretize the map into nodes and edges, artificial potential field methods convert targets and obstacles into potential values, or sampling-based methods approximate free space using random points, they essentially translate the real world into a mathematical problem first and then solve it with corresponding tools. This technical route has the advantages of clear theoretical basis and predictable behaviour, yet it also has an obvious drawback: real environments are rarely fully "known". The left branch in Figure 1 lists three types of representative methods, which are elaborated below one by one.

DOI: 10.54254/2755-2721/2026.38719

2.2. Graph-search-based path planning methods

Graph search methods contain only two core steps. First, the continuous workspace is decomposed into a graph, where nodes represent positions and edges represent passability. Second, a certain search strategy is adopted on this graph to find an optimal path from the start point to the target point. Maps can be discretized through grid maps, visibility graphs, Voronoi diagrams and other forms, and a series of classic algorithms have been developed for search strategies. Such methods possess a valuable property: a solution will definitely be found if a solution exists within the discretized space, namely resolution completeness. Figure 2 intuitively compares the search behaviours of the *Dijkstra* and *A** algorithms in a typical grid environment.

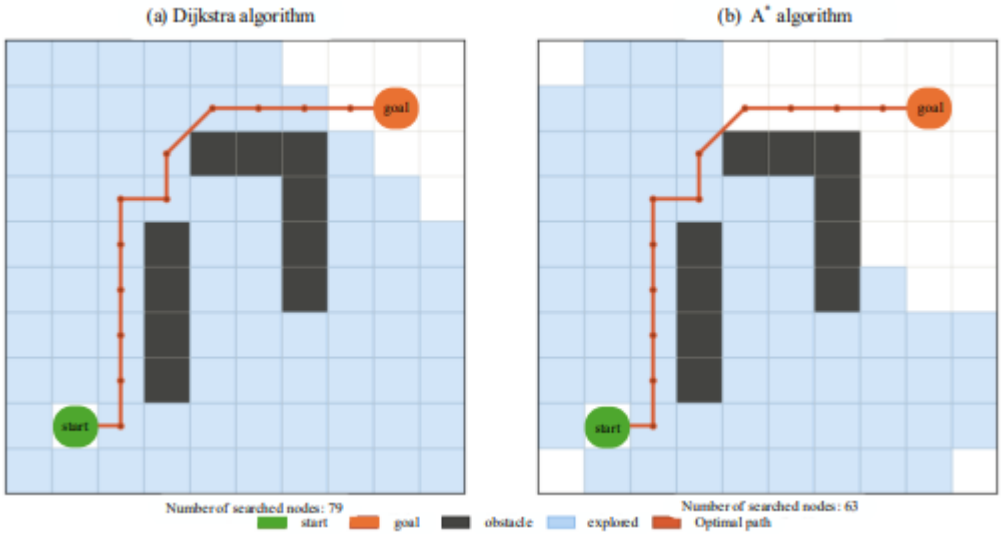


Figure 2. Comparison of search behaviours of *Dijkstra* and *A** algorithms on grid maps

2.2.1. Dijkstra algorithm

The *Dijkstra* algorithm dates back to 1959, and it was originally designed to solve the shortest path problem between cities [3]. Its logic is straightforward. Starting from the start node, the algorithm expands the node with the currently known minimum cost in each iteration. A priority queue maintains the sequence of nodes to be expanded, and the iteration repeats until the target node is extracted. This mechanism guarantees a globally optimal solution, on the condition that all edge weights in the graph are non-negative.

The strengths of *Dijkstra* are clear: it is both complete and optimal, and theoretically nearly flawless. Its weakness lies in efficiency. Since the algorithm has no information about the target direction at each step, expansion proceeds evenly outwards and wastes computation on a large number of nodes. This feature is illustrated in Figure 2(a): a vast area around the start point is explored, while most nodes contribute nothing to the final path. In terms of time complexity, the naive implementation runs at $O(V^2)$, and it can be optimized to $O((V+E) \log V)$ with a binary heap. For this reason, *Dijkstra* mostly serves as a benchmark algorithm today, and more efficient variants are selected for large-scale planning tasks in engineering practice.

DOI: 10.54254/2755-2721/2026.38719

2.2.2. A* algorithm

Hart, Nilsson and Raphael put forward an improved approach in their 1968 paper. Since the drawback of *Dijkstra* is the lack of directional information towards the target, heuristic hints about the target can be supplied to the algorithm. This idea gives rise to the *A** algorithm [4]. Its evaluation function is defined as f(n) = g(n) + h(n), where g(n) denotes the actual cost from the start point to the current node n, and h(n) is the heuristic estimate of the cost from node n to the target point. As long as h(n) satisfies the admissible condition, that is, it never overestimates the true cost, *A** can greatly narrow the search scope while retaining optimality.

Figure 2(b) clearly demonstrates the improvement of *A** over *Dijkstra*. The explored region tilts towards the target direction, bringing substantial progress in overall efficiency. In practical applications, the selection of heuristic functions needs to match the motion model. Manhattan distance applies to movements limited to horizontal and vertical directions, Euclidean distance suits diagonal movement, and Chebyshev distance is used for eight-connected grids. Nevertheless, *A** still has limitations. It suffers from high memory consumption in large-scale environments and poor responsiveness to dynamic environments. These deficiencies motivate researchers to develop various improved versions. For instance, Xu et al. [5] combined an improved *A** algorithm with the dynamic window approach, and achieved good experimental results in random obstacle avoidance tasks, which verified the effectiveness of the improved strategy.

2.2.3. D* algorithm

To handle dynamic environments, Stentz proposed the *D** algorithm, short for Dynamic A*, in 1994 [6], to address the limitations of *A** in dynamic and partially known environments. The key innovation of *D** lies in incremental replanning. Once environmental changes are detected, the algorithm does not restart the whole search from scratch, but carries out local modifications based on the existing planning result. This concept has been extended to multiple improved variants including Focused D* and D* Lite. Methods in the D* family are especially suitable for scenarios where obstacles suddenly appear or shift positions during navigation.

In engineering practice, *D** has been deployed successfully on outdoor mobile robots and planetary rovers. Compared with rerunning *A** from the beginning after every environmental change, *D** cuts computational cost significantly in dynamic scenarios. Still, there is no free lunch. *D** has much higher implementation complexity, and extra state information must be maintained to support incremental updates. Huang Lu and Zhou Feitong further optimized D* Lite. Experimental results show that the length of planned paths is reduced, which indirectly proves the engineering feasibility of the incremental replanning framework.

Apart from the three classic algorithms above, improved variants such as Theta*, Jump Point Search (JPS) and Anytime D* have emerged one after another. They provide solutions targeting path smoothness, search acceleration and time-bounded planning respectively. On balance, graph search methods enjoy solid theoretical guarantees and remain fundamental tools for mobile robot path planning up to now.

2.3. Artificial potential field based path planning methods

If graph search methods follow the idea of "combinatorial optimization", the artificial potential field method (*APF*) takes a different route, converting path planning into a continuous physical process. Proposed by Khatib in 1986 [7], this method remains a representative local planning approach to

DOI: 10.54254/2755-2721/2026.38719

this day. Its core principle can be summarized briefly: the target exerts an attractive force on the robot, while obstacles generate repulsive forces. The robot behaves like a charged particle and naturally moves toward the target under the combined force. Figure 3 illustrates the working principle of *APF*.

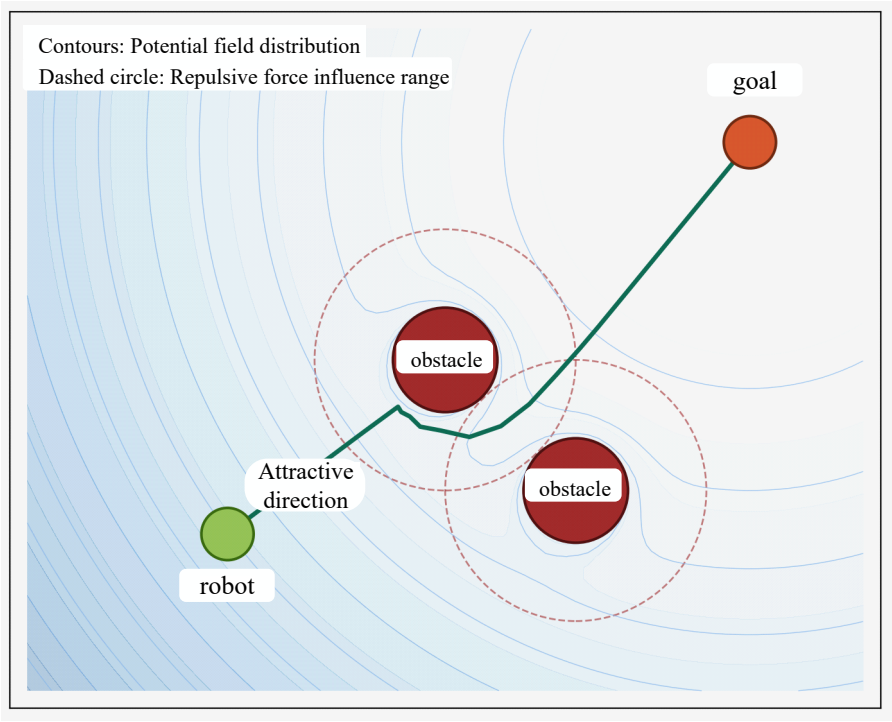


Figure 3. Schematic diagram of the artificial potential field method

In specific design, the attractive potential is generally a quadratic function of the distance between the robot and the target. The attractive force decreases as the distance shrinks and drops to zero when the robot reaches the target. The repulsive potential is often defined as an inverse function of the distance to obstacles and only takes effect within a certain influence range. Parameter tuning allows flexible trade-offs between safety margin and trajectory smoothness. *APF* has obvious advantages: simple implementation, low computational load and smooth trajectories. These features enable its wide application in ground robots, unmanned aerial vehicles and surface unmanned boats.

Nevertheless, the classic *APF* is not flawless and suffers from several long-standing drawbacks. The most severe one is the local minimum problem. When attractive and repulsive forces cancel each other out at a certain point, the robot gets trapped and cannot move forward or backward. Another issue is the GNRON (Goal Non-Reachable with Obstacle Nearby) problem. If the target lies close to an obstacle, strong repulsive force pushes the robot away and prevents it from reaching the target. Besides, alternating dominance of attractive and repulsive forces in narrow passages may trigger trajectory oscillation.

A large number of improved schemes have been put forward to tackle these limitations. Some introduce virtual target points or auxiliary forces to escape local minima [8]; some redesign the repulsive potential function by incorporating the relative distance between obstacles and the target. Many studies combine *APF* with *A** or sampling-based methods to build hybrid planning frameworks. Recent attempts also adopt fuzzy logic, neural networks and even reinforcement

DOI: 10.54254/2755-2721/2026.38719

learning to adjust potential field parameters dynamically. Despite these weaknesses, *APF* still occupies an important position in practical systems. It is almost a standard choice as the local planner within hierarchical planning architectures.

2.4. Random sampling based path planning methods

Modelling the entire high-dimensional configuration space brings excessive overhead. Sampling-based methods offer an elegant solution: instead of constructing the full space model, they sample random points, connect collision-free pairs, and gradually approximate the true connectivity of free space. This strategy works exceptionally well for high-dimensional problems and largely circumvents the curse of dimensionality. Figure 4 compares the tree growth behaviour of *RRT* and its optimal variant *RRT**.

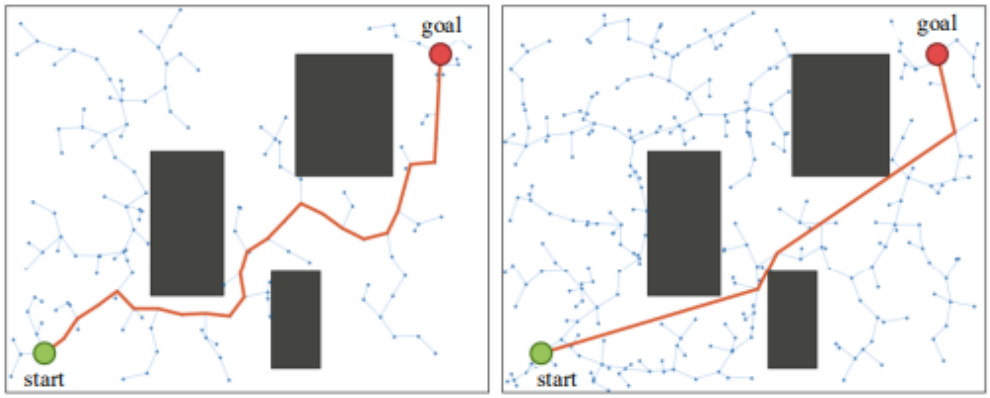


Figure 4. Sampling-based path planning: comparison between *RRT* (left) and *RRT** (right)

2.4.1. PRM algorithm

PRM was proposed by Kavraki et al. in 1996 [9]. It divides the planning process into two stages. In the learning phase, random samples are generated in the configuration space, and collision-free samples are connected via local planners to form a roadmap. In the query phase, the given start and target points are connected to this roadmap, and classic graph search algorithms are used to find a path.

This "roadmap construction followed by query" design holds probabilistic completeness. With sufficient samples, the probability of finding a valid solution converges to one. *PRM* shows prominent strengths for multiple query tasks in static environments because the same roadmap can be reused repeatedly. Its performance degrades sharply in narrow passages, however. The reason is straightforward: random samples have a low probability of falling into narrow channels. Dynamic environments also pose challenges for *PRM*, as roadmap reconstruction after environmental changes incurs high costs.

2.4.2. RRT algorithm

RRT adopts a different logic. It was presented by LaValle in a 1998 technical report [10]. The algorithm grows a search tree starting from the start node. In each iteration, a random sample is drawn. The nearest node on the existing tree is identified, and a small step is extended toward this random sample. The procedure repeats until a node lies sufficiently close to the target.

DOI: 10.54254/2755-2721/2026.38719

One key property of *RRT* is its tendency to expand toward unexplored regions. This makes it highly effective in high-dimensional complex environments, and it is widely applied to nonholonomic vehicles, manipulators and autonomous driving systems. Still, it has clear limitations. The resulting paths vary greatly in quality. As shown in Figure 4(a), paths contain many redundant detours and sharp turns, because *RRT* only guarantees feasibility rather than optimality.

2.4.3. RRT* algorithm

*RRT** was developed to address this shortcoming. Karaman and Frazzoli proposed it in 2011 [10]. Two critical improvements are added on the basis of *RRT*. First, the lowest-cost parent node is selected for each new node. Second, nearby nodes are rewired when a shorter path is discovered. These operations ensure that, as the number of samples approaches infinity, the path returned by *RRT** converges to the global optimum with probability one. The improvement is visible in Figure 4(b): paths generated by *RRT** are noticeably shorter and smoother than those from *RRT*.

At present, *RRT** ranks among the most widely used sampling-based planners in robot motion planning, and numerous variants have emerged. For example, Informed *RRT** restricts sampling within a heuristic ellipsoidal region to boost sampling efficiency; BIT* tries to combine merits of sampling methods and graph search. In domestic research, Lin Yifan et al. [11] simplified the collision detection pipeline and further accelerated *RRT**. Even so, sampling-based methods still face unsolved challenges concerning narrow passages, kinematic constraints and real-time performance in dynamic scenarios.

3. Intelligent path planning methods for mobile robots

3.1. Overview

The traditional methods discussed above perform well in known static environments, yet their limitations emerge when facing dynamic changes, high dimensionality or complicated constraints. For this reason, "intelligent" path planning methods inspired by biological evolution, swarm behaviour and human learning have developed rapidly over the past two decades. These methods share the feature of data-driven operation. They no longer require a pre-built environmental model, but "learn" proper policies from statistical patterns in data or interactive feedback. The right branch in Figure 1 lists three major representative categories: intelligent optimization algorithms, reinforcement learning and deep learning, which are introduced one by one below.

3.2. Path planning methods based on intelligent optimization algorithms

Intelligent optimization algorithms draw design inspiration from natural phenomena such as heredity, swarm behaviour and biological societies. These algorithms are all stochastic search approaches. They explore the solution space through certain heuristic or cooperative mechanisms. For path planning, a typical problem with multiple objectives, tight constraints and a large search space, they serve as flexible solving tools.

3.2.1. Genetic algorithm

The genetic algorithm (*GA*) was proposed by Holland in the 1970s [12], inspired by biological evolution. When applied to path planning, each candidate path is treated as a "chromosome", usually encoded as a sequence of waypoints. A fitness function, which generally combines path

DOI: 10.54254/2755-2721/2026.38719

length, smoothness and collision status, assigns scores to chromosomes. Three basic operations, namely selection, crossover and mutation, drive the continuous evolution of the population. After several generations of iteration, high-quality paths gradually dominate.

GA has strong global search capacity and fits naturally into multi-objective optimization tasks. It works especially well for offline planning in complex static environments. Its main drawback is that convergence demands numerous fitness evaluations, leading to poor real-time performance. Besides, the choice of encoding scheme, fitness function and genetic operators greatly affects algorithm performance, and parameter tuning is usually required for specific problems.

3.2.2. Particle swarm optimization algorithm

Particle swarm optimization (*PSO*) was put forward by Kennedy and Eberhart in 1995 [13], inspired by cooperative swarm behaviours including bird flocking and fish schooling. In this algorithm, each particle represents a candidate path inside the search space. Its position update relies on two pieces of information: the best position discovered by itself and the best position found by the whole swarm.

PSO features a simple structure, few parameters and fast convergence, so it is widely adopted in path planning. Nevertheless, it suffers from the same chronic defect as *GA*: premature convergence and trapping in local optima. Adaptive PSO, chaotic PSO and various hybrid PSO variants have been proposed to ease these issues to some extent.

3.2.3. Ant colony optimization algorithm

Ant colony optimization (*ACO*) was developed by Dorigo et al. in 1996 [14], inspired by the foraging behaviour of real ant colonies. In path planning tasks, "virtual ants" travel through the environment and deposit pheromone along traversed paths. Pheromone accumulates faster on good paths, attracting more ants to follow these routes. With the joint effect of positive feedback and pheromone evaporation, the algorithm eventually converges to a high-quality path.

ACO achieves outstanding performance on discrete optimization problems, especially path planning on grid maps. Its strengths include strong robustness, inherent parallelizability and moderate adaptability to dynamic environments. Its weaknesses lie in slow early-stage convergence and high parameter sensitivity. Improved algorithms such as the Max-Min Ant System (*MMAS*) and elitist ant systems have mitigated these problems considerably. In domestic research, Shi Enxiu et al. [15] adjusted heuristic factors and pheromone update strategies to improve both the convergence speed and path quality of ACO. It is also worth noting that, apart from GA, PSO and ACO, artificial bee colony (*ABC*), grey wolf optimizer (*GWO*) and whale optimization algorithm (*WOA*) have been introduced into path planning in recent years, expanding the available toolset.

3.3. Path planning methods based on reinforcement learning

The core idea of reinforcement learning (*RL*) can be stated concisely: an agent learns decision-making through repeated interactions with the environment [16]. In mobile robot path planning, the robot serves as this agent. It observes environmental states, selects actions following its current policy, receives reward feedback from the environment, and updates the policy to maximize long-term cumulative rewards. The most attractive feature of *RL* is that it does not require an explicit environmental model, which is especially valuable for dynamic and uncertain scenarios.

DOI: 10.54254/2755-2721/2026.38719

In early research, classic *RL* algorithms including Q-learning and SARSA were widely applied to mobile robot path planning. These methods work by maintaining a Q-table that stores the estimated value of each action under every state, and values are updated iteratively according to the Bellman equation. The limitation is that tabular methods can only handle low-dimensional discrete state spaces. They work for small-scale grid environments but fail when faced with moderately complex situations.

Deep reinforcement learning (*DRL*) breaks this limitation. The *DQN* proposed by Mnih et al. and published in *Nature* in 2015 [17] approximates the Q-function with deep neural networks, enabling *RL* to process high-dimensional inputs such as images. This is landmark work. After that, Double DQN, Dueling DQN, Deep Deterministic Policy Gradient (*DDPG*), Proximal Policy Optimization (*PPO*), Soft Actor-Critic (*SAC*) and other algorithms were proposed in succession. They allow robots to learn navigation policies directly from laser point clouds or RGB images.

In application, Tai et al. [18] presented interesting work at IROS 2017. They built a mapless navigation system using *DRL*. The robot completes navigation relying only on raw laser ranging data, with no dependence on a pre-built map. Many other studies combine *RL* with classical planners: the global planner generates a coarse overall path, and *RL* handles local dynamic obstacle avoidance to form a hierarchical hybrid architecture.

Nevertheless, path planning with *RL* has notable drawbacks. There are three main challenges. First, training requires a huge number of interaction samples. Training directly on physical robots costs too much, while simulation introduces the "reality gap" between simulation and the real world. Second, reward function design relies heavily on domain knowledge, and poor design easily leads to invalid policies. Third, learned policies may not generalize reliably to new environments, which is an unavoidable issue for safety-critical applications.

3.4. Path planning methods based on deep learning

The ability of deep learning to extract features from high-dimensional data has been fully verified in computer vision, natural language processing and other fields. In recent years, this technology has gradually reshaped path planning and brings new opportunities for joint modelling of perception, decision-making and planning.

The most representative application of deep learning in path planning is end-to-end navigation. The neural network directly maps raw sensor data such as images and point clouds to control commands or waypoints, skipping the separate modular steps of perception, mapping and planning in traditional pipelines. *CNN* processes visual inputs, while *RNN* or *LSTM* captures temporal dependencies. This end-to-end paradigm achieves strong performance in tasks including lane keeping, indoor navigation and off-road driving.

Another noteworthy direction is imitation learning. Robots observe navigation demonstrations from humans or expert planners and learn to replicate such behaviours. Compared with *RL*, its biggest advantage is that reward function design is unnecessary. Behaviour cloning and generative adversarial imitation learning (*GAIL*) are typical methods in this category. Besides, deep learning can assist traditional planners. For instance, trained networks predict heuristic values for *A** to speed up search; *CNN* identifies promising sampling regions for *RRT* to ease difficulties in narrow passages; graph neural networks (*GNN*) learn environmental representations for multi-robot scenarios.

Deep-learning-based path planning has three major strengths: strong perception for unstructured environments, the capacity to learn from large-scale datasets, and smooth integration with perception modules such as object detection and semantic segmentation. Its weaknesses are also

DOI: 10.54254/2755-2721/2026.38719

clear: it requires massive labelled data, suffers from low model interpretability, and produces unpredictable behaviours when encountering out-of-distribution cases. From the author's perspective, ensuring the safety and reliability of learned planners is the most critical unsolved problem at present.

4. Discussion and future development trends

4.1. Comparative analysis of different methods

Reviewing the content in the previous two sections reveals an interesting observation: different categories of methods differ far more than they resemble one another. Table 1 summarizes representative methods and compares them from four dimensions: representative algorithms, advantages, limitations and applicable scenarios.

Table 1. Comprehensive comparison of representative path planning methods

Method Category	Representative Algorithms	Main Advantages	Main Limitations	Applicable Scenarios
Graph-search-based	Dijkstra / A* / D*	Complete and optimal, solid theoretical foundation	High memory consumption, poor adaptability to dynamic scenarios	Known static maps
Artificial potential field based	APF and its improved variants	Easy implementation, smooth trajectories, favourable real-time performance	Prone to local minima, GNRON problem	Local real-time obstacle avoidance
Random sampling based	PRM / RRT / RRT*	Suitable for high-dimensional spaces, probabilistic completeness	Suboptimal solutions or slow convergence, poor performance in narrow passages	Robotic manipulators, unmanned aerial vehicles
Intelligent optimization based	GA / PSO / ACO	Strong global search capability, support for multi-objective optimization	Slow convergence, sensitive to parameters	Offline global planning
Reinforcement learning based	Q-learning / DQN / DDPG / PPO	No requirement for environmental models, adaptive to dynamic scenarios	High training cost, simulation-to-reality gap	Dynamic and unknown environments
Deep learning based	CNN / RNN / GNN / imitation learning	Powerful perception ability, end-to-end learning	Reliance on massive datasets, low interpretability	Unstructured environments

Several conclusions can be drawn from Table 1. Traditional methods feature clear theories and predictable behaviours, yet they struggle in dynamic, high-dimensional or unstructured settings.

DOI: 10.54254/2755-2721/2026.38719

Intelligent methods achieve much better adaptability, while interpretability, sample efficiency and safety guarantees become new obstacles. In other words, no single category of methods can adapt to all circumstances. Algorithm selection must take practical factors into full account, including the dimension of configuration space, environmental dynamics, computing resources and safety requirements. For this reason, many practical systems adopt hierarchical or hybrid architectures where different methods perform their respective functions. This scheme has become the mainstream in the industry.

4.2. Future development trends

Combined with the above analysis and recent research progress, six promising research directions worthy of attention are elaborated as follows.

(1) Hybrid planning frameworks. The integration of traditional and intelligent methods is one of the most popular research directions. A typical scheme adopts *A** or *RRT** for global planning to guarantee completeness, and reinforcement learning policies for local planning to handle emergencies. This design retains theoretical guarantees while gaining flexible adaptability.

(2) Multi-robot coordination. More and more tasks in warehousing and logistics, intelligent manufacturing, search and rescue and other fields rely on coordination among multiple robots. It brings a set of new challenges: conflict avoidance, efficient communication and truly decentralized decision-making. Multi-agent reinforcement learning (*MARL*) and graph neural networks show encouraging potential in this area, but they are far from mature.

(3) Learning-based planning under uncertainty. Real-world environments are never perfect. Sensors contain noise, obstacles move, and available information is always incomplete. How can learning-based planners actively perceive and cope with such uncertainty? Bayesian deep learning, probabilistic policy networks and risk-aware planning are feasible solutions. This step is indispensable for practical engineering deployment.

(4) Semantic-aware navigation. Obstacle avoidance alone is no longer sufficient for robots. Service robots need to distinguish different rooms, and autonomous vehicles need to infer the intention of surrounding vehicles. Integrating semantic segmentation, scene graphs and natural language understanding into the planning pipeline is critical for more intelligent and human-friendly navigation. This direction holds great potential and may witness major breakthroughs within the next five years.

(5) Simulation-to-reality transfer and lifelong learning. "Training in simulation before deployment in reality" has become a de facto standard workflow. Technologies including domain randomization and meta-learning still have large room for improvement to reduce transfer errors. The goal of lifelong learning is to enable robots to accumulate experience and optimize policies throughout their service life, or simply put, "perform better with more use". This prospect is appealing, yet its implementation faces considerable challenges.

(6) Planning in three-dimensional and unstructured environments. Existing research mainly focuses on two-dimensional planes, while applications such as unmanned aerial vehicles, underwater detectors and planetary rovers require path planning in complex three-dimensional terrain. Efficient representation of three-dimensional space is a key issue. Octrees and neural implicit fields have been explored by researchers. Combining these representations with advanced planning algorithms is essential to break through this bottleneck.

DOI: 10.54254/2755-2721/2026.38719

5. Conclusion

This paper systematically reviews path planning methods for mobile robots. On the side of traditional methods, graph search algorithms (*Dijkstra*, *A**, *D**), artificial potential field methods and sampling-based planners (*PRM*, *RRT*, *RRT**) are discussed in detail. With solid theories and predictable behaviours, these methods serve as a reliable foundation for engineering implementation. On the side of intelligent methods, optimization approaches including genetic algorithms, particle swarm optimization and ant colony optimization, as well as reinforcement learning and deep learning, are covered. They feature strong adaptability at the cost of interpretability and safety. A straightforward conclusion drawn from the comparison is that no single method outperforms others across all scenarios. Practitioners need to select algorithms comprehensively according to the dynamics, dimensionality and safety requirements of the application environment.

This paper has two main contributions. First, it provides a relatively complete comparison of methods for researchers and engineers, and clearly states the applicable conditions of various algorithms to support selection decisions. Second, six potential research directions are proposed based on current progress: hybrid planning frameworks, multi-robot coordination, navigation with uncertainty perception, semantic understanding, simulation-to-reality transfer, and three-dimensional planning. These directions are not isolated; in fact, substantial overlaps exist among them.

Admittedly, this review has several limitations. Restricted by page length, only the most representative algorithms in each category are elaborated, while many newly proposed improved variants in recent years are only briefly mentioned. The analysis is mainly qualitative. Quantitative comparisons on standardized benchmark datasets can be carried out in follow-up studies. Even so, it is hoped that this review can assist newcomers entering this field and, to some extent, promote the further development of mobile robot navigation technology.

References

[1] Patle, B. K., Babu, G. L., Pandey, A., Parhi, D. R. K., & Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. *Defence Technology, 15*(4), 582–606. https://doi.org/10.1016/j.dt.2019.04.011

[2] Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., & García-Cerezo, A. (2021). Path planning for autonomous mobile robots: A review. *Sensors, 21*(23), Article 7898. https://doi.org/10.3390/s21237898

[3] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*(1), 269–271. https://doi.org/10.1007/BF01386390

[4] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics, 4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

[5] Chi, X., Li, H., & Fei, J. Y. (2021). Research on robot random obstacle avoidance method based on fusion of improved A* algorithm and dynamic window method. *Chinese Journal of Scientific Instrument, 42*(3), 132–140.

[6] Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation* (pp. 3310–3317). IEEE.

[7] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research, 5*(1), 90–98. https://doi.org/10.1177/027836498600500106

[8] Ge, S. S., & Cui, Y. J. (2000). New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation, 16*(5), 615–620. https://doi.org/10.1109/70.880813

[9] Kavraki, L. E., Švestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation, 12*(4), 566–580. https://doi.org/10.1109/70.508439

[10] LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning* (Technical Report No. 98-11). Department of Computer Science, Iowa State University.

DOI: 10.54254/2755-2721/2026.38719

[11] Lin, Y. F., Chen, Y. J., He, B. W., et al. (2020). Motion planning method for mobile robots using RRT without collision detection. *Chinese Journal of Scientific Instrument, 41*(10), 257–267.
[12] Holland, J. H. (1992). Genetic algorithms. *Scientific American, 267*(1), 66–73.
[13] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks (Vol. 4)*, 1942–1948. IEEE. https://doi.org/10.1109/ICNN.1995.488968
[14] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26*(1), 29–41. https://doi.org/10.1109/3477.484436
[15] Shi, E. X., Chen, M. M., Li, J., et al. (2014). Research on global path planning methods for mobile robots based on the ant colony algorithm. *Transactions of the Chinese Society for Agricultural Machinery, 45*(6), 53–57.
[16] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.
[17] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236
[18] Tai, L., Paolo, G., & Liu, M. (2017). Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 31–36. IEEE. https://doi.org/10.1109/IROS.2017.8202134