

# A critical review of analytical approaches to path planning for autonomous mobile robots

Suhaib Al-Ansarry<sup>a</sup>, Salah Al-Darraj<sup>a</sup>, Asmaa Shareef<sup>a</sup>, Hamid Ali Abed AL-Asadi<sup>b</sup>,  
Zaid Ameen Abduljabbar<sup>b,c,d</sup>, Hothefa Shaker Jassim<sup>e</sup>, Vincent Omollo Nyangaresi<sup>f,g</sup>,  
Korondi Péter<sup>h,i,\*</sup>, Husam A. Neamah<sup>h,i,\*</sup>

<sup>a</sup> Department of Computer Science, College of Computer Science and Information Technology, University of Basrah, Basrah, 61004, Iraq

<sup>b</sup> Department of Computer Science, College of Education for Pure Sciences, University of Basrah, Basrah, 61004, Iraq

<sup>c</sup> Department of Business Management, Al-Imam University College, Balad 34011, Iraq

<sup>d</sup> Shenzhen Institute, Huazhong University of Science and Technology, Shenzhen 518000, China

<sup>e</sup> Modern College of Business and Science, Muscat, Oman

<sup>f</sup> Department of Computer Science and Software Engineering, Jaramogi Oginga Odinga University of Science and Technology, Bondo 40601, Kenya

<sup>g</sup> Department of Applied Electronics, Saveetha School of Engineering, SIMATS, Chennai, Tamilnadu, 602105, India

<sup>h</sup> Department of Electrical Engineering and Mechatronics, Faculty of Engineering, University of Debrecen, Debrecen, 4028, Otemeto u.4-5, Hungary

<sup>i</sup> Technical Engineering College, Al-Ayen University, Thi-Qar 64001, Iraq

## ARTICLE INFO

### Keywords:

Intelligent approaches  
Optimization methods  
Path planning  
Robot navigation  
Decision analysis  
Navigation decision-making

## ABSTRACT

In this research, strategies for mobile robot navigation are studied and discussed critically, especially on the development of path planning methods under various environments and constraints. It discusses classical and intelligent approaches in detail to draw out their merits, demerits and scope for future research. Classical approaches investigated are Dijkstra, A-star, D-star, D-star lite, roadmap approaches and cell decomposition. Intelligent approaches studied include the particle swarm optimization algorithm, the firefly algorithm, the shuffled frog leaping algorithm, the grasshopper optimization algorithm, the bacterium foraging optimization algorithm, the bee colony optimization algorithm, the ant colony optimization algorithm, the grey wolf optimization algorithm, the red wolf optimization algorithm, and neural networks, genetic algorithms, fuzzy logic, and potential fields. Besides, the current review also includes an in-depth examination of the contemporary methods of machine-learning-based approaches, which includes the Reinforcement Learning (RL), the Deep RL, Graph Neural Networks (GNNs), and the Transformer-based methods, which occupy a dominant role in the current academic research. The review addresses navigation in both single-robot and multi-robot systems, across static and dynamic environments. The findings indicate that intelligent strategies offer greater flexibility and effectiveness in diverse terrains compared to classical methods. Moreover, intelligent approaches are often employed to enhance classical techniques through the development of hybrid algorithms. Overall, mobile robot path planning increasingly relies on intelligent approaches. The exploration is climaxed with the presentation of an intense comparative framework that includes new tables and figures that define the execution, computational complexity and flexibility of different navigation strategies. The visualizations support the relevance of the strategies in particular robotic cases, and explain the significant research paths.

## 1. Introduction

Initially, Autonomous mobile robots were mostly utilized in manufacturing industries. However, their usage has significantly expanded in several industries including entertainment, healthcare, mining, search & rescue, education, military, space exploration, agriculture. While

achieving navigation tasks, these robots are equipped with complex technologies that allow them to analyze their surroundings, determine their position, control their movement, detect impediments, and utilize sophisticated methods to evade obstacles. The main objective of any navigational approach is its safe trajectory planning that is the

\* Correspondence to: University of Debrecen, Faculty of Engineering, Otemeto u.4-5, Debrecen, 4028, Hungary.

E-mail addresses: [suhaib.alansarry@uobasrah.edu.iq](mailto:suhaib.alansarry@uobasrah.edu.iq) (S. Al-Ansarry), [aldarraj@uobasrah.edu.iq](mailto:aldarraj@uobasrah.edu.iq) (S. Al-Darraj), [asmaa.shareef@uobasrah.edu.iq](mailto:asmaa.shareef@uobasrah.edu.iq) (A. Shareef), [hamid.abed@uobasrah.edu.iq](mailto:hamid.abed@uobasrah.edu.iq) (H.A.A. AL-Asadi), [zaid.ameen@uobasrah.edu.iq](mailto:zaid.ameen@uobasrah.edu.iq) (Z.A. Abduljabbar), [hothefa.shaker@mcbs.edu.om](mailto:hothefa.shaker@mcbs.edu.om) (H.S. Jassim), [vnyangaresi@jooust.ac.ke](mailto:vnyangaresi@jooust.ac.ke) (V.O. Nyangaresi), [korondi.peter@eng.unideb.hu](mailto:korondi.peter@eng.unideb.hu) (K. Péter), [husam@eng.unideb.hu](mailto:husam@eng.unideb.hu) (H.A. Neamah).

<https://doi.org/10.1016/j.dajour.2026.100722>

Received 17 August 2025; Received in revised form 18 May 2026; Accepted 28 May 2026

Available online 4 June 2026

2772-6622/© 2026 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license

(<http://creativecommons.org/licenses/by/4.0/>).

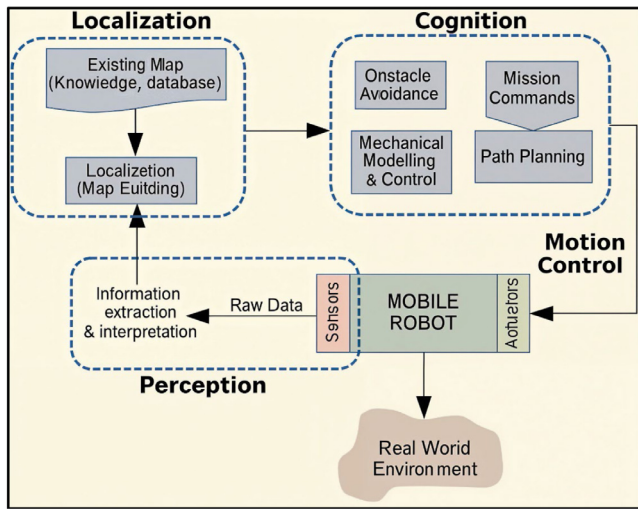


Fig. 1. Block diagram of mobile robot navigation.

recognition and avoidance of obstacles as a robot moves from its source point to its target location. Therefore, choosing the right navigational technique is essential when scanning environments (simple and complex). The navigation of mobile robot has been an area of considerable research, and scientists have addressed different problems of it.

The classification of navigation includes three types: global, local, and personal navigation. Global navigation refers to the process of identifying the relative position of things inside a referent place with respect to an inertial frame of reference and enabling the motion towards a target destination. Local navigation is the ability to determine and track the positions of various objects in relation to each other in dynamic environments. Personal navigation assigns and organizes special relations between items in the environment, and their locations. Fig. 1 shows the main steps in operation of a robot [1]. Here, we classify navigation strategies into two major classes: global and local navigation strategies. They are classified according to how much information about the environment existing before planning the path. Global navigation is based on the prior knowledge of the environment, like the exact position of obstacles as well as goals; Local navigation does not depend on this knowledge. Global navigation is appropriate for familiar environments, whereas local navigation is useful for traveling in unfamiliar or poorly understood territory.

Typical algorithms for path planning in scenarios are classical algorithms such as the Roadmap Algorithms (RM), Artificial Potential Fields (APF) and Cell Decomposition (CD). Such traditional methods have considerable but limited cognitive capability. Intelligent methods, as outlined in this review, are a sub-group of algorithms that have applied concepts of artificial intelligence, machine learning and computational intelligence to help enable adaptive, robust and often optimal path planning within environments of complexity and uncertainty. The methodologies outlined below namely: evolutionary algorithms, neural networks, fuzzy logic and swarm intelligence are distinguished by the ability to learn data, to compete with non-linearities and to work successfully in incomplete information conditions.

Although there are several survey papers on mobile robot navigation [2,3], they do not provide an in-depth investigation about sound navigation methods. In spite of the fact that the previous surveys have provided us with an excellent perspective of the field, they have been more focused on specific sub-domains, including 3D path planning [3] or data-driven control approaches [2]. On the contrary, the review provides a comprehensive, and critical summary of the classical and intelligent methods, with a strong focus on the emerging role of modern machine learning methods. Providing a strict comparative framework

and clarifying the gaps in the existing research, this manuscript will aim to provide a more robust and prospective view on the issue of the current state of the art.

This review paper therefore attempts to highlight some yet unexplored research topics and identify some novel opportunities in a particular area of mobile robot navigation. We present a comprehensive account of various algorithms designed for both static and dynamic settings, where mobile obstacles and goals are taken into the account. The study consists of simulations, experimental evaluations and various scenarios related to the navigation of mobile robots. It also includes an adaptation of multiple intelligent techniques, navigation in 3D environments, and military or defense use.

The review emphasizes the differences between classical and intelligent methods with regard to their effectiveness and the context in which they apply — in aviation, on land, underwater, in industry and hazardous environments. The main topic of [2] is on the use of intelligent techniques which rely on local sensor information, sliding mode control and decentralized model predictive control (MPC) techniques. On the contrary [3] focuses on 3D path planning. The next step in this study, Path Planning Approaches, then categorizes navigation approaches into classical and intelligent approaches, and a detailed discussion of the literature review is presented in Section 3. Section 2 of this study the Path Planning Approaches then classifies navigation strategies into classical and intelligent approaches, while Section 3 presents a comprehensive discussion of the literature review. Section 4 introduces a new critical analysis and synthesis of the reviewed methods. The conclusion finally in Section 5.

## 2. Path planning approaches

Throughout many decades, a plethora of scholars and scientists have presented a wide array of solutions for navigational methodologies. The methods employed for directing a mobile robot can be roughly categorized into two main groups: classical and intelligent approaches (refer to Fig. 2). The strategies of path planning for mobile robots presented in this review are classified according to a structured methodology that considers the several factors of computational complexity, efficiency, and adaptability of the algorithms and approaches. It reviews classical approaches, as well as intelligent approaches, including comparative performance metrics.

### 2.1. Classical approaches

In the early stages, conventional methods were frequently employed to tackle the challenges in robot navigation, as more advanced artificial intelligence technologies were not yet accessible. When using traditional ways to perform a task, it was seen that either the required result would be achieved or it would be determined that the desired result was not possible. This approach, however, has a major limitation in the form of high computation requirements and poor ability to respond quickly to environmental variables, thus limiting the applicability in a real-time context. This section briefly summarizes the classical methods, namely Cell Decomposition (CD), Artificial Potential Field (APF) and Roadmap (RM) procedures.

#### 2.1.1. Deterministic methods

##### The Algorithm of Dijkstra

The work provides comprehensive analysis of the Dijkstra algorithm [4]. This technique breaks down complex path finding breaking tasks down into smaller, easier challenges. The algorithm calculates the optimal route from the starting point to the neighboring locations [4]. The priority queue is used for Dijkstra's method to keep track of newly discovered vertices. It determines the closest vertex and keeps one interleaved node and thus it evolves as a singular shortest path. The Dijkstra algorithm is known to find the shortest path in a weighted graph, if the weights of all the edges are positive. This algorithm can

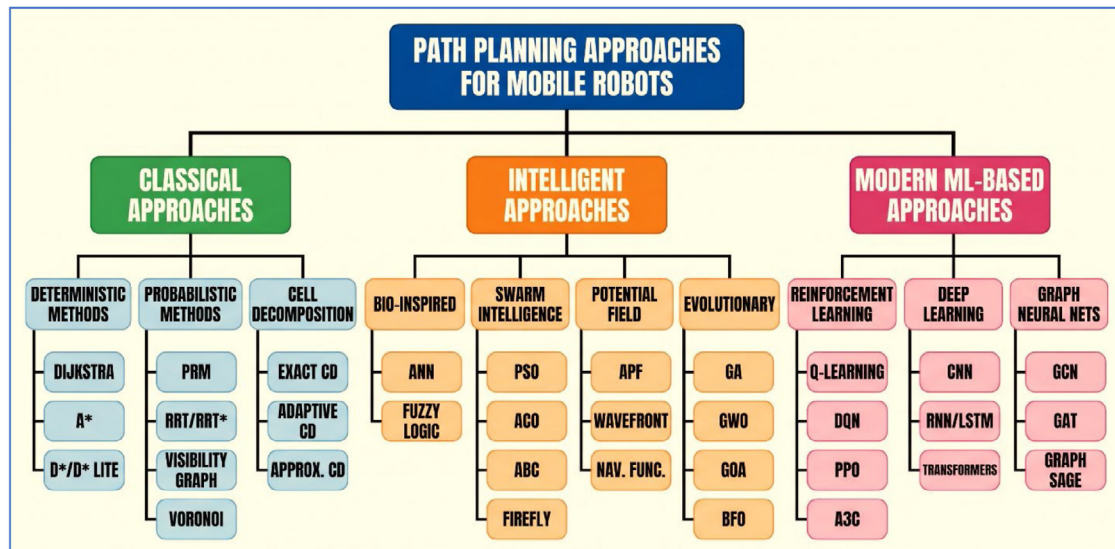


Fig. 2. Categories of robot path planning approaches.

be applied to graphs with cycles in it; although it cannot be applied to graphs with negative-weight cycles because they may lead to an infinite-loop within the algorithm.

Versions of Dijkstra algorithm are specially adapted to solve a specific application with various usages. Dijkstra and the standard version of it, utilized a greedy strategy to quickly count the path, tends to a solution the shortest path search directly. However, the vicinity of when a valid path might encounter and at what large cost is the focal point of the new form of the Dijkstra algorithm in which made use of gradually the integrated to the discovered positives with large cost [5]. This enhancement builds upon the standard technique, allowing it to be used in new and different situations. An enhanced version of Dijkstra's method tracks nodes that are at the same distance from the source node and cannot be direct connections, to mark them as intermediaries. Further investigations conducted on these nodes reveal all possible paths having minimum length [6], thus increasing the range of evaluation options. Dijkstra has an intrinsic limitation to access data after visiting the nodes. It uses a complex multi-tier dictionary structure to tackle this problem [7]. This is a combination of two dictionaries and an array of structured data. This system serves the purpose of mapping nodes with their respective nodes, and storing path information for each route.

The multi-layer dictionary is very useful in indoor areas, where the quality of Global Navigation Satellite System (GNSS) coordinates and Compass Orientation cannot be secured. This helps find the only angles required to be turned at the intersection, so the path with the lowest distance and least amount of the angle turns are found as the optimal path [7]. The Floyd algorithm is very high in performance for choosing the best way in a weighted graph and Dijkstra's algorithm has a great quality when solving single source shortest path problems in positive weighted graphs [8]. Though Dijkstra's Algorithm is famous for its reliability, it is limited by its large memory requirements and its inability to operate in the presence of negative edges. Due these limitations, several improved versions of the algorithm have been developed that are tailored to specific applications. We can conclude that the Dijkstra algorithm is most suitable for the scenarios requiring static/global path design as well as if there is a lot of required data that is predefined. It also has the potential to adapt to changing scenarios, utilizing either local or global planning to find the optimal path through dynamic obstacle information in real-time [9].

#### The A\* Algorithm

A\* algorithm is the most common methods for searching a graph and finding a path. The A\* algorithm has a similar operation to that

of Dijkstra's algorithm, but it distinguishes itself by limiting its search space to the most favorable states in general, which can lead to significant savings in terms of computational resources [10]. A\* is broadly recognized for its ability to achieve nearly optimal solutions [11] within a particular dataset or node pool. This is mainly used for stationary cases; however, it is also used in some active cases [12]. Depending on what the user wants, the main capabilities of A\* can be adjusted according to certain needs or particular situations. The A\* algorithm resembles Dijkstra's algorithm specifically that it builds a tree of the most direct routes from the initial location to its final destination. The main algorithm takes the least-cost path and grows it by applying the successor function in Eq. (1):

$$f(n) = g(n) + h(n) \quad (1)$$

Where the function  $f(n)$  is defined as  $g(n)+h(n)$ . Where  $g(n)$  the cost from node  $n$  to the starting node and  $h(n)$  the cost of the optimal path from the destination node to  $n$ . A\* applied widely in the gaming industry [13]. With the developments in artificial intelligence, refinements and adaptations of the A\* have been made for different applications such as robot pathing and finding, agricultural systems, visualization, and control systems-based applications [14–18]. Due to its computational simplicity and efficiency, this method is a good candidate for constrained and embedded systems and is one of the key advantages of the basic RRT compared with many other path planning algorithms [19, 20]. The A\* algorithm uses heuristic information to choose one of many potential paths. The method necessitates the mapping of nodes and the generation of suitable heuristic functions to assist in navigation, such as Euclidean distance, Diagonal distance, or Manhattan distance [21,22].

The algorithm's efficiency is dependent on two key aspects: How well the resources are used during the execution of tasks, and the time needed to complete each task, known as response or computation time. The time and accuracy of the process needs to be optimally balanced to implement A\* algorithm. This method can be made faster in context of time complexity at the cost of increased memory usage, or bends towards optimized memory usage with significant speed execution tradeoff. No matter what, we need to find the shortest path possible. At a crowded parking lot, the A\* algorithm can be utilized to compute the optimal path to the nearest available parking lot [20].

The hierarchical A\* algorithm breaks down the issue of path finding into separate steps, each will yield an optimal solution. These step-specific answers are combined to achieve the overall best solution, which is the fastest path to an unoccupied location. The method

has been further refined, resulting in an improved hierarchical A\* algorithm that uses best time as the evaluation measure.

A weighted sum variable is introduced alongside the heuristic as Eq. (2):

$$f_w(n) = (1 - w)g(n) + w * h(n) \quad (2)$$

Let  $w$  be a weight coefficient which represents the trade-off between the actual cost incurred since starting at the node, denoted  $g(n)$  and the estimated cost to the goal denoted  $h(n)$ . The coefficient is usually in the range  $[0, 1]$ . The larger value (e.g.  $w > 0.5$ ) gives preference to the heuristic, hence producing a faster but possibly less optimal search (e.g. a greedy best-first search). A smaller value (e.g.,  $w < 0.5$ ) on the other hand puts more stress on the actual cost and generates a slower but better path. At  $w = 0.5$  the behavior of the algorithm becomes the standard A\* algorithm.

A weighted linearly distance (like Euclidean) is commonly used to describe a weighted coefficient (which varies in accordance with the functionality for each application). The expanded version of the hierarchical A\* algorithm is noted for its greater efficiency and accuracy [23]. A variant of this approach, known as the Hybrid A\* method, is utilized in applications such as valet parking, where the trajectory of the ego-vehicle (the vehicles that are the focus in their surroundings) is sensitive to things such as speed, gear, and the steering wheel angle. The Hybrid A\* Algorithm reflects on the kinematics of the ego vehicle to estimate its movement and alters costs in the heuristic function. Different techniques have optimized the Hybrid A\* Algorithm and increased its performance [21–24].

Visibility graph is a graph-based search algorithm used in robotics path planning, where the aim is to find the shortest distance of the path from the source to the goal. It is used to predict the distance of the best route between the starting position, destination, and positions of obstacles (combined well with the A\* method). This algorithm is named the Guided Hybrid A\* algorithm [24].

The A\* algorithm, which is known for its efficiency in terms of computational cost compared to other alternatives for path planning. A\* is well-fitted for automotive applications considering vehicle kinematics, steering angle, and vehicle size. Eq. (3) shows that equal-step sampling is used to create new cost functions. The cost function combines the distance and steering costs — where only the distance traveled incurs costs but there is no penalty for moving. Taking these characteristics into account helps to reduce sudden directional changes, which can improve the smoothness of the path [25]. Considering the provided equation:

$$f(n) = k_1 g(n) + k_2 h(n) + k_3 p(n) \quad (3)$$

where  $k_1$ ,  $k_2$ , and  $k_3$  represent positive weights, whereas  $g(n)$  and  $h(n)$  denote the constituent elements of the A\* function. The variable  $p(n)$  denotes a penalty factor that is linked to the expense incurred when making a turn.

Lifelong Planning A\* (LPA\*) [26] combines the best elements of both incremental and heuristic search methods to be most efficient. What makes this method so effective is its ability to find path(s) that are as optimum as possible even when there are some variations in the cost associated with the connections in a graph or when some nodes are blocked or removed by obstacles. The rhs values of the algorithm have been set to infinity for all the vertices encountered and computed with the use of Eq. (4), thus, starting an A\* search process.

$$rhs(s) = \begin{cases} 0, & \text{if } S = S_{start} \\ \min_{\hat{s} \in pred(s)} (g(\hat{s}) + c(s, \hat{s})), & \text{otherwise} \end{cases} \quad (4)$$

During the subsequent stages, the algorithm adjusts the rhs-values and keys of vertices that are impacted by alterations in edge costs, in accordance with Eq. (5). Membership status is determined by evaluating the degree of local coherence inside a priority queue. Furthermore, the approach reassesses the optimal path from the initial location to the target by giving priority to expanding vertices that possess competing

g-values and rhs-values. This process persists until the target vertex achieves a state of local consistency, and the key value of the next vertex to expand surpasses the key value of the objective. It is essential to recognize that while using the LPA\* approach, it is imperative to re-plan the path from the initial position to the desired destination [27].

The algorithm possesses an exceptional attribute of amazing efficiency, surpassing the performance of the two individual search algorithms. The search process is made effective by reusing identical parts of the previous search tree in the current tree, while also using heuristic knowledge to guide the search.

$$K(s) = [\min(g(s), rhs(s)) + h(s); \min(g(s), rhs(s))] \quad (5)$$

The A\* algorithm is somewhat computationally efficient compared to numerous other planning algorithms. This feature makes it suitable for use in fixed environments. The efficacy of A\* and its modifications relies on the precision of the heuristic function employed [28,29].

### The D\* Algorithm

The development of the D\* (Dynamic A\*) algorithm was motivated by the crucial need for effective path planning in scenarios where only partial information is available and the environment is subject to change, namely for autonomous vehicles. The D\* method [30] calculates collision-free paths in the presence of moving obstacles. The informed incremental search technique may rectify both partially processed and previously computed cost maps.

The D\* method continues to iterate over a robot's state until it is no longer included in the open list. Simultaneously, the sequence of states is computed, which includes back pointers that direct the robot towards its destination or modify pricing based on recognized obstacles. The affected states are listed on the public roster. States in the open list are assessed until the cost of the path from the present state to the aim becomes less than a predetermined threshold. The succeeding state is modified by cost changes, and the robot navigates towards the goal by following back pointers in the new sequence [31]. D\* demonstrates a speed superiority exceeding 200-fold when compared to an optimal replanner [31,32]. However, the D\* method is impeded by its significant memory usage when compared to other forms of D\* [33].

### The D\* Lite Algorithm

The D\* Lite method [34] integrates the heuristic and iterative search techniques to address a range of navigation issues in a dynamic environment. The algorithm solely focuses on navigating around obstacles that are present at the beginning or arise while the robot is in motion, excluding consideration of obstacles that are themselves in motion. The D\* Lite improves the LPA\* algorithm by systematically inverting edges and exchanging the starting and target vertices. This improvement enables D\* Lite to compute the optimal path from the destination node to the origin node in a dynamic and unfamiliar environment. The D\* Lite method is more proper for quickly recalculating the paths of self-driving vehicles in intricate scenarios when they encounter unexpected obstacles [35].

Enhanced D\* Lite improves the obstacle avoidance properties of D\* Lite, through the elimination of movement between obstacles, the creation of virtual obstacles if needed and the elimination of redundant paths. The improvements help achieve the most efficient path from the destination to the origin [36]. The development of Enhanced D\* Lite was done using an actual system, using an AmigoBot equipped with sonar sensors for obstacle detection [36].

Field D\* is an improved version of the D\* variant that used linear interpolation to create globally seamless and cost-effective routes. Therefore, Field D\* is quite suitable for practical applications in robotic systems for working in various environments such as indoors and very harsh outdoor environments [37]. To summarize, D\* and its modifications are versatile algorithms that can be used for any problem that involves optimizing path costs, especially when the costs change while looking for the most optimal path to the goal. The performance of D\* Lite is outstanding when these adjustments are detected near the current node in the exploration area. This approach can be used in various situations, such as when designing missions for planetary rovers [38].

### 2.1.2. Probabilistic methods

#### Roadmap Algorithms

Roadmaps entail the incorporation of many curves of one-dimensional into the configuration space, with the objective of linking the starting and destination points by traversing all the points along these curves. After establishing these sites, many routes are built to form a “Roadmap”, which is then used to select the most optimal route. If there is a single, continuous pathway connecting the starting and ending points, then construct a connection between the two locations to reach the end point. If there are many possible routes from the beginning point to the destination, Dijkstra’s method will be used to determine the most optimal path. There are various sorts of techniques used for creating roadmaps, like the Visibility Graphs, Voronoi Diagrams, Probabilistic Roadmaps, and Rapidly Exploring Random Trees.

#### a- Probabilistic Roadmap (PRM)

The Probabilistic Roadmap (PRM) technique [39] uses the configuration space to create roadmaps, enabling effective track planning for autonomous robots in large and multi-dimensional environments. The methodology depicted in Fig. 3 consists of two separate stages: the creation stage and the inquiry stage. During the construction phase, a roadmap is generated by employing a random sampling technique on the configuration space. The objective is to find regions devoid of obstacles that can be further investigated. Once these places are identified, the remaining vacant spaces are filled. In addition, points that are above a predetermined threshold are linked to their adjacent points. The probability of collisions is low with this approach, as any point that lies within the obstacle’s region will be ignored. During the query phase, an environmental map can be created, enabling the identification of the most economical route between the initial and destination places. This is accomplished by employing a unique and inexpensive search technique that is exclusive to our company.

One intrinsic benefit of PRM is its high likelihood of discovering a viable route, assuming that one exists. Nevertheless, a drawback of the PRM is its tendency to produce points randomly, leading to a diminished probability of producing points within small and narrow areas. Another constraint is the requirement for an exact description of obstacles, and the computation of path-cost is both costly and difficult in comparison to alternative approaches, owing to the utilization of binary impediments in PRMs. Furthermore, another issue is the need to revise the entire roadmap in case of any changes in the environment. Several different versions of PRMs exist, one example being the model suggested by Chen et al. [40] in 2025.

#### b- Rapidly-Exploring Random Trees (RRT)

In contrast to algorithms like A\*, which are specifically created for stationary scenarios and depend on a pre-established route, we will now investigate dynamic and online algorithms such as RRT [41,42]. The RRT has the distinctive capability to dynamically expand and investigate many regions without the need for a pre-established path. A path is established from the starting point to the destination by considering nodes that have been assigned weights. RRTs were specifically created to efficiently tackle different path planning issues, with a specific focus on overcoming non-holonomic constraints. It is difficult to integrate these limitations into positional constraints. Both RRT and PRM possess advantageous attributes. Both methods were devised without significant dependence on heuristics or arbitrary parameters, leading to enhanced performance and reliable results.

PRM often involves establishing multiple connections between states (configurations) to find a solution, while RRT can obtain a solution without needing any such connections. RRT is very suitable for Non-holonomic and Kino-dynamic planning because of this attribute [43].

The expansion technique of RRT involves rapidly selecting random points in the space, beginning at a specific point and progressively expanding until the tree closely approximates the intended goal point.

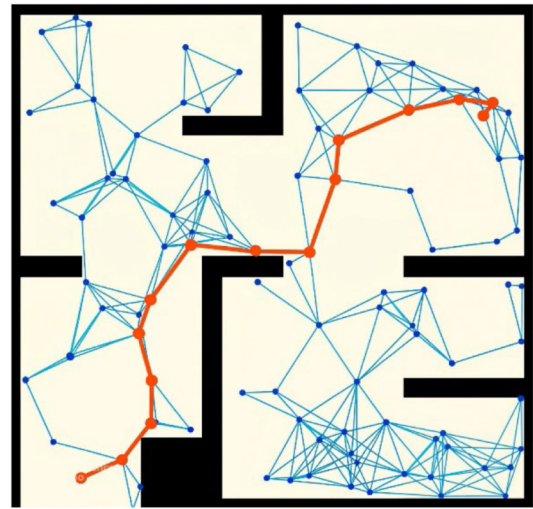


Fig. 3. Path construction of PRM algorithm.

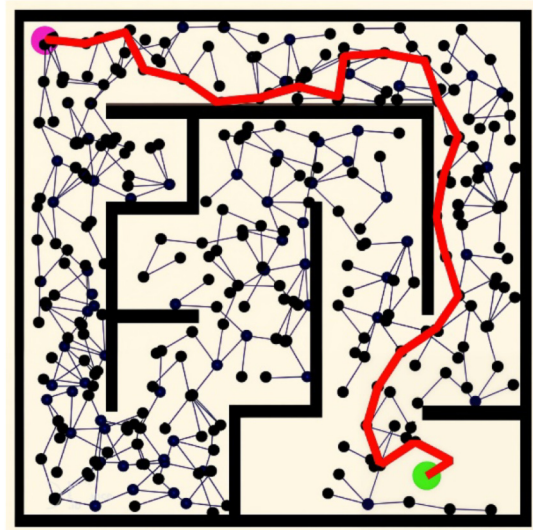


Fig. 4. Path construction of RRT algorithm.

During each cycle, the tree extends towards the nearest node of the randomly generated node. The selection of the nearest vertex is decided by a distance metric, such as Euclidean, Manhattan, or any other applicable metric. RRT differ from naïve random trees in that they mostly expand into unexplored locations of the robot’s configuration space, instead of locations that have already been inspected. For visual representation, please consult Fig. 4.

Consequently, RRT typically chooses places that have not undergone thorough investigation. The distribution of RRT vertices is uniform. The technique is straightforward, as RRTs maintain connectedness by minimizing the number of edges. Given the flexibility of RRT algorithms to handle non-holonomic constraints, they can be applied to practically wheeled system. Users can choose a specific variant of RRTs that best suits their requirements [44]. A particular iteration is referred to as RRT Connect or bi-directional RRTs. This approach combines two RRTs – one starting from the initial position and the other from the target point – by constructing a connection between them using a heuristic. This specific version is particularly appropriate for matters that do not involve unequal limitations. In each iteration, a solitary tree is enlarged by linking the newly formed vertex to the closest vertex in the other

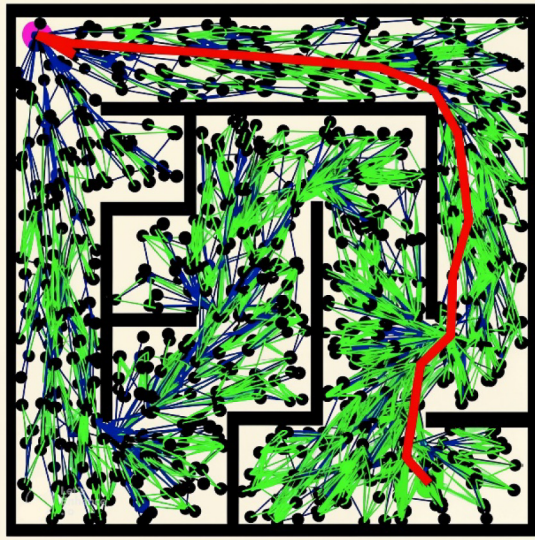


Fig. 5. Path construction of RRT\* algorithm.

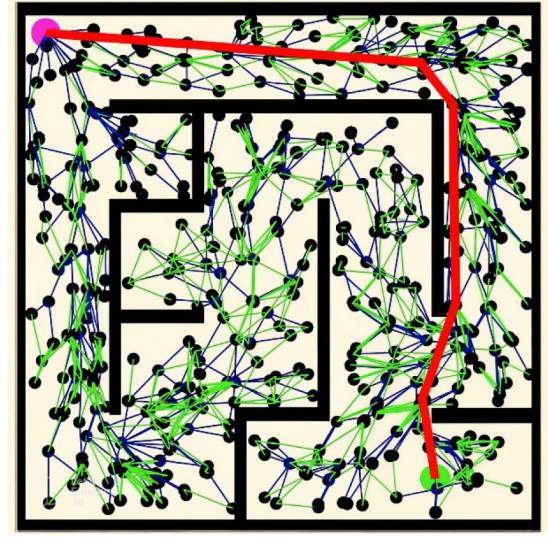


Fig. 6. Path construction of RRT\*-smart algorithm.

tree. As a result, the roles of the two trees are swapped, allowing both of them to explore the unbounded configuration space. This approach is particularly advantageous for planning the motions of robotic arms with a significant number of degrees of freedom [45].

Sensor-Based Random Trees (SRTs) are carried out to make a map of the confined studied area, including specific Safe Regions [46,47]. Sensors are used to detect Local Safe Region (LSR). Every SRT component consists of a unique structure in addition to its own Local Safe Region. The term “safe zone” is refer to the areas where there is no occupation around the robot which estimates empty space in a particular configuration. The shape of the LSR depends on the sensor characteristics of the robot, and it can either be spherical or star-shaped.

The growth of trees in SRTs occurs in directions that are chosen randomly, ensuring that both the new arrangement and the route to the test robot are enclosed inside the immediate safe area of the node. During each iteration, the robot gathers and analyzes sensor data to generate a plausible region that approximates the available space around its current configuration. Consequently, a new node is added to the tree, incorporating the present configuration and LSR. The choice of perception methodologies may vary depending on the sensors employed by the robot. Empirical evidence has demonstrated that the LSR exploration technique, characterized by a star-shaped arrangement, exhibits exceptional accuracy [46].

#### c- RRT\* Algorithm

RRT\* is a modified version of the RRT algorithm that seeks to discover the optimal route between an initial and target node by incorporating the concept of triangle inequality (see Fig. 5). With an increase in the number of nodes, the identification of more cost-effective (optimal) routes takes place. RRT-connect is particularly convenient for robotic arms, such as Programmable Universal Machine for Assembly (PUMA) robots [45]. The connection heuristic is most effective when the majority of planning questions are anticipated to involve areas that are relatively accessible or unrestricted. The link heuristic was initially formulated considering such scenarios [48].

#### d- RRT\*-smart Algorithm

RRT\*-smart [49] is an enhanced variant of RRT\* that operates in the same way as RRT\*. Even so, include an additional step of path improvement once the original path is discovered. This improvement procedure eliminates unwanted nodes from the original obtained path. Furthermore, it further detects beacon nodes as shown in Fig. 6 to enhance path optimization.

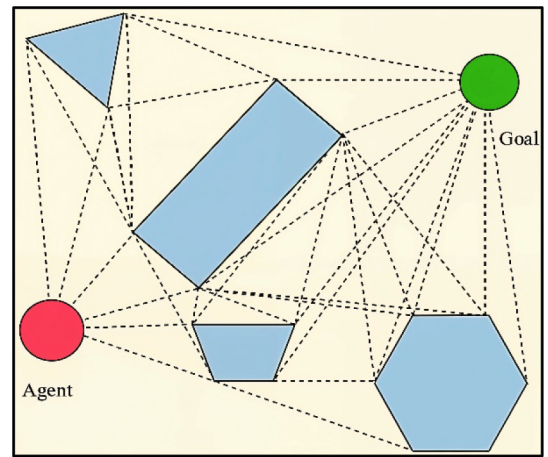


Fig. 7. Path construction of visibility graph.

The second main attribute implemented by RRT\*-smart is the incorporation of intelligent sampling. This method of sampling differs from random sampling as it reveals a biasing to beacon nodes which are a part of an optimal path. The Biasing Radius is used to determine the radius for intelligent exploration near a specific beacon. Whenever RRT\*-smart determines a shorter path, it begins the path improvement process once more in order to build fresh beacon nodes. Therefore, RRT\*-smart improves the speed of the convergence path and enhances the path cost. Moreover, it employs a Biasing Rate to intelligently choose between Intelligent Sampling and Uniform Sampling. RRT\*-smart demonstrates a more effective path compared to the RRT\* algorithm [50].

#### e- Visibility Graphs

NJ Nilsson [51], has invented the method of Visibility Graphs, which is considered as one of the initial methods for planning. It is employed for organizing areas that possess two dimensions. Fig. 7 depicts a graph consisting of shaded areas representing obstacles and lines connecting the vertices of these obstacles. These connections are impeded by barriers and eventually dictate the trajectory of the robot.

A fundamental limitation of this technique is that the paths it generates closely follow the obstacles, which means that the robot must have a safety margin around each obstacle to account for this. On

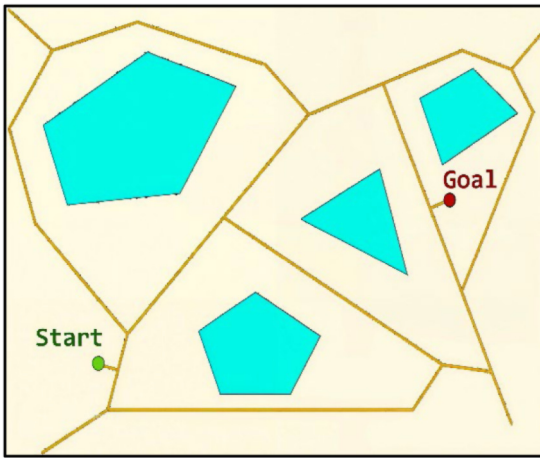


Fig. 8. Path construction of Voronoi Diagram.

the other hand, this results in the planner's inefficiency. Another issue emerges due to the irregular or circular shapes of obstacles, especially in outdoor environments, which makes the work more challenging.

#### f- Voronoi Diagrams

In 1644, René Descartes introduced this approach as one of the proximity diagrams. Subsequently, in 1908, Georgy Voronoi (Russian mathematician) made more enhancements to the method and successfully applied it to spaces of any number of dimensions. This method is considered to be one of the most used approaches in Roadmaps. The Voronoi Diagrams [52] employ a random distribution of points throughout the configuration space, followed by the creation of a polygon that encompasses each individual point. The resultant cell has a patchwork morphology. The determination of Voronoi edges that travel past barriers is ultimately based on the equidistant borders of the cells surrounding each obstacle. By employing this method, autonomous mobile robot can be navigate along a secure trajectory, in contrast to the visibility graph technique which entails the path intersecting with vertices of obstacles, hence jeopardizing the robot's motion.

Referring to Fig. 8, a graphical illustration of the Voronoi Diagram can be observed. Alterations to the path are required to address this problem and mitigate any possible conflicts. It should be emphasized that the Voronoi technique is inefficient, as it produces a suboptimal path.

#### Cell Decomposition Algorithms (CD)

The basic idea of Cell Decomposition [53] is to break the planning area down into smaller, specific, and dense segments known as cells. The whole configuration space is composed of these cells exactly and each one corresponds to a portion of the configuration space. The cells acquired from this procedure are adjacent. A graph of the relationships between neighboring cells is called a connection graph. The planner reads this graph and determines the path that the robot should take to get to the destination. The trajectory is a set of cells which the robot has to pass through. The price to visit cells is not necessarily the same for different metrics used to compute the optimal itinerary. The next section provides a detailed description of these operations, focusing on the Approximate, Adaptive and Exact cell decomposition methods.

##### a- The Approximate CD

An Approximate CD method break down each cell into four smaller cells that have the same shape but are smaller in size. The original proposal for this technique was made by Brooks and Lozano-Perez [54] and later developed further by Zhu and Latombe [55] and other scholars. The formation is created by overlaying a standardized grid across the entire workspace. The application is made easier by the predetermined proportions of the workspace. Hence, if an object exists in any part

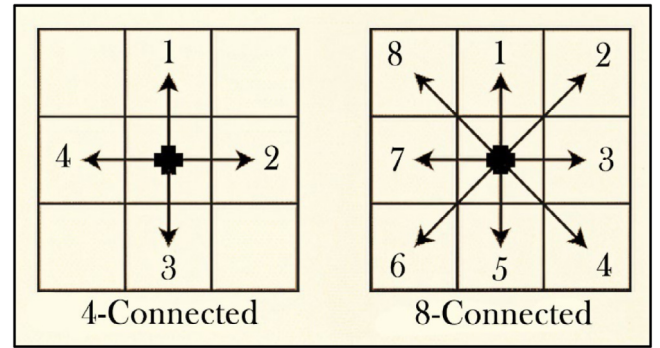


Fig. 9. Approximate CD (4-connected & 8-connected cells).

of the grid, the grid components within that part will be recognized as obstructed. Conversely, this region is considered to be a vacant territory. The nodes that need to be examined to find the path are denoted by the centroids of all grid cells in the search graph.

Fig. 9 demonstrates that these nodes can exhibit either 8-connectivity or 4-connectivity, and whether the robot can traverse them diagonally or not. CD is a method that collects samples from a graph and then assesses if the region is empty, partially populated, or completely inhabited, without specifically targeting any particular object. The term "approximate" is employed to characterize this method due to the lack of exact alignment between the boundaries of the object in the physical world and the grid on the workstation. This strategy is widespread due to the following reasons:

- Increasing the cell size would result in a reduction in computational time. Because there are just a small number of cells that need to be examined.
- Compared to other methods, this strategy is easier to use and can be used more easily.
- Applying for a job in the workspace provides increased flexibility.

Nevertheless, it is also burdened by other limitations. One of the challenges encountered is digital prejudice. In this methodology, if the dimensions of an obstacle are smaller than those of a grid cell (element), the entirety of the cell will be designated as occupied. Thus, utilizing these methods leads to the detection of multiple vacant spaces adjacent to the obstacles, which are impassable when trying to create a route. High population density and the presence of obstacles in the surroundings may greatly affect path planning and perhaps impede progress towards the goal. To tackle this problem, a more intricate grid is utilized, resulting in a necessary rise in both the processing time and storage needed for path planning. It is worth noting that this method can sometimes produce the same results as the Exact Cell Decomposition method below.

##### b- The Adaptive CD

The quadtree, initially suggested by Naniwa [56] and Samet [57], is the predominant form of Adaptive Cell Decomposition. Adaptive cell decomposition distinguishes itself from regular cell decomposition by maintaining the integrity of the cells while decreasing their dimensions using split and merge methods. This is done to make space for unrestricted areas adjacent to barrier sections. Using these spaces improves the mechanism of identifying the most efficient path, especially in expansive terrains. Moreover, it efficiently addresses the issue of digitization bias when utilizing traditional cell decomposition. As a result, the processing time and storage requirements are decreased. This strategy commences by constructing a substantial cell that encompasses the complete area that needs planning. When a cell is only partially occupied, it undergoes a process of division into four equal halves. The division technique is iteratively applied to each of these portions until the entire cell is either empty or fully occupied, as seen in Fig. 10.

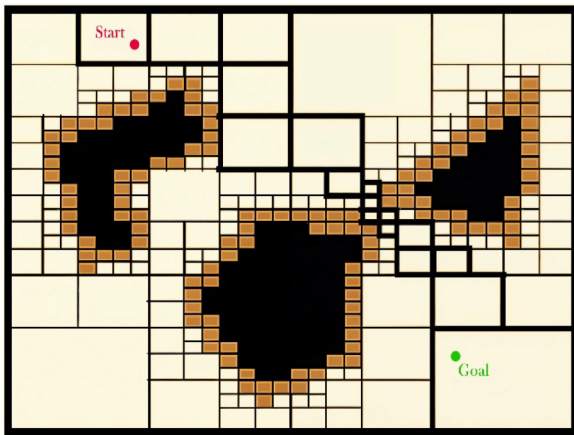


Fig. 10. Path construction of adaptive CD.

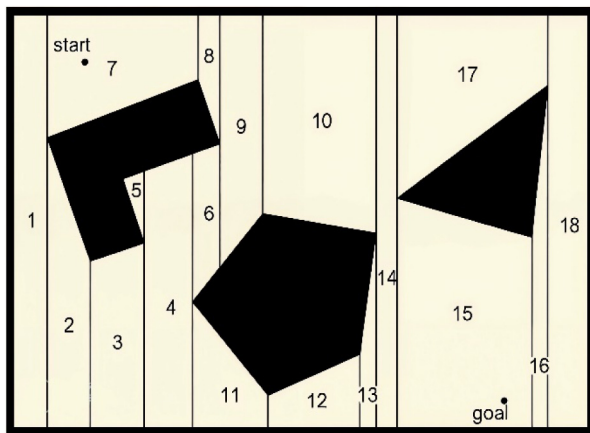


Fig. 11. Path construction of exact CD.

A challenge that arises with adaptive cell decomposition in dynamic environments is the necessity for the robot to update its environment map as it gains new information regarding the locations of impediments. It is imperative to meticulously restructure the map hierarchy in this situation. Quadrees encounter an extra challenge in attaining a path that is almost perfect and free from imperfections.

### c- The Exact CD

This method is based on the concept of dividing the available space into a collection of separate and non-overlapping portions, known as cells. Subsequently, the connectivity graph is generated and scrutinized. Upon the discovery of a path, a channel is formed, comprising a sequence of cells that establish a connection between the initial location in the cell configuration space and the desired destination. Fig. 11 illustrates the path of this channel through all of these cells. Debnath et al. [58] and Jan et al. [59] have presented multiple examples of the Exact Cell Decomposition.

Outdoor navigation can be difficult because of the restricted visual range, which complicates the process of identifying obstacles. Thus, it is more beneficial to employ it in indoor environments. The algorithm is unable to determine the barriers within a particular cell due to insufficient information regarding their shape and size. Nevertheless, by utilizing a map of the surroundings, the algorithm can precisely discern the form and positioning of items inside the given environment. This method has able to generate a route as long as there is at least one accessible, but it does not ensure that the resulting path will be the most optimal.

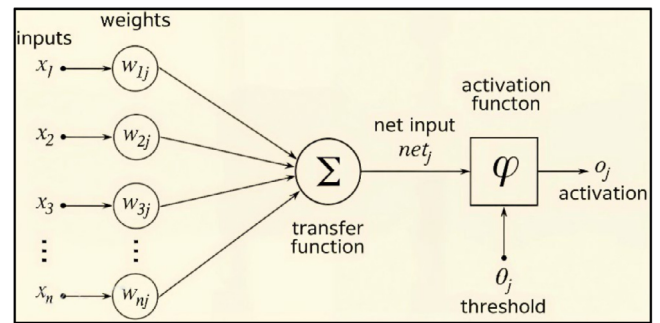


Fig. 12. Neural network architecture.

## 2.2. Intelligent approaches

Currently, advanced methods like genetic algorithms, fuzzy logic, neural networks, firefly algorithms, particle swarm optimization, ant colony optimization, bacterial foraging optimization, artificial bee colony, cuckoo search, shuffled frog leaping algorithm, and various other algorithms (such as bat algorithm, harmony search algorithm, etc.) have gained popularity for mobile robot navigation, passing the classical methods. These adaptive strategies exhibit exceptional abilities in effectively handling uncertainty in the environment. The subsequent sections delve into several responsive approaches.

### 2.2.1. Modern machine learning-based approaches

Over the last few years, the concept of robotic navigation has experienced an impressive shift in the incorporation of modern machine learning (ML) algorithms, especially based on the one of deep learning. In many cases these techniques offer better levels of flexibility and strength and have proven to be more effective approaches to learning in difficult and ambiguous real-world situations than more traditional techniques. This part offers a detailed examination of three approaches that are relevant today and are based on ML, namely Reinforcement Learning (RL) and Deep RL, Graph Neural Networks (GNNs) for spatial reasoning, and Transformer-based models for planning.

### Artificial Neural Network (ANN)

Artificial neural networks (ANN) are complicated systems made up of interconnected processing units that pass on information by means of dynamic state responses to external inputs. Neural Networks (NN) as illustrated in Fig. 12 are networks of nodes with activation functions. They exhibit traits like as the capacity for generalization, extensive parallel processing, distributed representation, the ability to learn, and resilience to faults.

The qualities that make NN suitable for the navigation of mobile robots have been successfully demonstrated by its use in different applications, including the navigation of wheeled robots [60], autonomous learning methods [61] and integration with simultaneous localization and mapping (SLAM) methods [62]. There have been studies on hybrid systems combining neural networks (NN) with other techniques such as NN-APF [63] or NN-FL [64] that allow for collaborative coordination and navigation in disordered environments for multiple mobile robots. NN has been used in path planning applications for three-dimensional (3D) or two-dimensional (2D) humanoid robots [65], industrial robots [66], ship hull cleaning robots [67] and aerial robots [68]. NN has shown promising applications in defense and aerospace industry, such as equipment guidance, navigation, targeting and control. It has been used extensively in practice in various areas including missile control [69], target detection [70] and selection of unmanned aerial vehicles (UAVs). [71]

### Genetic Algorithm (GA)

Genetic algorithm (GA) is an optimization method that has been widely used and is based on the principles of genetics and natural

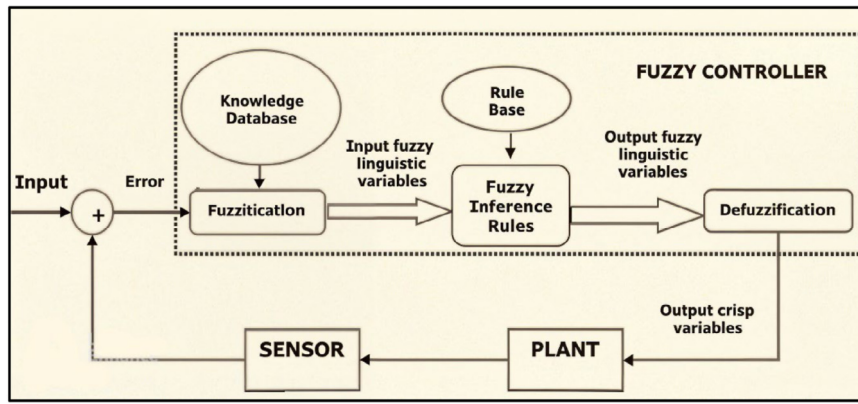


Fig. 13. Fuzzy logic controller.

selection. The concept of Genetic Algorithms was first proposed by Bremermann in 1958 [72] but computer science-based GAs was first introduced by Holland in 1975 [73]. Then, GA have been used widely in several diverse areas, such as robot navigation. Genetic algorithms are designed to solve a problem with an objective function that needs to be maximized or minimized considering certain constraints, which are complex problems.

The fitness of individuals in a population is judged by the objective function, which is based on the value of the population, and has a similar structure. They can choose to pass on the next generation their genetic material allowing for variety and not convergent too early due to crossover and mutation. Some applications of genetic algorithms have been seen in the case of mobile robot navigation in scenarios like static environments [74], online path planning [75], navigation in unknown regions [76] and handling non-linear obstacles in the environment [77]. In order to improve robot path planning, GA is not used alone, instead, combining GA with other intelligent algorithms in a hybrid technique such as GA-FL [78], GA-NN [79] and GA-PSO [80]. Genetic Algorithms (GA) have been used to solve the path planning for multiple mobile robots.

These algorithms improve collision avoidance [81], coordinate a multitude of goals [82], and navigate dynamic environments [83]. GA is beneficial, yet it has some limitations, like the slow convergence rate, uncertainty of optimal solutions, and complex design of mutation rates and population sizes. To surmount these limitations, scientists have implemented customized genetic algorithm path planner. These planners bet on co-evaluation mechanisms within the population [84] and adjusted chromosome length to improve the output [85].

The Genetic Algorithm has been applied in 3D path planning of underwater robots [86], aerial robots [87,88], and 2D path planning of humanoid robots [89] as it has been shown to perform well in multiple scenarios. It has also been used in defense technologies, such as missile guidance [90] and military and ocean surveillance [91].

#### Fuzzy Logic (FL)

Fuzzy logic (FL) [92] was first introduced by Zadeh in 1965, after that, it has been used in various research and development areas. Fuzzy logic is popular where uncertainty, complexity and nonlinearity are common, and has been investigated for use in navigation systems for various problems. Among them are obstacles avoidance, loop navigation, path retracing, dead-end escape, narrow passageway navigation, and navigating curves. The If-Then rules of a simple FL system are shown in Fig. 13.

FL makes use of human provided rules as “If-Then” statements which are then converted into the mathematical equivalents. “Once their equations are understood, system designers and computers can accurately analyze the behavior of real-world systems. Navigation methods developed in Florida have been presented for omnidirectional mobile robots [93], for obstacle avoidance [94], and in unstructured static

and dynamic environments [95–99]. Due to its high maneuverability and real-time decision-making ability [99], FL has been integrated with sensor-based [100], reinforcement-based [101], and algorithm-based navigation approaches involving neural networks (NN) [102], genetic algorithms (GA) [103], artificial potential field (APF) [104], and ant colony optimization (ACO) [105].

#### Potential Field Algorithms (PF)

Contrary to Cell Decompositions and Roadmaps, Potential Fields (PF) [106] aim to utilize a mathematical function throughout the full area of robot movement, rather than only mapping the search space. Fig. 14 depicts the fundamental notion that in any given environment, the robot must have both repulsive and attractive forces, which correspond to obstacles and targets, respectively.

The following section will provide a detailed explanation of several efficient strategies for designing the trajectory of potential fields. The next section will give a detailed explanation of several efficient ways of designing the trajectory of potential fields. Potential fields were initially not defined as a type of field explicitly for path planning purposes, but rather for real-time collision avoidance using the sensor data [107]. Thus, the best path to take to get to the desired position can be determined with a minimum amount of computation. The restriction on robots in local minima for particular potential fields results in less than optimal pathways, or difficulty finding the path. To overcome this issue, a navigation function [108] or a probabilistic navigation function [109] can be used.

#### a- Wavefront based PF

An improved variant of the Breadth-First search is the Wavefront method [110] shown in Fig. 15. This method can be split into two parts; the first part is reminiscent of the Flood Fill algorithm, and the second is similar to the Gradient Descent method. A robot navigation algorithm could be used in the robot path planning problem to avoid a small minimum area, that is, a grid-based navigation algorithm. This method involves assigning a cost to each cell location in the grid proportional to the distance to the goal; distance depends on how far away one is from the goal [111].

Once the Wavefront arrives at the desired cell, the route to that cell can be determined by selecting the cells having the smallest cells in the opposite direction from the Objective cell to the start cell. Obstacles will create increased cost to adjacent cells because of their effect from potential fields associated with the obstacles.

#### b- Depth-First based PF

Another alternative approach to path planning is the Depth-First method, which is based on Potential Fields [112]. It works by dividing the entire path in the configuration space into a continuous sequence of connected portions along the way of potential function. All segments must, therefore, “fall” towards the goal, with the end of each segment becoming the beginning of the next. Thus, this procedure is repeated at

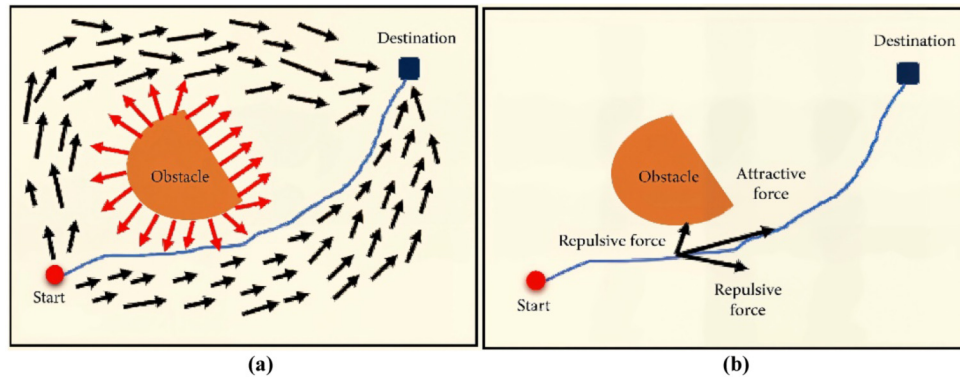


Fig. 14. Forces of artificial potential field (attractive &amp; repulsive).

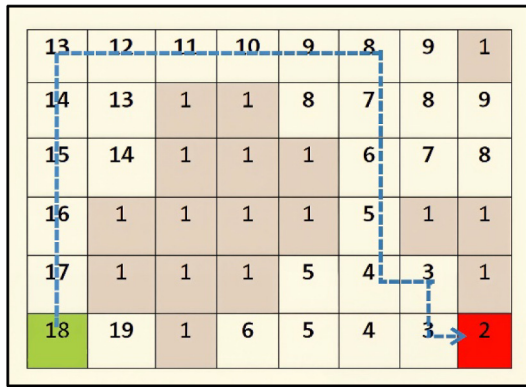


Fig. 15. The wavefront PF path construction.

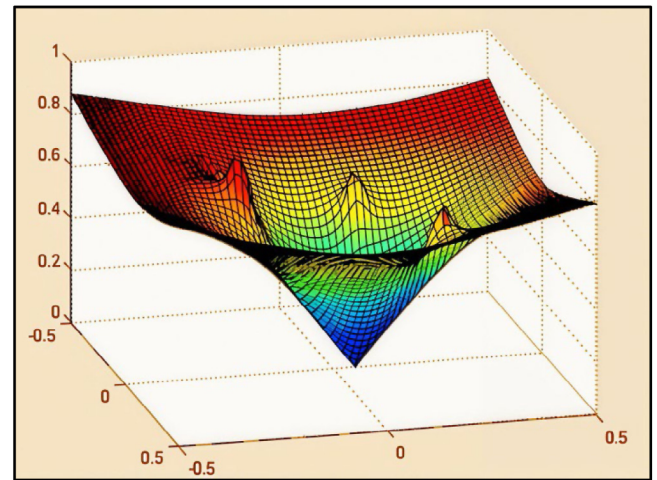


Fig. 16. Navigation function PF mathematical representation.

regular time intervals until the desired goal is reached. It is necessary during the path creation phase to make the segments sufficiently small in order to avoid obstacles and to not overshoot the goal [113]. On the other hand, the problem of local minima presents the most common barrier to this technique.

#### c- Breadth-First based PF

The Best-First algorithm with Potential Fields [112] is a possible solution for the path planning problem. The method starts with a starting point which becomes the root of the exploration tree. Then, it explodes into the configuration space to create a set of nodes that are connected to the main node. In the end, the next nodes with a similar configuration are checked. The neighbor with the lowest value is expanded and other neighbors around the expanded neighbor are kept in the node list.

This procedure is repeated until the desired goal is achieved. If local minima are found, then the algorithm will select a different node from the list, the potential value of which is the lowest. This will replace the node in the search process that currently exists. This approach resolves the local minima problem as opposed to the depth-first search method, which was afflicted by this problem [114]. Once the path is successfully traversed to the goal point and the search, the path can be obtained by traversing the path back from the goal to the starting point.

#### d- Navigation Functions based PF

Navigation Function is a set of functions for designing a trajectory of a robot. These functions are used to generate safe paths without colliding with the surrounding entities and walk towards the goal. As mentioned, there are local minima in potential fields. However, this problem can be eased if this potential field has one minimum only, representing the desired outcome, as shown in Fig. 16.

Other methods exist for the formulation of potential fields but the most effective one is the “function of global navigation” [115]. In

this case the goal path is chosen based on a prior knowledge of the environment. Obstacles are awarded greater magnitudes, whilst goals are allocated the lowest magnitudes. The robot must subsequently traverse the incline of the surface in order to reach its intended location. In addition, there are alternate approaches for accomplishing this goal, such as the methodologies suggested by Rimón [116] and Koditschek [117].

The numeric navigation function, sometimes referred to as an alternative way, is utilized to generate prospective fields with a decreased amount of intricacy. This is accomplished by depicting the configuration space as a grid spanning over vacant regions. The effectiveness of potential field strategies for general planning tasks, especially in outdoor scenarios with extensive and complex configuration spaces, is constrained [118]. Nevertheless, it can function as a technique for preventing collisions in a certain region, owing to its straightforward computational procedure. This enables the robot to traverse seamlessly between distinct territories, assuming that these areas have been pre-determined.

#### 2.2.2. Swarm optimization

##### Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a metaheuristic algorithm which is inspired by the collective behavior observed in social species like fish schools and bird flocks. PSO developed by Eberhart and Kennedy in 1995 [119], has gained significant popularity as an optimization technique in engineering and research domains. The PSO algorithm which is demonstrated in Fig. 17 emulates the actions of

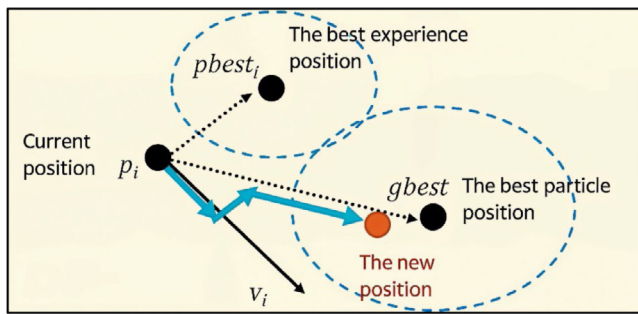


Fig. 17. Graphical demonstration of PSO algorithm.

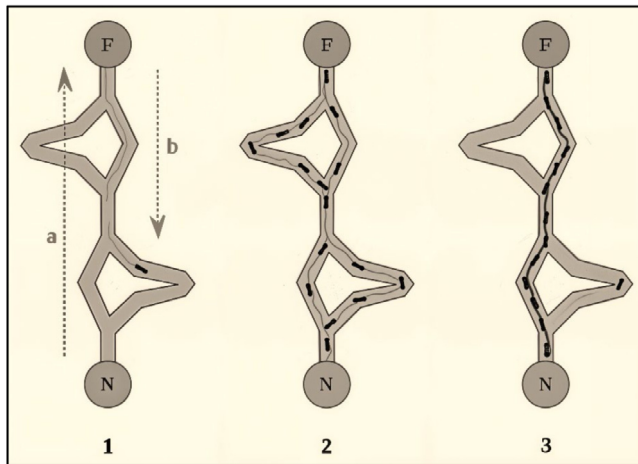


Fig. 18. Searching food behavior of ant colony.

social animals without necessitating the presence of a leader in the group to achieve the desired objective. While the birds' flock seeks food, they do not need leaders, they follow the member closest to the food.

PSO has been employed in the domain of mobile robot navigation to address issues pertaining to mapping and localization [120], accurate trajectory planning [121], and dynamic constraint problems [122]. The study of cooperative motion path planning for multi-robot systems has utilized PSO, which integrates the dynamics of multibody systems [123]. Furthermore, PSO has been used to facilitate the movement of several robots in actual settings [124]. A method known as self-adaptive learning PSO (SLPSO) has been proposed for path planning in complex environments with diverse constraints [125]. Hybrid techniques, such as combining PSO with the improved Gravitational Search Algorithm (IGSA), have been suggested to achieve optimal path planning for various mobile robots [126]. PSO has been successfully employed in the domain of path planning for several categories of robotics, such as underwater vehicles [127], aerial robots [128], humanoid robots [129], and industrial robots [130]. The non-deterministic navigation of unmanned aerial vehicles (UAVs) using PSO has been subject to examination in the defense industry [131].

#### Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is an algorithm of swarm intelligence devised by Marco Dorigo in 1992 [132]. It is employed to address problems related to combinatorial optimization. This strategy relies on the behavior of the population and is impacted by the navigational proficiency of ants, namely their ability to locate the shortest path from their nest to a food source (refer to Fig. 18). ACO has been successfully utilized in diverse fields such as job-shop scheduling, vehicle routing, traveling salesman problems, and graph coloring.

ACO is crucial in mobile robot navigation for the purposes of obstacle avoidance and optimizing path planning. Revanna et al. [133]

presented the real-time path planning of mobile robot using ACO. ACO outperformed other algorithms such as the Genetic Algorithms in terms of convergence speed, solution consistency, computational efficiency and dynamic convergence. Liu et al. [134] solved this problem by applying ACO to solve a problem of many mobile robots in the stationary environment. They used a collision avoidance algorithm and penalty function to make sure deadlocking does not occur at tight corners. Castillo et al. [135] presented a new method that combines ACO with Fuzzy Logic to navigate in a fixed environment. In their research project, Kumar et al. [136] used RA-ACO to guide a humanoid robot through complex and crowded environments.

In 2024, Sun and Ji [137] proposed improvements to ACO for static cases involving the combination of geometrical local optimization and pheromone diffusion to find the best routes. Yao et al. [138] proposed a new method to update pheromones to avoid redundant repetitions and to speed up the convergence in dynamic environments. In the early phase of a laser sensor, Song et al. [139] used ACO to select and optimize fuzzy rules to optimize alignment between the measurement velocity and accuracy, and then point laser displacement sensor probe scan path planning was investigated. In addition, a flowchart (see Fig. 19) was proposed by Brand et al. [140] to analyze the efficiency of a mobile robot in a similar environment by simulation and experimentation.

ACO is not only applied in flat or two-dimensional environments on earth but also in three-dimensional underwater environments for path planning [141]. Also, it has been applied to solve the routing problem in an unmanned aerial vehicle (UAV) field under battle condition. Zhao et al. [142] improved the ACO by adding the Q-learning approach to deal with the stagnation phenomenon and to improve search performance. There are a lot more applications of ACO than just mobile robot navigation. In this regard, Wu et al. [143] explored the application of ACO in the problem of designing trajectories for missiles. Their investigation revealed not only better performance with respect to the optimal length of the route but also to the convergence speed reached.

#### Artificial Bee Colony (ABC)

The ABC algorithm, developed by Kharaboga [144], is an intelligent algorithm based on swarm behavior that takes inspiration from the foraging habits of honey bees (refer to Fig. 20). The ABC algorithm is widely recognized for its simplicity, fast computation, and effectiveness in solving optimization issues. It functions by utilizing a food search cycle that has three rules.

Worker bees are sent out to evaluate the quality of nectar at feeding locations. Observer bees choose food sources based on information obtained from forager bees and evaluate the nectar's quality through calculations. Scout bees are responsible for examining potential food sources.

Contreras-Cruz et al. [145] modified ABC algorithm for the purpose of navigating mobile robots in a stationary environment. Their approach employs ABC for local search and evolutionary algorithms to determine the optimal route, which is then validated by real-time experiments done in an indoor setting.

Saffari et al. [146] utilized the ABC algorithm in a fixed context, yielding outcomes that were confined to simulated scenarios. In their study, Han et al. [147] proposed a hybrid method that integrates the ABC algorithm-based Ground threat prediction-based path planning to address the challenge of real-time navigation. The ABC approach for multiple mobile robots and multi-object navigation in static circumstances was successfully proposed [148,149].

The ABC method has expanded its use beyond wheeled mobile robots, with successful implementations in aerial navigation [150], underwater navigation [151], and autonomous vehicle routing challenges [152]. Aslan et al. [153] made changes to the ABC algorithm to improve the path planning of Unmanned Combat Aerial Vehicles (UCAVs) in a three-dimensional (3D) environment. They used a balance-evolution strategy (BES) to enhance the precision of convergence between local exploitation and global exploration capabilities.

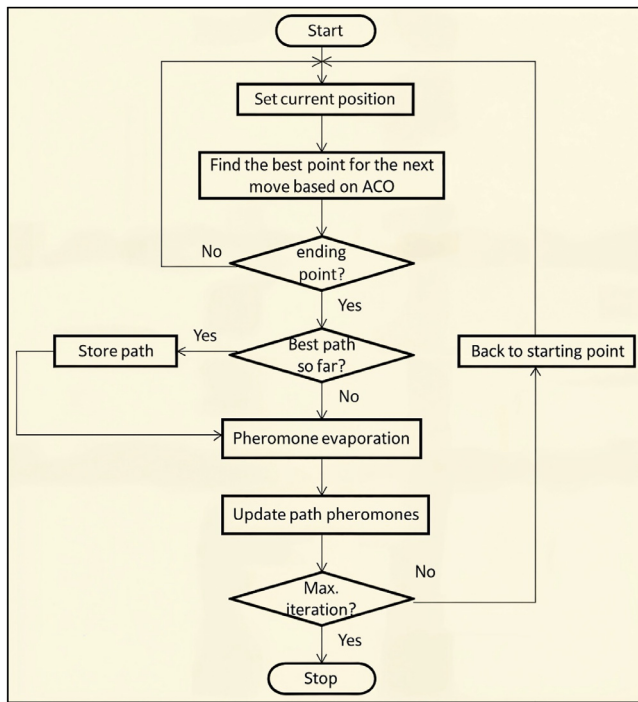


Fig. 19. ACO flowchart for robot path planning.

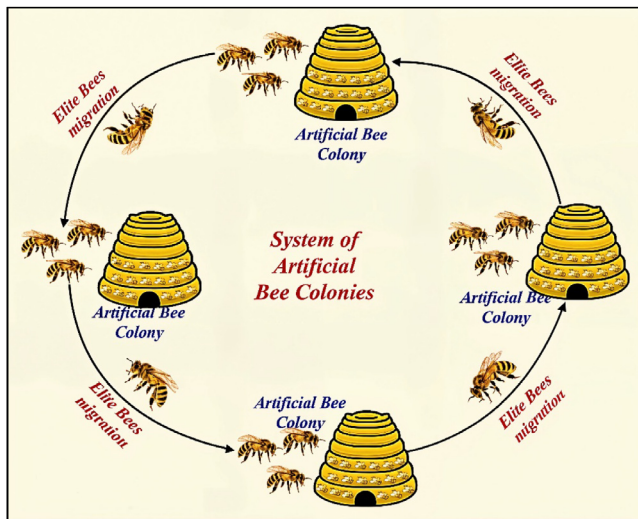


Fig. 20. Searching food behavior of bee colony.

Ding et al. [154] showcased the effectiveness of the ABC algorithm in the defense industry by implementing it in an unmanned helicopter for duties such as intelligence gathering, precise measurements, and border patrol. The researchers utilized a distinctive identification methodology, which involved the integration of a chaotic operator and an ABC controller, in order to identify unfamiliar parameters of linear models.

#### Grasshopper Optimization Algorithm (GOA)

Insects are a group of animals that are classified as grasshoppers, which have a negative effect on agriculture and agricultural yield. Fig. 21 is a diagram of grasshopper's life cycle. Though typically solitary in nature, grasshoppers can aggregate in small numbers in their natural habitats, and can even congregate and create one of the largest swarms of all insects [155]. These swarms may be as large as continent

size and can be a major challenge for farmers. One of the most striking features of grasshopper swarms is that the swarming activity of both nymphs and adults. It is a large-scale phenomenon for a large number of nymphs making it an important aspect of the grasshopper's life.

GOA in general as an optimization algorithm [156] and within autonomous mobile robots has been widely recognized and valuable in diverse areas of our lives, especially in complex environments [157]. One of the crucial functions of the robot is to strategically plan its path from a starting point to a specific destination, skillfully maneuvering past obstacles in its environment. Lately, there has been an increased emphasis on path planning, resulting in the creation of several techniques. Extensive study in this field has emphasized the efficacy of swarm intelligence, particularly in complex and both known and unknown contexts.

In [158] authors present a method for global path planning that utilizes the GOA in a known and unchanging environment. In order to improve the efficiency of the algorithm and optimize the resulting path, a bias factor is integrated. Moreover, the resulting path is further refined using an enhanced version of the multinomial logistic regression technique (MLR). The algorithms' performance is evaluated in three separate huge environments that have different levels of complexity. Fig. 22 explains the main procedures of developed GOA for path planning.

The GOA algorithm is being compared to the ACO in identical situations. The experimental findings unequivocally establish the superiority of our method in terms of both time convergence and cost-effectiveness. The study presented in 2021 [159] tackles the complex challenges of guiding mobile robots in dynamic and static situations. Existing techniques falter under chaotic conditions, become overwhelmed in complex environments, and have difficulty in identifying optimal pathways. They offer a novel approach, inspired by the grasshopper, that uses an extremely sensitive sensory system to detect obstacles, and a clever strategy to escape from both stationary and unpredictable obstacles. Our robot controller is able to handle randomness and a lot of people in a simulated environment. This navigation technique is faster, more efficient, more stable and more durable than any other.

#### The Bacterial Foraging Optimization (BFO)

Bacterial Foraging Optimization (BFO) method is a nature-inspired optimization algorithm introduced by Passino in 2002 [160]. Its shape is influenced by the foraging behavior of *E. Coli* and *M. Xanthus* bacteria are actively foraging for resources as shown in Fig. 23. Such bacteria are able to find sources of food over a given period of time, and to use energy effectively. The BFO algorithm considers key factors including chemotaxis, swarming, reproduction, elimination and dispersal. The points below present this behavior clearly:

- Bacteria consistently migrate in pursuit of places with higher nutrition availability on the map. Bacteria that have an ample supply of food have an extended lifespan and undergo binary fission, dividing into two equal pieces. Conversely, bacteria residing in regions with little nutrients will disperse and perish.
- Bacteria in nutrient-rich areas are drawn towards each other through chemical attraction, while bacteria in nutrient-poor areas emit a distinct signal as a warning to other bacteria.
- Bacteria occupy a region on the globe that is rich in nutrients.
- Bacteria are once again scattered over the map to access a fresh area with nutrients.

Coelho et al. [161] employed the BFO method to navigate mobile robots in stationary scenarios, incorporating adjustable velocity defined by uniform, Gaussian, and Cauchy distributions. This method was extended to accommodate diverse impediments [162–164]. Kasruddin et al. [165] demonstrated the capability for a fuzzy control design of a flexible manipulator.

Li et al. [166] introduced an improved BFO approach for an enhanced path planning for wheeled robots. This algorithm uses a repulsion force against obstacles and attraction to the goal in the form of

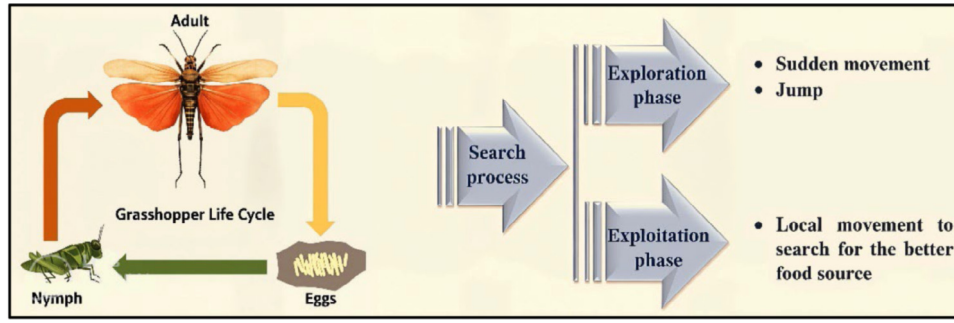


Fig. 21. GOA life cycle with the (exploration &amp; exploitation) search process.

```

Input : start_point, target_point, obstacles, max_iterations, population_size,
lower bound, upper_bound
Output: path
initialize the swarm positions  $X_i$  ( $i = 1, 2, \dots, \text{population\_size}$ );
initialize  $c_{max}$  and  $c_{min}$  ;
paths = I1 ;
calculate the fitness of each grasshopper ;
B = the best grasshopper ;
iter = 1 ;
while iter <= max iterations do
     $C = c_{min} + \text{iter} \cdot ((c_{max} - c_{min}) / \text{max\_iterations})$  ;
    for i = 1 to population size do
        normalize the distance between grasshoppers ;
        new_position = calculateNewPosition ( $X_i$ , C, B)
        if outsideBoundaries ( new position, lower bound, upper bound) then
            new position = recalculate( new position, lower bound, upper bound);
        end
        line = linesegment ( $X_i$ , new_position) ;
        if not intersectobstacle (line, cbstacles) then
             $X_i = \text{new\_position}$  ;
            path(i).append ( $X_i$ ) ;
            if fitness ( $X_i$ ) < fitness (B) then
                B =  $X_i$  ;
            end
            if reached ( $X_i$ ) then
                path = paths[i] ;
                return path;
            end
        end
    end
    iter = iter +1 ;
end
Return no path found;

```

Fig. 22. Main procedures of goa algorithm.

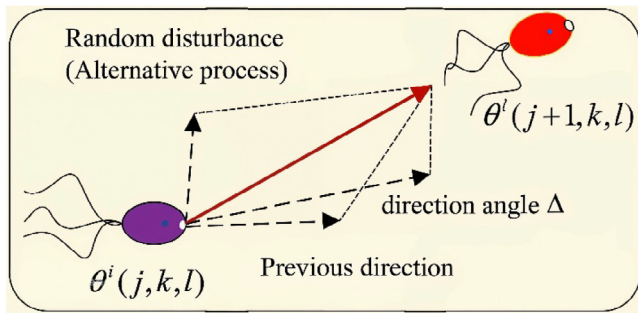


Fig. 23. BFO searching behavior.

the Artificial Potential Field (APF) method. In order to overcome the issues of local optimum and fixed step size search, Tao et al. [167] improved the Bacterial Foraging search by the adaptive step length in multi dimension.

The practical use of BFO algorithm can be applied to wheeled robotic systems. When applying the BFO algorithm to an industrial manipulator, Wang et al. [168] found that the augmented BFO algorithm

had a better performance compared to the classic BFO approach. Tao et al. [169] applied BFO to the navigation of Unmanned Surface Vessel (USV). The BFO algorithm has proven its effectiveness in different types of applications like static and dynamic environments, multi-robots and UAV navigation. Thus, a very useful tool for solving problems in mobile robot navigation.

#### The Firefly Algorithm (FA)

In 2008, Yang proposed the Firefly Algorithm (FA) [170] which is an inspiration from firefly's luminous behavior. Firefly is a flying beetle of the family Lampyridae, which is known to produce light. Light is produced by the fast oxidation of Luciferin without the presence of the enzyme Luciferase. The production of light is known as bioluminescence and FA uses it to produce light without losing thermal energy. Bioluminescence is used by fireflies for mating selection, communication and sometimes to deter predators. The FA algorithm flowchart is shown in Fig. 24.

The stochastic nature of FA that mimics the random behavior and trial-and-error movements of fireflies in nature is better suited to problems in mobile robot navigation. A finite automaton has been used to enhance the effectiveness of path length, smoothness and safety for the individual robot navigation [171] and to suggest some

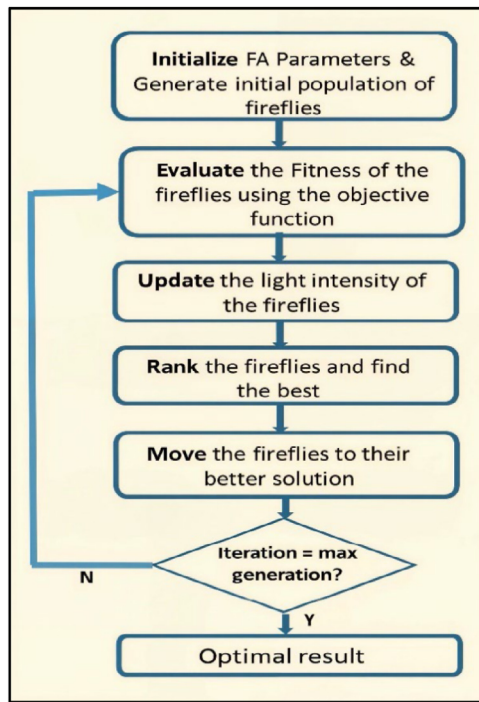


Fig. 24. Flowchart of FA algorithm.

novel approaches to multi-target navigation [172]. Moreover, FA has also been applied to co-operative approaches in automatic assembly systems, with its effectiveness successfully demonstrated for industrial robots [173].

The concentric sphere-based approach is one of the changes proposed, which aims to make the FA more effective [174]. It has been applied in various applications such as vision-based systems [175], learning methodologies for FA (FA-Q) [176] and hybrid techniques [177]. It has been effectively applied in robots with legs [178] and robots designed for UAV use [179]. The FA has risen to the occasion, skilfully navigating in evolving environments by efficiently tracking the motion of objects. [180]

#### The Shuffled Frog Leaping Algorithm (SFLA)

A Shuffled Frog Leaping Algorithm (SFLA) metaheuristic optimization technique has been developed by Eusuff and Lansey [181]. It is derived from the “foraging” activity of the frog, when it hunts for food (Fig. 25). This algorithm is famous for its better convergence speed, easy implementation, less number of parameters, high success rate, and better uncertainty search performance.

SFLA algorithm is widely used in the navigation of mobile robot. Ni et al. [182] proposed a path planning algorithm based on the median strategy to tackle the problem of local optimal solutions. The fitness function was adjusted to obtain the most efficient path, and the position of the best frog in the world was used to guide the robot’s movement in the case of stationary and moving obstacles.

In navigation, Hidalgo-Paniagua et al. [183] proposed a multi-objective method based on the SFLA technique, which aimed to maximize path safety, minimize path length and smoothness. They showed better smoothness of the path in static situation than GA. In their study, Kundu et al. [184] investigated an adaptive navigational approach using SFLA (Swarm-based Firefly Algorithm) in 3D underwater environments. The objective was to optimize the path and navigation time by effectively avoiding local minima circumstances. This was achieved by the implementation of an adaptive technique, as depicted in Fig. 26.

The simulation results were validated by experimental results, demonstrating a high level of concordance in a complex scenario. SFLA

variants have also been utilized for UAV 3D path planning [185] and Land Use/Land Cover Prediction [186]. Gao et al. [187] devised a robot control system using an improved SFLA. This strategy encompasses control in space searching ability which improves the performance of the control system. The ISFLA algorithm finds the optimal path for the robot reaching to the goal that reduces the robot consuming energy for the task achievement and enhances the efficiency of the system.

#### Grey-Wolf Optimization (GWO)

First Grey Wolf Optimization (GWO) algorithm proposed in 2014 was by Mirjalili et al. [188], which imitates the hunting skills and social organization of grey wolves. The system classifies grey wolves into four hierarchical levels as illustrated in Fig. 27 (alpha, beta, delta, and omega). In a wolf pack, the alpha wolf assumes the role of the leader, while the omega wolves have the lowest position in the social hierarchy. The omega wolves fulfill many responsibilities including scouting, guarding, providing guidance, hunting, and caring for others. In addition to its role in social leadership, the GWO algorithm also contains the fascinating hunting method of grey wolves.

The strategy of the grey wolf hunting, combined with its social leadership mechanism, introduces a fascinating aspect to the GWO algorithm. Wolf hunting is a collective activity that highlights their intriguing social dynamics. Wolves cooperatively track and surround prey under the leadership of the alpha wolf. The mathematical model of the grey wolf hunting strategy posits that alpha, beta, and delta wolves yield more insights into probable forage locations. Hence, the GWO algorithm employs the alpha, beta, and delta solutions to relocate the wolves, thereby improving its overall effectiveness. Fig. 28 shows the flowchart of GWO:

This method was suggested to tackle the difficulties in robot route planning [189], particularly in finding the most optimal path from the starting point to the target one while avoiding obstacles (refer to Fig. 29).

The performance of the GWO algorithm was assessed in a zone including three circular obstacles of different sizes. Several methods for planning the robot path have been proposed in [190], and many improved GWO varies in the same domain can be found [191–193].

#### Red-Wolf Optimization (RWO)

Foxes are widely recognized for their impressive capacity to adapt to various environments and their superior intelligence. There are two personas of red foxes that can be distinguished. Individuals with a fixed living and those who frequently move. Each group is led by a specific number of leaders (alphas), who guide the others [194]. The RFO algorithm [195] was employed to simulate interactions among predators, prey, hunters, foxes, and the rest of the fox family. Each part of the model indicates a unique aspect of fox development and predation that has played a role in its enduring success within various environments. This algorithm has lately been utilized to deal with a range of optimization issues [196–199]. Such bio-inspired metaheuristics align with the broader trend in robotics research, where optimization models are increasingly applied to enhance system efficiency and adaptability [200].

In this context, the RFO method was employed in this study [201] to tackle the path planning problem. The RFO is considered a novel metaheuristic optimization algorithm that borrows inspiration from the hunting behavior of red foxes [195]. Upon observing prey in a faraway location, the fox starts a global search for food over its whole region. The fox moves through its surroundings to locate the prey as closely as it can before initiating an attack, which is considered a local hunt. Fig. 30 illustrates the conceptual model of views towards fox hunting.

The RFO algorithm does the global explore by simulating the exploration of the solution space, similar to how a fox hunts for food at a vast distance. On the other hand, local search emulates the traversal of regions intending to get as close as possible to the prey before attacking it [198] (see Fig. 31).

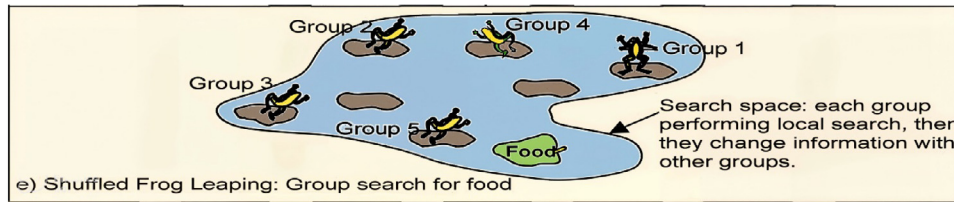


Fig. 25. SFLA searching food behavior.

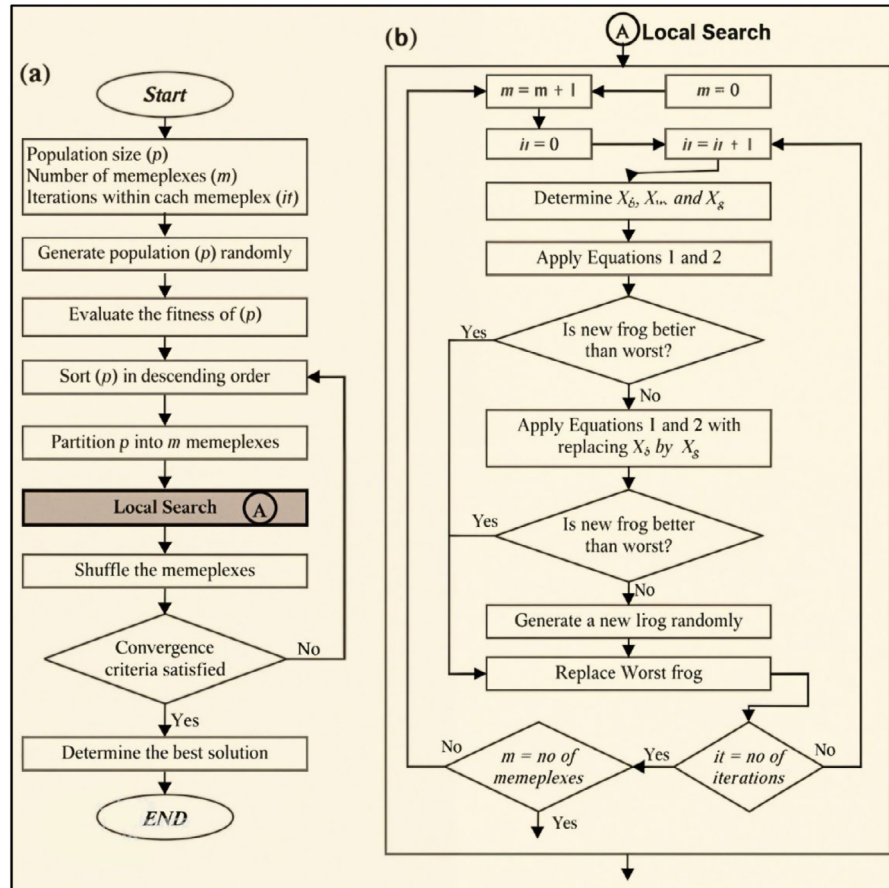


Fig. 26. Flowchart of SFLA algorithm.

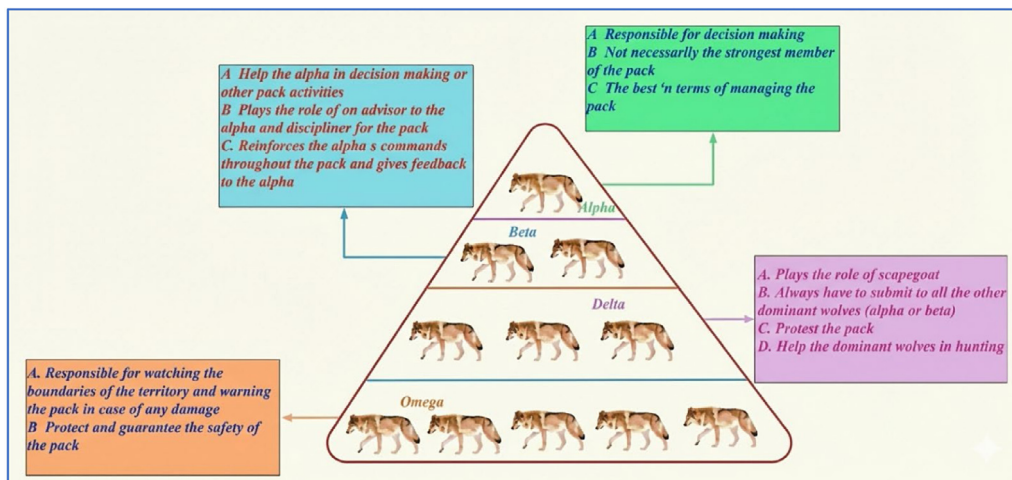


Fig. 27. Hierarchical levels of grey wolves.

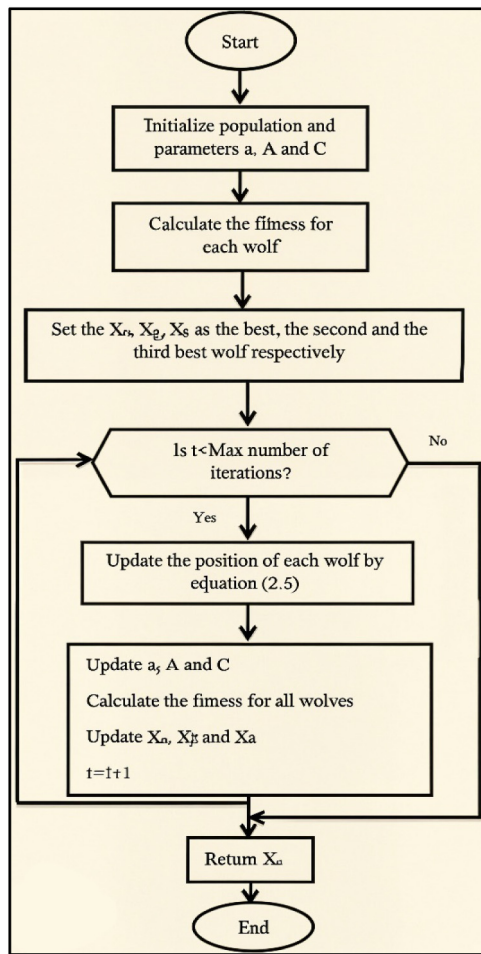


Fig. 28. Flowchart of GWO algorithm.

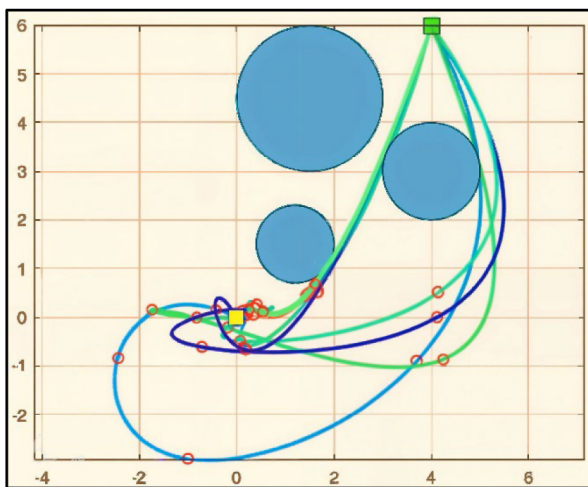


Fig. 29. Construction of GWO Path Planning.

The last few years have seen a significant change in the field of mobile robot navigation and multi-robot coordination, with new algorithmic advances in a variety of frameworks. For example, educational methods for trajectory prediction for dynamic obstacle avoidance have been considered, and curve fitting methods have been demonstrated to enhance robot navigation performance [202]. Likewise, most recently, the technical challenges of human-robot interaction (O-HRI) in the

context of mobile robot navigation among humans were studied, and the limitations that arise in the context of safe motion planning were discussed [203].

In addition to classical designs, their work also includes an optimized deep reinforcement learning version of the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm that performs well on complex navigation tasks [204] and a memory-based trap recognition technique, VFH+T [205] that eliminates persistent local minima. In the case of collective motion, Cao et al. extended the Dynamic Window Approach to allow robot formations to dynamically avoid obstacles by real-time changes in speed and heading [206]. In addition to these strategies, Low et al. implemented extensions of Q-learning, such as Dynamic Multi-Model Tabular Q-Learning (DMMTQL) and Deep Q-Learning (DOQL) which are both very flexible for static and dynamic environments [207].

### Reinforcement Learning (RL) and Deep RL

Reinforcement Learning (RL) is a paradigm whereby an agent learns to make decisions by interacting with the environment whereby it is provided with feedback in terms of rewards or penalties. In the context of robotic navigation, the agent (in this case, the robot) comes up with a policy that maps the current state (say sensor measurements or its position) to an action (say go forward or turn right) in such a way that it maximizes the cumulative reward, say that it arrives at a target avoiding obstacles [208].

Deep Reinforcement Learning (Deep RL) is a combination of RL with deep neural networks, thus allowing the agent to learn complex policies directly, as an input of high-dimensional sensory data, including camera images or LiDAR point clouds. This synergy has helped create great progress in autonomous navigation which enables robots to perform in the complex and dynamic environments with no need to follow a set of behavioral schemes through explicit programming of the robots [209]. The main strength of Deep RL is its ability to support the end to end policies of navigation in which the raw sensor data is converted directly to control commands and thus no hand-made features or intermediate representations are necessary. Experimental results show that it is effective in a wide variety of applications, such as indoor navigation, outdoor terrain traversal, dynamic obstacle avoidance [210].

Deep Q -networks (DQN) are trained on a value function, which can be described as Q -function, that approximates the expected reward of each action in a particular state. With the use of a deep neural network to estimate the Q, DQN can be scaled to large state spaces. Introduction of experience replay and target networks, were also used to stabilize the training and this was a major innovation that enabled the use of deep learning in the RL structures [211]. Recent comparative studies have shown that DQN can do well in discrete action space tasks, but have challenges when dealing with continuous control tasks that are commonly found in robotics. DQN has demonstrated success rates of 85–92 percent in simulated milieus in the context of unmanned aerial vehicle (UAV) obstacle avoidance, but with an effective convergence, reward shaping had to be carefully considered [212].

Proximal Policy Optimization (PPO), Advantage Actor-Critic (A3C) and other RL methods directly learn a policy, a mapping or state-to-action. PPO is a state-of-the-art policy gradient algorithm that is also stable and sample-efficient and thus, makes it especially applicable to real-world robotic tasks [213]. The algorithm uses a clipped surrogate objective function, which reduces the large policy updates and, as a result, there is increased training stability. A3C uses two or more agents concurrently to search the environment more effectively, and thus significantly shortening training times. The modern studies have shown that the PPO can perform better in continuous control tasks succeeding over 95% in complicated navigation conditions [214]. Integration of a two-attention system plus the use of PPO has been demonstrated to increase autonomous navigation in narrow passages that lead to a 23% gain in the efficiency of navigation compared to the usual PPO model [215].

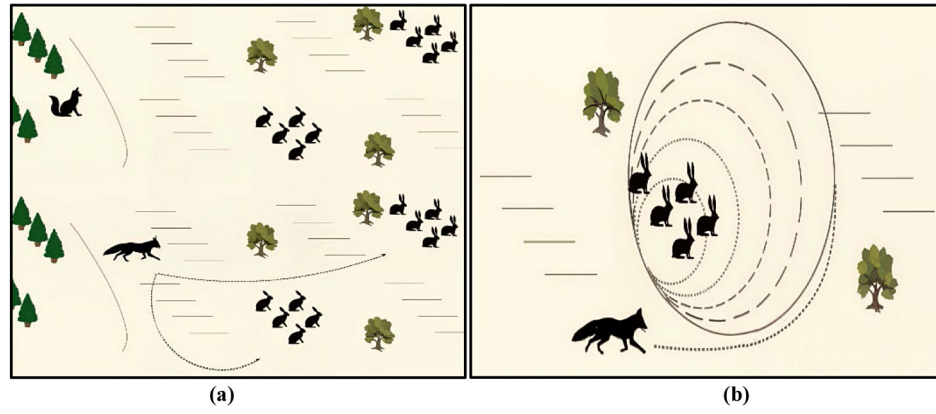


Fig. 30. Red-Fox Hunting behavior (a) Globally, (b) Locally.

```

Input: start_position, target_position, obstacles
Output: path
Setnp:
Determine algorithm's parameters: fitness function  $f(\cdot)$ , dimensions of search area
solution (a, b), T is the number of iterations, n is the maximum population size.
1. Generate population consisting of  $n$  foxes around start position
2.  $t := 0$ 
3. Define scaling parameter  $\alpha$ 
4. while  $t \leq T$  do
5.   Sort population based on the fitness function (3)
6.   Select  $(x_{best})^t$ 
7.   foreach fox, in the current population do
8.     Calculate the new position of individuals according to (2)
9.     if the new location is valid then
10.      if the new position is preferable to the previous position
11.        then
12.          if their local minima then
13.            Change direction
14.            Move the fox;
15.          else
16.            Move the fox;
17.          endif
18.        else
19.          Return the fox; to its previous location
20.        endif
21.      else
22.        Return the fox; to its previous location
23.      endif
24.      if reached the target then
25.        return the path
26.      endif
27.    endfor
28.    Update fitness
29.     $t++$ 
30.  endwhile
31. Stop

```

Fig. 31. Main Procedure of RFO Algorithm.

Actor-Critic Methods such as Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3) or Soft Actor-Critic (SAC) combine value-based methods with policy-based methods. Both methods consist of two networks: an actor which learns the policy and a critic which approximates the value function. DDPG is a popular actor-critic algorithm in continuous action space, which can allow the robotic systems to be controlled smoothly and without errors [216]. TD3 optimizes

DDPG by addressing the bias of overestimation by implementing twin Q-networks and policy update delay. The map-less navigation tasks presented in the study by TD3 have shown excellent performance, achieving path-planning efficiency improvement of 15–30 percent over DDPG [217]. SAC is an exploration method that encourages regularization of entropy to encourage stronger policies. Recent studies show that TD3-based methods have a success rate of between 94 and 98 percent

in dynamic settings with moving obstacles, thus performing better than that of both DQN and the conventional DDPG [218]. TD3 has been applied to the control of coverage planning has revealed that robots are able to intelligently and efficiently manoeuvre through complex environments without losing the overall area coverage [219].

### Graph Neural Networks (GNNs) for Spatial Reasoning

Graph Neural Networks (GNNs) are a type of neural network architectures that have been designed with the express purpose of being used on graph-structured data. When applied to robot navigation, GNNs facilitate the modeling of the environment in the form of a graph where spatial points or objects are represented by the nodes and their interrelationships by the edges. Such an image enables advanced spatial calculations, enabling a robot to discover paths that are navigable, understand the relationship between environmental features and make better navigational choices [220]. In contrast to a traditional convolutional neural network that is designed to handle grid-structured data, GNNs are designed to support irregular and non-Euclidean geometries of a real-world environment, which makes them especially important in robotic tasks.

One of the primary benefits of the GNNs is that it can support decentralized multi-robot path planning. GNNs learn communication and coordination policies enabling efficient and collision-free navigation by building a graph, where each robot is a node and inter-robot communication links are the edges [221]. The initial research by Li et al. (2020) proved that the hybrid CNN-GNN network can be used to automatically generate local communication and decision-making policies concerning robot work in a limited workspace. They were able to perform as well as centralized expert algorithms would with just local communication and observation and demonstrated astonishing generalization to large environments and robot teams [222].

Recent studies have sought to combine GNNs with reinforcement learning in order to complement navigation further. Multi-agent Collaborative Navigation System (MACNS) is a graph of GNNs and deep reinforcement learning to enable dynamic trajectory planning in multi-agent systems [223]. The approach has been already used in the logistics of deliveries, automation of warehouses, and individual tourism solutions, which reinforces the adaptability of GNN-based navigation systems. At the same time, GNNs that are geometrically encoded are developed, which makes them more effective at encoding the spatial relationships between the robots, providing improved coordination of operations in group navigation tasks [224].

The paradigm of graph-relational reinforcement learning has become an effective framework in mobile robot navigation in large and crowded environments. Representing roads as a graph in a two-dimensional form, processing the spatial interactions with the help of GNNs, the robots are able to learn how to move quickly and avoid the dynamic objects like pedestrians [225]. This has led to significant gains in the success rates in navigation and the efficiency of the paths compared to the conventional means. Besides, object-level map navigation methods based on graph matching have been produced, which combines point-cloud registration with graph matching to provide precise pose estimation [226].

### Transformer-based Approaches for Planning

Transformers were first developed in the natural language processing field; they have recently been applied in robotics and planning. Their attention system provides the models to have the ability of capturing long distance dependencies and complex relationships in sequential data. Transformers can also be used in the framework of path planning, in which the input is the environment and the output is a series of actions or waypoints defining the desired path [227]. The self-attention mechanism allows the model to place different levels of emphasis in divergent components of the inputs when generating the output, and this is of great benefit when negotiating a complicated environment where there are many obstacles.

ViTs have been effectively implemented with robot path planning. ViT-A method is a combination of the Vision Transformer and A\* algorithm that is used to navigate a legged robot [228]. Under this paradigm, camera images are run through ViT to generate a map of costs of the environment and then used by the A\* algorithm to find optimal path. Empirical assessments on actual robotic quadrupeds, i.e., Boston Dynamics Spot and Unitree A1, proved that ViT-A has better results in overcoming obstacles on the way than the traditional techniques. The variability of the terrain type enabled by the data-driven character of this approach allows adapting to different terrain types without the need of adjusting the parameters manually [229].

Imitative reinforcement learning with transformers has been found as an effective tool in the path planning of multi-robots. Transformer architectures were introduced in policy neural network robotic coordination by Chen et al. (2023), which trains robots to exhibit the inter-agent coordination [230]. It is a synergistic methodology that unifies imitation learning and reinforcement learning to initialize the policy with expert demonstrations and then improve the performance of the policy through reinforcement learning. The transformer architecture aids in the encoding of the inter-robot dependencies, which are complex hence resulting in more efficient collision avoidance and path planning.

Transformers-based mission-conditioned path planning deals with the issue of integrating mission requirements into the planning flow [231]. This plan takes advantage of a deep-learning architecture that is able to produce paths meeting specific mission constraints, e.g. traveling to a specified set of waypoints in a given order or avoidance of certain areas. The attention mechanism facilitates the model to focus on mission constraints relevant in the synthesis of the trajectory. Other more recent studies have explored the combination of Transformers and Graph Neural Networks to manage the smart logistics. A path-planning algorithm suggested by Luo et al. (2025) utilizes both Transformer and Graph Convolutional Network (GCN) to process multimodal data and generate the most suitable paths to the logistics robots [231]. Deep reinforcement learning algorithms have also been provided with attention mechanisms. Agile DQN technique uses adaptive deep recurrent attention so as to improve the performance of the navigation in dynamic environments [232]. Simultaneously, hierarchical reinforcement learning systems with attention modules have been designed in the context of UAV autonomous navigation, which provides better results in the complex three-dimensional environment [233].

## 3. Critical analysis and synthesis

This section gives a detailed comparative analysis and synthesis of the reviewed path planning methodologies. Table 1 analysis considers and compares algorithms based on agreed-upon metrics rather than mechanistic descriptions as in Section 2, looks at the performance of hybrid fusion strategies, and explores key research gaps and areas of unexploited potential. The analysis is organized along four main topics: comparative efficiency, assessment of the hybrid method (fusions, convergence, sensitivity of parameters), performance benchmarking, and new research directions.

### 3.1. Comparative analysis of efficiency and performance

Path-planning algorithms are indispensable for robot applications in the real world, and their performance and efficiency are key. Dijkstra and A\* are classical algorithms that can be proved to be optimal for the static environment and have polynomial time complexity,  $O(E \log V)$  for Dijkstra and  $O(bd)$  in worst case for A\* algorithm, but they do not possess any good adaptability properties, and high re-computation cost in dynamic environment. Intelligent approaches (PSO, GA, ACO) are more flexible because the population-based search, but are more expensive in terms of computation and come without optimality guarantees. The modern ML-based methods (DRL, GNNs, Transformers)

**Table 1**  
Comparative framework of path planning algorithms.

| Algorithm                                 | Completeness | Space complexity    | Computational cost | Reference        |
|-------------------------------------------|--------------|---------------------|--------------------|------------------|
| Dijkstra's Algorithm                      | Complete     | $O(V)$              | Medium             | [4–9]            |
| A* Algorithm                              | Complete     | $O(V)$              | Medium             | [10–25]          |
| Weighted A*                               | Complete     | $O(V)$              | Low-Medium         | [17]             |
| Hierarchical A*                           | Complete     | $O(V)$              | Medium             | [20]             |
| Lifelong Planning A* (LPA*)               | Complete     | $O(V)$              | Medium             | [26]             |
| D* Algorithm                              | Complete     | $O(V)$              | Medium-High        | [30–33]          |
| D* Lite                                   | Complete     | $O(V)$              | Medium             | [34–36]          |
| Field D*                                  | Complete     | $O(V)$              | Medium-High        | [37]             |
| Probabilistic Roadmap (PRM)               | Complete     | $O(n^2)$            | High               | [39,40]          |
| RRT (Rapidly-exploring Random Tree)       | Complete     | $O(n)$              | Medium             | [41–47]          |
| RRT-Connect                               | Complete     | $O(n)$              | Medium             | [45]             |
| RRT*                                      | Complete     | $O(n)$              | Medium-High        | [48–50]          |
| RRT*-Smart                                | Complete     | $O(n)$              | Medium             | [49]             |
| SRT (Sensor-based Random Tree)            | Complete     | $O(n)$              | Medium             | [46,47]          |
| Voronoi Diagram                           | Complete     | $O(n)$              | Medium             | [52]             |
| Visibility Graph                          | Complete     | $O(n^2)$            | High               | [54]             |
| Cell Decomposition (Exact)                | Complete     | $O(n)$              | Medium-High        | [58,59]          |
| Cell Decomposition (Approximate)          | Complete     | $O(n)$              | Low-Medium         | [58,59]          |
| Quadtree/Octree                           | Complete     | $O(n)$              | Medium             | [56,57]          |
| Artificial Potential Field (APF)          | Incomplete   | $O(1)$              | Very Low           | [106,107], [118] |
| Navigation Function                       | Complete     | $O(n)$              | Low                | [116,117]        |
| Wave-front Algorithm                      | Complete     | $O(n^2)$            | Medium             | [110,111]        |
| Breadth-First Search (BFS)                | Complete     | $O(V)$              | Low                | [113,114]        |
| Depth-First Search (DFS)                  | Complete     | $O(V)$              | Very Low           | [113,114]        |
| Genetic Algorithm (GA)                    | Incomplete   | $O(p \times n)$     | High               | [73–91]          |
| Adaptive GA                               | Incomplete   | $O(p \times n)$     | High               | [83–86]          |
| Hybrid GA-PSO                             | Incomplete   | $O(p \times n)$     | High               | [80]             |
| Particle Swarm Optimization (PSO)         | Incomplete   | $O(p \times d)$     | Medium             | [119–131]        |
| Multi-Objective PSO                       | Incomplete   | $O(p \times d)$     | Medium-High        | [122]            |
| Self-Adaptive PSO                         | Incomplete   | $O(p \times d)$     | Medium             | [125]            |
| PSO-DWA Hybrid                            | Incomplete   | $O(p \times d)$     | Medium             | [122]            |
| Ant Colony Optimization (ACO)             | Incomplete   | $O(n^2)$            | Medium-High        | [132–143]        |
| Quantum ACO                               | Incomplete   | $O(n^2)$            | High               | [138]            |
| ACO with Q-Learning                       | Incomplete   | $O(n^2)$            | High               | [142]            |
| Artificial Bee Colony (ABC)               | Incomplete   | $O(n \times d)$     | Medium             | [144–154]        |
| Multi-Objective ABC                       | Incomplete   | $O(n \times d)$     | Medium-High        | [149]            |
| Chaotic ABC                               | Incomplete   | $O(n \times d)$     | Medium             | [154]            |
| Grasshopper Optimization Algorithm (GOA)  | Incomplete   | $O(n \times d)$     | Medium             | [156–159]        |
| Bacterial Foraging Optimization (BFO)     | Incomplete   | $O(n \times d)$     | Medium-High        | [160–169]        |
| Hybrid Spiral-BFO                         | Incomplete   | $O(n \times d)$     | Medium-High        | [165]            |
| Firefly Algorithm (FA)                    | Incomplete   | $O(n)$              | Medium             | [170–180]        |
| Hybrid Whale-FA                           | Incomplete   | $O(n)$              | Medium-High        | [177]            |
| Shuffled Frog Leaping Algorithm (SFLA)    | Incomplete   | $O(m \times n)$     | Medium             | [181–187]        |
| Multi-Objective SFLA                      | Incomplete   | $O(m \times n)$     | Medium-High        | [183]            |
| Grey Wolf Optimizer (GWO)                 | Incomplete   | $O(n \times d)$     | Medium             | [188–193]        |
| Multi-Strategy GWO                        | Incomplete   | $O(n \times d)$     | Medium             | [191]            |
| PSO-GWO Hybrid                            | Incomplete   | $O(n \times d)$     | Medium-High        | [193]            |
| Red Fox Optimization (RFO)                | Incomplete   | $O(n \times d)$     | Medium             | [195–201]        |
| Fuzzy Logic Controller                    | Incomplete   | $O(r)$              | Very Low           | [92–105,234–237] |
| Type-2 Fuzzy Controller                   | Incomplete   | $O(r)$              | Low                | [234]            |
| Neuro-Fuzzy Controller                    | Incomplete   | $O(r \times n)$     | Low-Medium         | [64,65], [102]   |
| Fuzzy-GA Hybrid                           | Incomplete   | $O(p \times r)$     | Medium             | [103]            |
| Fuzzy-ACO Hybrid                          | Incomplete   | $O(m \times r)$     | Medium             | [105], [135]     |
| Artificial Neural Network (ANN)           | Incomplete   | $O(n \times h)$     | Low -inference     | [60–71]          |
| Deep Neural Network (DNN)                 | Incomplete   | $O(h^2 \times l)$   | Low -inference     | [60], [69–71]    |
| Convolutional Neural Network (CNN)        | Incomplete   | $O(k^2 \times c)$   | Low (inference)    | [70]             |
| GBNN (Grid-Based Neural Network)          | Incomplete   | $O(n^2)$            | Medium             | [67]             |
| Q-Learning                                | Incomplete   | $O(s \times a)$     | High               | [207]            |
| Deep Q-Network (DQN)                      | Incomplete   | $O(n)$              | Very High          | [15], [101]      |
| Deep Deterministic Policy Gradient (DDPG) | Incomplete   | $O(n)$              | Very High          | [90]             |
| Twin Delayed DDPG (TD3)                   | Incomplete   | $O(n)$              | Very High          | [204]            |
| Proximal Policy Optimization (PPO)        | Incomplete   | $O(n)$              | High               | [101]            |
| Multi-Objective RL                        | Incomplete   | $O(n \times o)$     | Very High          | [123]            |
| A*-PSO Hybrid                             | Complete     | $O(V + p \times d)$ | Medium-High        | [8,9]            |
| A*-DWA Hybrid                             | Complete     | $O(V)$              | Medium             | [76]             |
| RRT*-Neural Network                       | Complete     | $O(n + h)$          | Medium             | [48]             |
| APF-Neural Network                        | Incomplete   | $O(h)$              | Low                | [63]             |
| GA-Deep RL                                | Incomplete   | $O(p \times n)$     | Very High          | [90]             |
| PSO-RL Hybrid                             | Incomplete   | $O(p \times d)$     | High               | [131]            |
| Dynamic Window Approach (DWA)             | Incomplete   | $O(1)$              | Very Low           | [206]            |
| Improved DWA                              | Incomplete   | $O(1)$              | Very Low           | [206]            |
| Vector Field Histogram (VFH)              | Incomplete   | $O(n)$              | Very Low           | [205]            |
| VFH+ (Trap Avoiding)                      | Incomplete   | $O(n)$              | Low                | [205]            |

offer a completely new trade-off: high offline training cost, but very fast inference speed, thus allowing real-time decision-making in complex dynamic environments. This distinction is quantitatively shown in Table 3, where classical methods perform best on a static setting, and score poorly on the dynamic suitability metrics, while DRL methods score well on all of the dynamic metrics, although they have a high training burden.

### 3.1.1. Analytical evaluation of hybrid methods

The combination of different algorithms is an important research path, which attempts to combine the best features of each individual algorithm. The success of these methods relies heavily on the type of fusion mechanism used, the convergence developed in each cycle, and the sensitivity to the parameter configuration.

#### Fusion Mechanisms

Based on the literature reviewed, three principle fusion architectures have been identified. First, a global planner (such as A\*, GA or PSO) is used to calculate an approximate optimal path based on the available static prior knowledge, and then a local planner (such as APF, DWA) is used to implement real-time obstacle avoidance. In the A\*-APF hybrids, A\* overcomes the well-known local minima problem inherent in the APF by introducing global waypoints which direct the potential field gradient, and APF provides the smooth continuous control command that is lacking from grid-based A\*. The task of planning is broken down to stages and each stage is processed sequentially in the pipeline, with the outcome of one algorithm being passed to the next. In the case of RRT\*-neural network hybrids, for instance, RRT\* itself produces a feasible initial path and then improves the path using a trained neural network for smoothness and energy-efficiency. Multiple algorithms are executed in parallel on a cooperative architecture, with the results of each being chosen and combined according to performance measures. In the PSO-GA hybrid approach for example, the PSO's velocity-based exploration and the GA's crossover-mutation operators work concurrently, where the best solution is selected at each iteration.

#### Convergence Behavior

Convergence properties of the hybrid methods are significantly different from those of the basic algorithms. However, it is known that the standard PSO exhibits premature convergence in complex multimodal search spaces as the diversity of the population is lost quickly as the particles converge towards the global best. The incorporation of GA's mutation operators brings in the element of stochastic diversity that mathematically changes the property of the Markov chain of the search process, resulting in theoretical assurances of global asymptotic convergence when certain conditions are met. For ACO-based hybrids, the speed of convergence is directly affected by the pheromone evaporation rate ( $\rho$ ): a high evaporation rate ( $\rho > 0.5$ ) leads to rapid exploration but slow convergence, a low evaporation rate ( $\rho < 0.1$ ) leads to faster convergence but risk of premature convergence on suboptimal trails. Hybrid ACO-Q-Learning methods overcome this by dynamically changing the update rules of the pheromone using Q-values, without compromising on the quality of the solution found or the convergence speed. The convergence is controlled by the reward function design and the stability of estimates of the policy gradient for DRL-based hybrids. Actor-critic architectures (e.g., TD3, SAC) can also be used to help converge the network to a stable solution by using twin Q-networks to mitigate the overestimation issue, and the number of training steps needed for empirical convergence requires 500K-2M, depending on the environment complexity.

#### Parameter Sensitivity

Hyperparameter tuning is a practical challenge with intelligent and hybrid methods. In PSO, the inertia weight ( $w$ ) can be used to adjust the balance between exploration and exploitation: a value close to 0.9 gives a higher exploration tendency and slows down the convergence;

a value below 0.4 gives a higher exploitation tendency and speeds up the convergence, but also leads to higher risk of entrapment in local optima. The cognitive ( $c1$ ) and social ( $c2$ ) acceleration coefficients further modulate the individual and swarm learning rates, and the default values are  $c1 = c2 = 2.0$ , which offer a good balance. In ACO, the search behavior is a combination of the pheromone importance factor ( $\alpha$ ) and the heuristic importance factor ( $\beta$ ). High  $\alpha$  values have the tendency to converge fast, but not necessarily optimally, because ants start to follow the other ants' trail more and more, and high  $\beta$  values increase the computational cost, because ants start to rely more on distance heuristics than on learned information. The rate of crossovers (usually between 0.6 or 0.9) and mutation (usually between 0.01 or 0.1) should be well balanced: high mutation rates kill good solutions; low mutation rates cause genetic drift and premature convergence. Recent methods make use of adaptive parameter tuning techniques, including the linearly reduced inertia weight of the PSO particles or the self-adaptive mutation rate of GAs, to reduce sensitivity and enhance robustness to various problem instances.

### 3.2. Performance benchmarking

The application of universal benchmarking of all path-planning algorithms is difficult to do directly because the range of application situations and performance measures is quite broad. In this respect, we introduce a general comparison (Table 2), that summarizes findings presented in recent studies; the table provides information on the application domain and an environment type of most important algorithms in specific contexts.

### 3.3. Trend analysis

The field of robot path planning is continuously evolving. To visualize the research trends, we have created a new chart (Fig. 32) that illustrates the number of publications per year for different categories of algorithms. This chart highlights the growing interest in intelligent and machine learning-based approaches. Fig. 33 illustrates the distribution of research publications by method category

### 3.4. Analysis of method limitations and hybrid approaches

The most important part of this review is the analysis of the limitations of each approach in details. Although classical algorithms present theoretical guarantees, they might fail to cope with reality which is full of complexities and uncertainties. Sophisticated methods, conversely, are more flexible but cannot have a formal sense of optimality or completeness. Innovation of hybrid approaches is increasingly emerging to overcome the shortcomings of individual approaches. These algorithms are a combination of the strengths of other algorithms in order to strike a superior balance among the performance, robustness and computational cost. An example is where a hybrid algorithm combines a classical algorithm to solve the global path planning with an intelligent algorithm to solve the local obstacle avoidance. This will enable the robot to be subject to the optimality of the classical algorithm with the flexibility of the intelligent algorithm.

The success of hybrid strategies has been proved in a number of studies. As an example, the A + DWA (Dynamic Window Approach) combination has been demonstrated to be useful to navigate the dynamic environment. On a similar note, fuzzy logic when incorporated alongside the ant colony optimization has been applied to enhance the functionality of multi-robot systems.

**Table 2**  
A recent literature comparison based (Environment, Application).

| Algorithm             | Environment type | Application domain      | Reference    |
|-----------------------|------------------|-------------------------|--------------|
| Dijkstra's Algorithm  | Static Grid      | Indoor AGV              | [6]          |
| Dijkstra's Algorithm  | Static Known     | Warehouse Robotics      | [7]          |
| Modified Dijkstra     | Static Grid      | Large-scale Networks    | [5]          |
| PSO-Dijkstra Hybrid   | Static/Dynamic   | Autonomous Vehicles     | [9]          |
| A* Algorithm          | Static Grid      | AGV Navigation          | [11]         |
| A* Algorithm          | Static Grid      | Game AI                 | [13]         |
| Improved A*           | Static Grid      | Mobile Robots           | [25], [28]   |
| Hierarchical A*       | Static Known     | Large Parking Lots      | [20]         |
| Hybrid A*             | Semi-Dynamic     | Valet Parking           | [24]         |
| RFA-star              | 3D Static        | UAV (High Obstacle)     | [22]         |
| D* Algorithm          | Dynamic          | AGV Safety              | [31]         |
| D* Lite               | Dynamic          | Mobile Robots           | [34–36]      |
| Enhanced D* Lite      | Dynamic          | Autonomous Robots       | [35,36]      |
| Field D*              | Dynamic Terrain  | Outdoor Navigation      | [37]         |
| PRM                   | Static Complex   | High-DOF Manipulators   | [39,40]      |
| Improved PRM          | Static Complex   | Robot Arms              | [40]         |
| RRT                   | 3D Static        | UAV Navigation          | [44]         |
| RRT-Connect           | Static Complex   | Manipulator Planning    | [45]         |
| RRT*                  | Static Complex   | Continuum Arms          | [48]         |
| RRT*-Smart            | Static           | General Navigation      | [49]         |
| idb-RRT               | Dynamic          | Kinodynamic Planning    | [43]         |
| APF                   | Dynamic          | Real-time Avoidance     | [106,107]    |
| Improved APF          | Dynamic          | Multi-target            | [118]        |
| APF-NN Hybrid         | Dynamic          | Multi-Agent             | [63]         |
| Genetic Algorithm     | Static           | Mobile Robots (Static)  | [74]         |
| Improved GA           | Static           | Mobile Robots           | [74], [84]   |
| Adaptive GA           | 3D Dynamic       | Underwater AUV (3D)     | [83], [86]   |
| Multi-Objective GA    | Static/Dynamic   | Industrial Automation   | [82]         |
| GA-PSO Hybrid         | Static           | Multi-Joint Robots      | [80]         |
| GA-Deep RL            | Dynamic          | Missile Guidance        | [90]         |
| PSO                   | Static/Dynamic   | Mobile Robots           | [119–121]    |
| Self-Adaptive PSO     | Dynamic          | 2D Lidar Robots         | [125]        |
| Multi-Objective PSO   | Dynamic          | AGV (Dynamic)           | [122]        |
| PSO-DWA Hybrid        | Dynamic          | AGV Navigation          | [122]        |
| Hybrid S-curve-PSO    | Static           | Industrial Robots       | [121], [130] |
| PSO-RL Hybrid         | Dynamic          | AGV (Industrial)        | [131]        |
| PSO-Gauss             | 3D Dynamic       | Underwater AUV          | [127]        |
| Parallel PSO-Sparrow  | 3D Static        | UAV                     | [128]        |
| ACO                   | Static           | Multi-Robot Systems     | [132–134]    |
| Improved ACO          | Static           | Mobile Robots (Static)  | [137]        |
| Quantum ACO           | Dynamic          | Autonomous Driving      | [138]        |
| ACO-Q-Learning        | Dynamic          | Autonomous Vehicles     | [142]        |
| ACO-3D DWA            | 3D Dynamic       | AUV Communication       | [141]        |
| Fuzzy-ACO             | 3D Static        | 3D Laser Scanning       | [105], [139] |
| ABC                   | Static           | Mobile Robots           | [144–146]    |
| Hybrid Enhanced ABC   | Dynamic          | UAV (Threat Avoidance)  | [147], [150] |
| Multi-Objective ABC   | Static/Dynamic   | Mobile Robots           | [149]        |
| 3D ABC                | 3D Static        | UAV Path Planning       | [151]        |
| Chaotic ABC           | Dynamic          | UAV Identification      | [154]        |
| GOA                   | Static/Dynamic   | Mobile Robots           | [156–159]    |
| Online GOA            | Dynamic Unknown  | Unknown Environments    | [159]        |
| BFO                   | Static           | Humanoid Robots         | [160–163]    |
| Improved BFO          | Dynamic          | USV (Multi-constraint)  | [164]        |
| Spiral-BFO Hybrid     | Static           | Flexible Manipulator    | [165]        |
| Bio-inspired BFO      | Static           | Robotic Manipulators    | [168]        |
| Firefly Algorithm     | Static/Dynamic   | Mobile Robots           | [170–174]    |
| Multi-Objective FA    | Static           | Mobile Robots           | [171]        |
| Improved FA           | Static           | Industrial Robots       | [173]        |
| Hybrid Whale-FA       | Static/Dynamic   | Mobile Robots           | [177]        |
| FA-TPM                | Dynamic          | Cluttered Environments  | [180]        |
| 3D FA (UAV-AGV)       | 3D Cooperative   | Electric Power Patrol   | [179]        |
| SFLA                  | Static           | Robot Path Planning     | [181–184]    |
| Multi-Objective SFLA  | Static           | Mobile Robots           | [183]        |
| Adaptive SFLA         | Dynamic          | Underwater Robots       | [184]        |
| Improved SFLA         | Static           | Robot Control           | [187]        |
| SFLA (UAV Inspection) | 3D Static        | UAV Autonomous          | [185]        |
| GWO                   | 3D Static        | Multiple AUVs           | [188–190]    |
| Multi-Strategy GWO    | Static/Dynamic   | Robot Path Planning     | [191]        |
| Modified GWO          | Static           | Autonomous Robots       | [192]        |
| PSO-GWO Hybrid        | Static/Dynamic   | Autonomous Guided Robot | [193]        |
| RFO                   | Static/Dynamic   | Mobile Robots           | [195–197]    |
| RFO (Path Planning)   | Static           | Autonomous Robots       | [201]        |

(continued on next page)

Table 2 (continued).

| Algorithm                    | Environment type  | Application domain        | Reference      |
|------------------------------|-------------------|---------------------------|----------------|
| Fuzzy Logic Controller       | Dynamic           | Mobile Robots             | [92–99]        |
| Type-2 Fuzzy                 | Complex Dynamic   | Multi-Humanoid            | [234]          |
| Neuro-Fuzzy                  | Dynamic           | Mobile Robots             | [64,65], [102] |
| Fuzzy-GA Hybrid              | Dynamic           | UAV Control               | [103]          |
| Fuzzy-APF Hybrid             | Dynamic           | Mobile Robots             | [104]          |
| Fuzzy-ACO                    | 3D Static         | 3D Laser Scanning         | [105]          |
| DNN                          | Dynamic           | Wheeled Robots            | [60]           |
| Hybrid DNN                   | Dynamic           | Missile Guidance          | [69]           |
| Lightweight CNN              | Dynamic           | Real-time Detection       | [70]           |
| Ultra-fast DNN               | Dynamic           | Nano-UAV                  | [71]           |
| GBNN                         | Static            | Ship Hull Coverage        | [67]           |
| NN-H2/H <sub>∞</sub> Control | Dynamic           | Modified UAV              | [68]           |
| Q-Learning                   | Static/Dynamic    | Mobile Robots             | [207]          |
| Deep RL (Agricultural)       | Outdoor Dynamic   | Agricultural Robots       | [15]           |
| RIDER (RL)                   | Unstructured      | Unstructured Environments | [101]          |
| TD3 (Optimized)              | Dynamic Human     | Crowded Environments      | [204]          |
| Multi-Objective RL           | Space/Complex     | Multi-Body Trajectory     | [123]          |
| DWA                          | Dynamic           | Mobile Robot Formation    | [206]          |
| Improved DWA                 | Dynamic           | Robot Formation           | [206]          |
| VFH+T                        | Dynamic Unknown   | Unknown Environments      | [205]          |
| A*-DWA Hybrid                | Dynamic           | Mobile Robots             | [76]           |
| Grid-APF Hybrid              | Dynamic Uncertain | Uncertainty Environments  | [198]          |
| Bi-directional Adaptive      | Complex Dynamic   | Complex Dynamic           | [199]          |

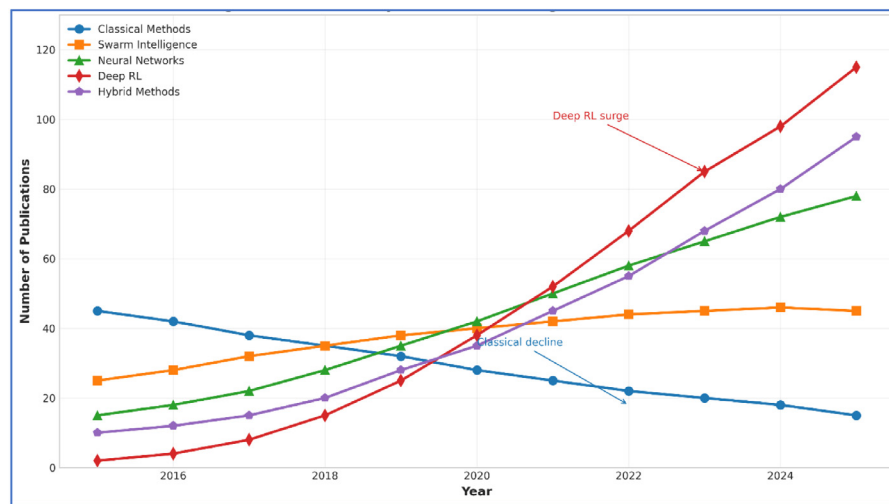


Fig. 32. Trend analysis chart showing the evolution of research in path planning.

#### 4. Discussion

After conducting an extensive examination of the literature, it is evident that navigational methods can be categorized into: Classical and Intelligent approaches. Classical ones are no longer common, because they have a lot of drawbacks, such as high computational complexity, sensitivity to local minima, difficulties with uncertainty, requirement for a precise response from the environment and a precise real-time sensing. These boundaries make it less convenient to use classical approaches for real-time applications, however, and make them perform in unpredictable ways. Despite all attempts to overcome these limitations, it has been seen that classical approaches still face problems of achieving the real time performance when compared to Intelligent approaches like Artificial Potential Field and hybrid algorithms. Classical strategies work best in familiar environments where there is prior information about the environment available.

Conversely, Intelligent strategies excel in unexpected situations, demonstrating their capacity to manage significant levels of uncertainty. They possess the qualities of being easily implemented, intelligent, and efficient, which makes them suitable for handling real-time navigation difficulties and yielding superior results when compared to classical systems. Nevertheless, Intelligent approaches have certain

drawbacks, such as increased processing time, intricate design, learning phases, memory demands, and incompatibility with low-cost robots. Table 3, demonstrate the difference in efficiency among the methods in various scenarios (Static and Dynamic).

At now, about 95% of research is conducted using Intelligent and Machine Learning algorithms. Fuzzy logic (FL), genetic algorithms (GA), and neural networks (NN) have made substantial contributions to the field of mobile robot navigation, accounting for more than 50% of the research articles in this area. Moreover, newly developed meta-heuristic algorithms such as Firefly Algorithm (FA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Shuffled Frog Leaping Algorithm (SFLA), and other miscellaneous algorithms have gained rapid popularity. Alongside these, modern ML-based approaches, particularly Deep Reinforcement Learning (DRL), Graph Neural Networks (GNNs), and Transformers, are rapidly emerging as the new state-of-the-art, demonstrating superior performance in complex, dynamic environments. Research articles employing Intelligent and ML-based approaches are more common than those employing classical methodologies. The classic method known as APF has demonstrated its effectiveness in guiding robots across surroundings that are partially recognized or completely unfamiliar. It has demonstrated superior performance in comparison to Cell Decomposition (CD) and Roadmaps (RM). The utilization of APF has had

**Table 3**  
A comparison of path planning methods efficiency based on (Path, Time).

| Algorithm                      | Path optimality (Static) | Path optimality (Dynamic) | Time cost (Static) | Time cost (Dynamic) |
|--------------------------------|--------------------------|---------------------------|--------------------|---------------------|
| Dijkstra                       | Low                      | Low                       | Fast               | Moderate            |
| A*                             | Moderate                 | Moderate                  | Fast               | Fast                |
| D*                             | Moderate                 | Low                       | Moderate           | Fast                |
| D* Lite                        | Low                      | Low                       | Moderate           | Fast                |
| PRM                            | Moderate                 | High                      | Moderate           | High                |
| RRT                            | Moderate                 | Moderate                  | Fast               | Fast                |
| RRT*                           | Moderate                 | Moderate                  | Moderate           | Moderate            |
| RRT*-smart                     | Moderate                 | Moderate                  | Fast               | Moderate            |
| Visibility Graph               | Low                      | High                      | Fast               | Slow                |
| Voronoi Diagram                | Low                      | Moderate                  | Fast               | Moderate            |
| Exact Cell Decomposition       | High                     | Moderate                  | Moderate           | Slow                |
| Adaptive Cell Decomposition    | Moderate                 | Moderate                  | Moderate           | Moderate            |
| Approximate Cell Decomposition | Low                      | Moderate                  | Fast               | Moderate            |
| Neural Networks                | High                     | High                      | High               | High                |
| Genetic Algorithms             | Moderate                 | Moderate                  | Moderate           | Moderate            |
| Fuzzy Logic                    | Moderate                 | Moderate                  | Fast               | Moderate            |
| Potential Field (PF)           | Low                      | Moderate                  | Fast               | Moderate            |
| Wavefront (PF)                 | Low                      | Moderate                  | Fast               | Moderate            |
| Depth-First (PF)               | Moderate                 | High                      | Moderate           | Slow                |
| Breadth-First (PF)             | Moderate                 | High                      | Moderate           | Slow                |
| Navigation Function            | Moderate                 | Moderate                  | Moderate           | Moderate            |
| PSO                            | Moderate                 | Moderate                  | Moderate           | Moderate            |
| ACO                            | Moderate                 | High                      | Moderate           | High                |
| BCO                            | Moderate                 | Moderate                  | Moderate           | Moderate            |
| GOA                            | Moderate                 | Moderate                  | Moderate           | Moderate            |
| BFO                            | Moderate                 | Moderate                  | Moderate           | Moderate            |
| Firefly                        | Moderate                 | Moderate                  | Moderate           | Moderate            |
| SELA                           | Moderate                 | Moderate                  | Moderate           | Moderate            |
| GWO                            | Moderate                 | Moderate                  | Moderate           | Moderate            |
| RWO                            | Moderate                 | Moderate                  | Moderate           | Moderate            |
| DQN                            | Moderate                 | Moderate                  | High               | Fast                |
| PPO                            | High                     | High                      | High               | Fast                |
| A3C                            | High                     | High                      | High               | Fast                |
| DDPG                           | High                     | High                      | High               | Fast                |
| TD3                            | High                     | High                      | High               | Fast                |
| SAC                            | High                     | High                      | High               | Fast                |
| GNNs                           | High                     | High                      | High               | Fast                |
| Transformers                   | High                     | High                      | High               | Fast                |

Rating Criteria: The qualitative ratings in this table are based on explicit quantitative ratings which have been taken from the reviewed literature. High Robustness: >90% success in dynamic obstacles environments; Moderate: 60–90%; Low: <60%. High Optimality: path length is within 5% of theoretical optimal; Moderate: path length is from 5 to 15% from the theoretical optimal; Low: path length is more than 15% from the theoretical optimal. Fast: less than 100 ms per cycle of planning decisions; Moderate: 100 ms – 1 s per cycle of planning decisions; Slow: greater than 1 s per cycle of planning decisions. High Scalability: more than 1000 nodes/cells; Moderate: 100–1000; Low: Less than 100.

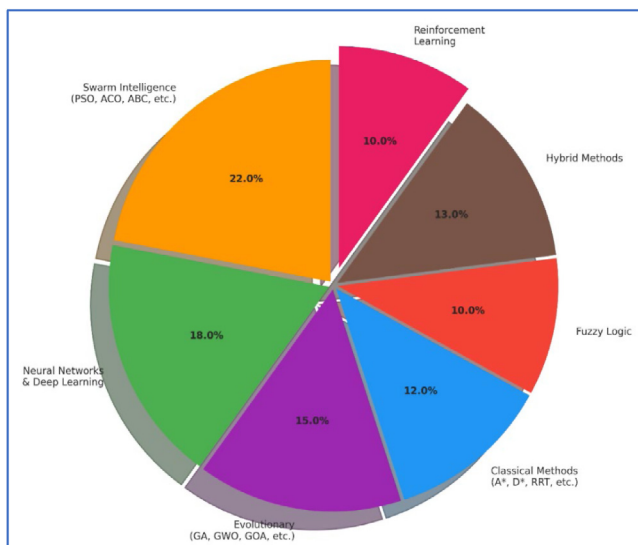


**Fig. 33.** Distribution of research publications by method category (2020–2025).

a significant increase over the past two decades in order to address navigation challenges involving both stationary and moving obstacles, as well as dynamic goal conditions.

One notable trade regime that is expressed in Table 3 is the trade-off between the cost and the performance of computation. Classical algorithms like Dijkstra and A-Star are optimal paths with the least time consumption in a static environment but their usefulness is less in dynamic environments. Traditional intelligent approaches such as particle swarm optimization (PSO) and genetic algorithms (GA) will provide better flexibility to uncertainty and moderate to high computing costs and will not provide any formal optimality program. In contrast, modern machine-learning-based algorithms (e.g., deep Q-networks (DQN), proximal policy optimization (PPO), graph neural networks (GNNs), and transformers) take a different approach: they require significant initial cost in training (high preprocessing time cost), but have low dynamic cost during deployment (low dynamic time cost) as well as high-quality path planning (high path quality). These features make these methods especially appropriate to real-time and autonomous decision-making in dynamic and complex tasks.

The examination of classical and intelligent methodologies has been expanded to encompass the field of 3D path planning for aerial and sub-surface vehicles. Classical approaches are inadequate for autonomous path planning in three-dimensional settings and are often augmented with fuzzy logic (FL), genetic algorithms (GA), and other methods to enhance their efficiency. The methods of deep reinforcement learning

(DRL) and graph neural network (GNN) are increasingly applied to three-dimensional path-planning problems of unmanned aerial vehicles (UAVs) and autonomous underwater vehicles (AUVs), with steadily better performance than traditional methods in both complicated aerial and aquatic settings.

Intelligent procedures are preferred over classical procedures in military and defense applications because of their efficiency in exploration, computation, promptness in response, flexibility in operation, and ability to use advanced intelligence systems and make autonomous decisions. These capabilities are further enhanced by the modern machine-learning-based solutions, offering a greater level of autonomy and adaptability that is very important in the operations that are performed in unpredictable and hostile warfare conditions that lack direct human intervention. These systems can be controlled remotely through GPS-based control systems or through control rooms and deep-reinforcement-learning agents will make real-time tactical decisions.

In the field of mobile robot navigation, intelligent and ML-based approaches have become popular, because they are able to effectively deal with uncertainty in real time. In summary of this study, the different types of mobile robot navigation methods presented here can be categorized as follows with the following observations:

- Intelligent strategies outperform conventional approaches in effectively handling environmental uncertainty.
- Intelligent and ML-based approaches are particularly well-suited for resolving real-time navigation problems.
- Advanced machine-learning paradigms, including DRL, GNNs, and transformer architectures, have even better quality in the resulting trajectories in both fixed and dynamic operating scenarios, but they need a computationally-intensive offline training process.
- Striking characteristic of recent ML-based algorithms is the trade-off between considerable and upfront computational costs incurred during training and less dynamic costs incurred during the actual execution in real-time.
- Research publications on navigation in dynamic contexts are relatively scarce compared to those focused on static situations.
- There is a lack of research that specifically addresses the navigation of robots in dynamic conditions for moving objective problems, rather than moving obstacle problems.
- The majority of publications mostly focus on simulation analysis, with only a few papers dedicated to real-time applications.
- Research publications on the navigation of multiple mobile robot systems are scarce in comparison to those on single robot systems.
- The number of papers on hybrid algorithms is lower compared to standalone algorithms.

The utilization of newly developed algorithms, such as Shuffled Frog Leaping Algorithm, Cuckoo Search, Invasive Weed Optimization, Bat Algorithm, Harmony Search, Differential Evolution, Bacterial Foraging Optimization, Artificial Bee Colony, and Firefly Algorithm, holds significant promise for traversing intricate unfamiliar environments characterized by substantial levels of uncertainty. These algorithms can be used to develop innovative hybrid approaches. Hybridization solutions can enhance the efficiency of traditional operations by integrating them with intelligent methods.

## 5. Conclusion

This review has presented an in-depth and systematic analysis of path planning strategies for autonomous mobile robots, covering both classical and intelligent approaches across static, dynamic, and uncertain environments, several path planning algorithms has also been comprehended and evaluated, along with suggesting potential methods for future research. In this review, the 204 articles that were introduced

are spread as shown in Fig. 34, according to the search strategy. Classical approaches like Dijkstra, A\*, D\* and PRM have been used for a long time to give a theoretical support for navigation, and they are determinist, complete and optimal in structured domains. However, they are not capable of adapting rapidly to dynamic changes, have high computational load in large search spaces, and are dependent on predefined maps to work, but they are not adaptable in real-world scenarios due to their high computational load in large search spaces, requirement of predefined maps, and inability to respond to dynamic changes quickly. Nevertheless, the field has been moved towards more adaptable, robust and scalable algorithms by intelligent and bio-inspired algorithms such as GA, PSO, ACO, BFO, GWO, GOA and recently the new RFO. It is characterized by a change towards modern machine-learning techniques, such as reinforcement learning and deep learning, a paradigm change and the robot is able to learn from experience and act in complex and unstructured environments with an increased level of autonomy than ever before. These approaches are highly effective in dynamic, partially observable and multi-robot settings, where they can boost robustness in the presence of uncertainties, provide flexibility in decision making and have the potential to learn from experience.

One salient aspect to be noted is the shift towards hybrid systems that combine the effectiveness and assurances of traditional approaches with the flexibility and learning power of intelligent techniques. Reinforcement learning and deep neural models, when used along with heuristic search or metaheuristics, are becoming more and more than responsive and optimal navigators. The increased interest in multi-robot systems, in which collaborative planning is used for scalability of large robot fleets in logistics, agriculture, surveillance and rescue applications is another key aspect. Yet, despite all these advances, there are still a number of challenges ahead to make simulation-based results “real” in practical deployment, to deal with the noise of the sensors and limitations of the hardware, to maintain safety in human–robot mixed environments, and to ensure explainability of “black boxes” such as AI models.

In the future, the direction in which research in robot navigation is headed is straightforward, and it is going towards system level integration, away from algorithmic exploration. This demands more efficient algorithms as well as their integration in cognitive architectures which enable an integration of perception, planning and control in real-time in a robust manner, under uncertainty. The future path planning solutions for high impact applications like autonomous vehicles, medical robotics, disaster and smart city operations must be optimal, adaptive, interpretable and safe.

## Future Research Directions

- Hybrid Path Planning Frameworks: Prioritize the fusion of classical optimal algorithms with intelligent metaheuristics or reinforcement learning to combine theoretical guarantees with adaptive behavior in uncertain environments.
- In this current review, the focus is placed on the analytical and algorithmic decision-making logic of path planning, while emerging friction-aware and energy-aware path planning algorithms are not included. Friction-aware approaches include physical-layer modeling of terrain friction coefficients and the modeling of the traction dynamics, whereas energy-aware approaches need to incorporate motor consumption models, battery state-of-charge dynamics and regenerative braking strategies. The topics are suitable for a separate survey in between the algorithmic path planning and physical robot design. Future studies could examine how energy usage costs can be incorporated into existing path planning algorithms like A\* and DRL, and see how they can be adapted to also consider the energy needed to navigate a rough, real-world environment.
- Real-Time and Scalable Deployment: Advance lightweight and computationally efficient algorithms suitable for embedded systems, ensuring robust performance in hardware-constrained robots operating in real-time.

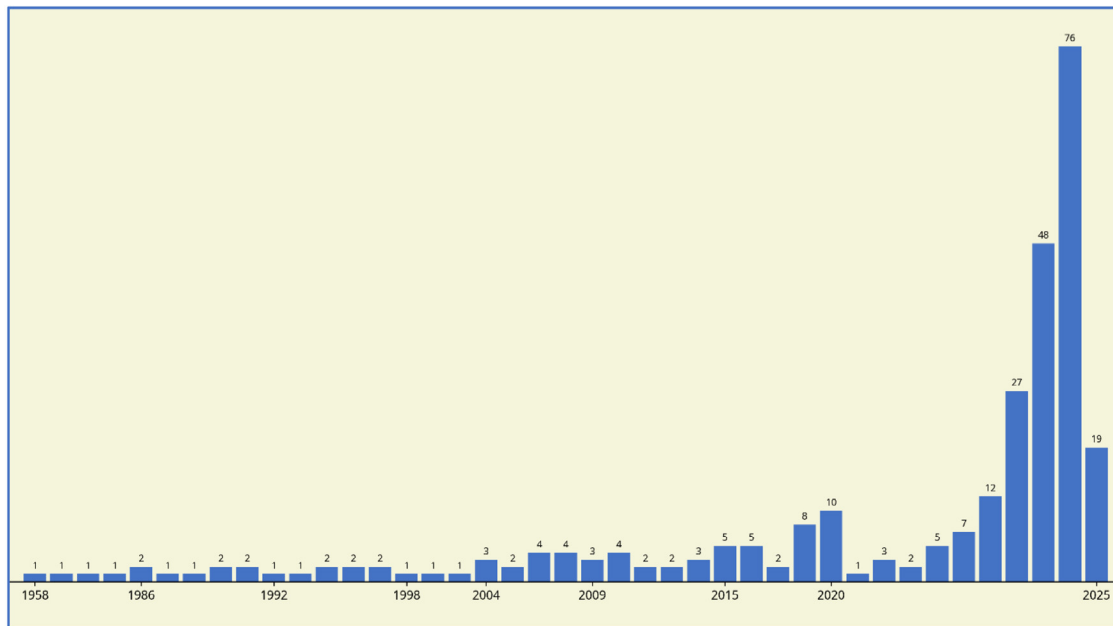


Fig. 34. Number of publications per year.

- **Explainable and Trustworthy AI:** Develop interpretable intelligent planning systems that can provide transparency in decision-making, which is crucial for safety-critical applications such as healthcare robots or autonomous driving.
- **Multi-Robot Coordination:** Expand scalable frameworks for cooperative path planning in large robot swarms, focusing on energy efficiency, communication overhead, and resilience to dynamic obstacles.
- **Human–Robot Interaction (HRI):** Investigate safe and intuitive path planning strategies for robots navigating in human-populated environments, emphasizing prediction of human motion and adaptive social navigation.
- **Benchmarking and Standardization:** Establish unified evaluation benchmarks and real-world testing protocols to fairly compare algorithms and accelerate their translation from simulation to deployment.

Eventually, path planning for autonomous mobile robots is undergoing a paradigm shift from static, deterministic solutions to dynamic, adaptive, and hybrid models. The integration of intelligent optimization, deep learning, and bio-inspired approaches represents not only a methodological evolution but also a decisive step towards reliable, explainable, and scalable autonomy in complex real-world environments. This review consolidates the current state of the art while articulating the challenges and opportunities that define the next generation of research, positioning path planning as a cornerstone for achieving trustworthy autonomy in robotics

#### CRediT authorship contribution statement

**Suhaib Al-Ansarry:** Conceptualization. **Salah Al-Darraj:** Conceptualization. **Asmaa Shareef:** Conceptualization, Methodology, Writing – original draft. **Hamid Ali Abed AL-Asadi:** Methodology, Writing – original draft. **Zaid Ameen Abduljabbar:** Methodology, Writing – original draft. **Hothefa Shaker Jassim:** Methodology, Writing – original draft, Writing – review & editing. **Vincent Omollo Nyangaresi:** Writing – review & editing. **Korondi Péter:** Writing – review & editing, Supervision. **Husam A. Neamah:** Conceptualization, Writing – review & editing, Supervision.

#### Consent for publication

All authors have reviewed and approved the final manuscript and consent to its submission to the journal.

#### Ethics statement

Not applicable. This study does not involve human participants, animals, or data from social media.

#### Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

#### Declaration of competing interest

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

#### Acknowledgments

The authors wish to thank the support of the Hungarian Research Fund (OTKA K143595).

#### Data availability

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

#### References

- [1] B.K. Patle, Intelligent Navigational Strategies for Multiple Wheeled Mobile Robots using Artificial Hybrid Methodologies (Thesis), 2016.
- [2] D. Castellanos-Cárdenas, N.L. Posada, A. Orozco-Duque, L.M. Sepúlveda-Cano, F. Castrillón, O.E. Camacho, R.E. Vázquez, A review on data-driven model-free sliding mode control, *Algorithms* 17 (12) (2024) 543.
- [3] H. Mazaheri, S. Goli, A. Nourollah, A survey of 3D space path-planning methods and algorithms, *ACM Comput. Surv.* 57 (1) (2024) 1–32.
- [4] E.W. Dijkstra, A note on two problems in connexion with graphs, in: Edsger Wybe Dijkstra: His Life, Work, and Legacy, 2022, pp. 287–290.

- [5] O.A. Gbadamosi, D.R. Aremu, Design of a modified dijkstra's algorithm for finding alternate routes for shortest-path problems with huge costs, in: 2020 International Conference in Mathematics, Computer Engineering and Computer Science, ICMCECS, 2020, pp. 1–6.
- [6] Y. Sun, M. Fang, Y. Su, AGV path planning based on improved dijkstra algorithm, J. Phys. Conf. Ser. 1746 (1) (2021) 012052.
- [7] S.A. Fadzli, S.I. Abdulkadir, M. Makhtar, A.A. Jamal, Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries, in: 2015 2nd International Conference on Information Science and Security, ICISS, 2015, pp. 1–4.
- [8] H.I. Kang, B. Lee, K. Kim, Path planning algorithm using the particle swarm optimization and the improved dijkstra algorithm, in: 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, vol. 2, 2008, pp. 1002–1004.
- [9] S. Sundarraj, R.V.K. Reddy, M.B. Basam, G.H. Lokesh, F. Flammini, R. Natarajan, Route planning for an autonomous robotic vehicle employing a weight-controlled particle swarm-optimized dijkstra algorithm, IEEE Access 11 (2023) 92433–92442.
- [10] D. Ferguson, M. Likhachev, A. Stentz, A guide to heuristic-based path planning, in: Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, ICAPS, 2005, pp. 9–18.
- [11] D. Zhang, C. Chen, G. Zhang, AGV path planning based on improved A-star algorithm, in: 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference, IAEAC, vol. 7, 2024, pp. 1590–1595.
- [12] P. Bagad, P. Dahatonde, R. Gagare, I. Athare, G. Ghule, Optimizing pathfinding in dynamic environments: A comparative study of a\* and D-lite algorithms, in: 2024 IEEE Pune Section International Conference (PuneCon), 2024, pp. 1–11.
- [13] X. Cui, H. Shi, A\*-based pathfinding in modern computer games, Int. J. Comput. Sci. Netw. Secur. 11 (1) (2011) 125–130.
- [14] Liu, Hui, Te. Shan, Wei. Wang, Automatic routing study of spacecraft cable based on A-star algorithm, in: 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference, ITOEC, IEEE, 2020, pp. 716–719.
- [15] J. Yang, J. Ni, Y. Li, J. Wen, D. Chen, The intelligent path planning system of agricultural robot via reinforcement learning, Sensors 22 (12) (2022) 4316.
- [16] K. Shanmugam, D. Durgabhavani, A\* star algorithm visualization, 2024.
- [17] T.-E. Fan, S.-M. Liu, H. Yang, P.-H. Li, B. Qu, A fast active balancing strategy based on model predictive control for lithium-ion battery packs, Energy 279 (2023) 128028.
- [18] F. Jing, H. Liu, X. Gao, Research on cable automatic routing of spacecraft based on MBD and A-star algorithm, in: Ninth International Conference on Mechanical Engineering, Materials, and Automation Technology (MMEAT 2023), vol. 12801, 2023, pp. 1506–1511.
- [19] P.T. Kyaw, A.V. Le, L. Yi, P. Veerajagadheswar, M.B. Vu, M.R. Elara, Greedy heuristics for sampling-based motion planning in high-dimensional state spaces, 2024, arXiv preprint arXiv:2405.03411.
- [20] L. Cheng, C. Liu, B. Yan, Improved hierarchical A-star algorithm for optimal parking path planning of the large parking lot, in: 2014 IEEE International Conference on Information and Automation, ICIA, 2014, pp. 695–698.
- [21] J. Yao, C. Lin, X. Xie, A.-J. Wang, C.-C. Hung, Path planning for virtual human motion using improved a\* star algorithm, in: 2010 Seventh International Conference on Information Technology: New Generations, 2010, pp. 1154–1158.
- [22] W. Zhang, J. Li, W. Yu, P. Ding, J. Wang, X. Zhang, Algorithm for UAV path planning in high obstacle density environments: RFA-star, Front. Plant Sci. 15 (2024) 1391628.
- [23] T. Goto, T. Kosaka, H. Noborio, On the heuristics of A\* or a algorithm in ITS and robot path-planning, in: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 2, 2003, pp. 1159–1166.
- [24] S. Sedighi, D.-V. Nguyen, K.-D. Kuhnert, Guided hybrid A-star path planning algorithm for valet parking applications, in: 2019 5th International Conference on Control, Automation and Robotics, ICCAR, 2019, pp. 570–575.
- [25] Y. Li, J. Yang, Path planning and obstacle avoidance technology for mobile robots using improved A-star algorithm, in: 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology, AINIT, 2024, pp. 121–124.
- [26] S. Koenig, M. Likhachev, D. Furcy, Lifelong planning A\*, Artificial Intelligence 155 (1–2) (2004) 93–146.
- [27] K. Xie, J. Qiang, H. Yang, Research and optimization of D-start lite algorithm in track planning, IEEE Access 8 (2020) 161920–161928.
- [28] B. Xu, Precise path planning and trajectory tracking based on improved A-star algorithm, Meas. Control. 57 (8) (2024) 1025–1037.
- [29] N. Khelif, N. Khraief, S. Belghith, IFTOMM International Symposium on Robotics and Mechatronics, Springer Nature, Switzerland, 2024, pp. 251–262.
- [30] A. Stentz, The D\* Algorithm for Real-Time Planning of Optimal Traverses, Carnegie Mellon University, The Robotics Institute, 1994.
- [31] W. Zheng, H. Yu, Research on the safety of AGV path planning based on d\* algorithm, in: 2024 6th International Conference on Artificial Intelligence and Computer Applications, ICAICA, 2024, pp. 216–220.
- [32] V. Yevsieiev, A. Abu-Jassar, S. Maksymova, A. Alkhalaileh, Route constructing for a mobile robot based on the D-star algorithm, 2024.
- [33] A. Stentz, The focussed D\* algorithm for real-time replanning, in: IJCAI, vol. 95, 1995, pp. 1652–1659.
- [34] S. Koenig, M. Likhachev, D\* lite, in: Eighteenth National Conference on Artificial Intelligence, 2002, pp. 476–483.
- [35] D.D. Lama, Enhancing the D\* Lite Algorithm for Bypassable Obstacles in Dynamic Environment Navigation (Master's thesis), State University of New York at Binghamton, 2023.
- [36] V. Ganapathy, S.C. Yun, T.W. Chien, Enhanced D\* lite algorithm for autonomous mobile robot, Int. J. Appl. 1 (1) (2011) 58–73.
- [37] D. Ferguson, A. Stentz, Field D\*: an interpolation-based path planner and replanner, in: Robotics Research: Results of the 12th International Symposium ISRR, 2007, pp. 239–253.
- [38] F. Gul, I. Mir, U. Gul, A. Forestiero, A review of space exploration and trajectory optimization techniques for autonomous systems: Comprehensive analysis and future directions, in: International Conference on Pervasive Knowledge and Collective Intelligence on Web and Social Media, Springer Nature, Switzerland, 2022, pp. 125–138.
- [39] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. 12 (4) (1996) 566–580.
- [40] S. Chen, G. Yang, G. Cui, S. Yi, L. Wu, Improved path planning and controller design based on PRM, IEEE Access (2025).
- [41] S. LaValle, Rapidly-Exploring RandOm Trees: A New Tool for Path Planning, Research Report 9811, 1998.
- [42] S.M. LaValle, Planning Algorithms, Cambridge University Press, 2006.
- [43] J. Ortiz-Haro, W. Hönig, V.N. Hartmann, M. Toussaint, L. Righetti, Idb-rrt: Sampling-based kinodynamic motion planning with motion primitives and trajectory optimization, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2024, pp. 10702–10709.
- [44] D.K. Muhsen, F.A. Raheem, A.T. Sadiq, A systematic review of rapidly exploring random tree RRT algorithm for single and multiple robots, Cybern. Inf. Technol. 24 (3) (2024).
- [45] J.J. Kuffner, S.M. LaValle, RRT-connect: An efficient approach to single-query path planning, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation, vol. 2, 2000, pp. 995–1001.
- [46] G. Oriolo, M. Vendittelli, L. Freda, G. Troso, The SRT method: Randomized strategies for exploration, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 5, 2004, pp. 4688–4694.
- [47] Z. Yiping, G. Jian, Z. Ruilei, C. Qingwei, A SRT-based path planning algorithm in unknown complex environment, in: The 26th Chinese Control and Decision Conference (2014 CCDC), 2014, pp. 3857–3862.
- [48] B.H. Meng, I.S. Godage, I. Kanj, Rrt\*-based path planning for continuum arms, IEEE Robot. Autom. Lett. 7 (3) (2022) 6830–6837.
- [49] F. Islam, J. Nasir, U. Malik, Y. Ayaz, O. Hasan, RRT\*-smart: Rapid convergence implementation of rrt\* towards optimal solution, in: 2012 IEEE International Conference on Mechatronics and Automation, 2012, pp. 1651–1656.
- [50] Z. Hu, B. Shirinzadeh, Comparative analysis of UAV path planning algorithms based on RRT, in: Proceedings of the 19th International Conference on Intelligent Unmanned Systems: ICIUS 2023 1248, Springer Nature, 2023, p. 273.
- [51] N. Nilsson, Shakey the robot, sri ai center, 1984, Technical Note 323.
- [52] H. Choset, Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph, California Institute of Technology, 1996.
- [53] W. Regli, Robot Lab: Robot Path Planning, Lectures Notes of Department of Computer Science, Drexel University, 2007.
- [54] R.A. Brooks, Planning collision-free motions for pick-and-place operations, Int. J. Robot. Res. 2 (4) (1983) 19–44.
- [55] D.J. Zhu, Constraint Reformulation and Graph Searching Techniques in Hierarchical Path Planning, Report No. STANCS-98-1279, 1989.
- [56] H. Noborio, T. Naniwa, S. Arimoto, A quadtree-based path-planning algorithm for a mobile robot, J. Robot. Syst. 7 (4) (1990) 555–574.
- [57] H. Samet, An overview of quadtrees, octrees, and related hierarchical data structures, Theor. Found. Comput. Graph. CAD (1988) 51–68.
- [58] S.K. Debnath, R. Omar, S. Bagchi, E.N. Sabudin, M.H.A.S. Kandar, K. Foysool, T.K. Chakraborty, Different cell decomposition path planning methods for unmanned air vehicles-a review, in: Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019: NUSYS'19, Springer Singapore, 2021, pp. 99–111.
- [59] G.E. Jan, C. Luo, C.-Y. Yang, H.-C. Hsieh, A survey of cell decomposition-based path planning, in: 2023 4th International Conference on Artificial Intelligence, Robotics and Control, AIRC, 2023, pp. 83–87.
- [60] M.G. Abdalnasser, M. Elsamanty, A.A. Ibrahim, Trajectory tracking of wheeled mobile robot through system identification and control using deep neural network, Eng. Res. J. 183 (3) (2024) 1–17.
- [61] A.K. Abbas, Y. Al Mashhadany, M.J. Hameed, S. Algburi, Review of intelligent control systems with robotics, Indones. J. Electr. Eng. Inf. (IJEEI) 10 (4) (2022) 734–753.

- [62] X. Bian, W. Zhao, L. Tang, H. Zhao, X. Mei, FastSLAM-MO-PSO: A robust method for simultaneous localization and mapping in mobile robots navigating unknown environments, *Appl. Sci.* 14 (22) (2024) 10268.
- [63] Z. Zhu, Z. Zhang, H. Zhou, X.-B. Chen, Adaptive path planning for multi-agent systems using improved artificial potential field with neural network approximation, *Eng. Lett.* 32 (3) (2024).
- [64] M.H. Haider, Z. Wang, A.A. Khan, H. Ali, H. Zheng, S. Usman, R. Kumar, M.U.M. Bhutta, P. Zhi, Robust mobile robot navigation in cluttered environments based on hybrid adaptive neuro-fuzzy inference and sensor fusion, *J. King Saud Univ. Comput. Inf. Sci.* 34 (10) (2022) 9060–9070.
- [65] M.K. Muni, D.R. Parhi, P.B. Kumar, C. Sahu, S. Kumar, Towards motion planning of humanoid using a fuzzy embedded neural network approach, *Appl. Soft Comput.* 119 (2022) 108588.
- [66] L.I. Bo, W. Tian, C. Zhang, F. Hua, G. Cui, Y. Li, Positioning error compensation of an industrial robot using neural networks and experimental study, *Chin. J. Aeronaut.* 35 (2) (2022) 346–360.
- [67] X. Wang, C. Xu, D. Ji, Coverage path planning for ship hull based on improved GBNN algorithm, in: 2024 43rd Chinese Control Conference, CCC, 2024, pp. 4451–4456.
- [68] F.W. Alsaade, H. Jahanshahi, Q. Yao, M.S. Al-zahrani, A.S. Alzahrani, A new neural network-based optimal mixed H<sub>2</sub>/H<sub>∞</sub> control for a modified unmanned aerial vehicle subject to control input constraints, *Adv. Space Res.* 71 (9) (2023) 3631–3643.
- [69] J. Farooq, M.A. Bazaz, Hybrid deep neural network for data-driven missile guidance with maneuvering target, *Def. Sci. J.* 73 (5) (2023).
- [70] J. Yun, D. Jiang, Y. Liu, Y. Sun, B. Tao, J. Kong, J. Tian, X. Tong, M. Xu, Z. Fang, Real-time target detection method based on lightweight convolutional neural network, *Front. Bioeng. Biotechnol.* 10 (2022) 861286.
- [71] L. Lamberti, L. Bellone, L. Macan, E. Natalizio, F. Conti, D. Palossi, L. Benini, Distilling tiny and ultra-fast deep neural networks for autonomous navigation on nano-UAVs, *IEEE Internet Things J.* (2024).
- [72] H.J. Bremermann, *The Evolution of Intelligence: The Nervous System As a Model of Its Environment*, University of Washington, Department of Mathematics, 1958.
- [73] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications To Biology, Control, and Artificial Intelligence*, MIT Press, 1992.
- [74] M.N. Ab Wahab, A. Nazir, A. Khalil, W.J. Ho, M.F. Akbar, M.H.M. Noor, A.S.A. Mohamed, Improved genetic algorithm for mobile robot path planning in static environments, *Expert Syst. Appl.* 249 (2024) 123762.
- [75] Z. Gyenes, L. Bölöni, E.G. Szádeczky-Kardoss, Can genetic algorithms be used for real-time obstacle avoidance for lidar-equipped mobile robots? *Sensors* 23 (6) (2023) 3039.
- [76] Y. Li, J. Zhao, Z. Chen, G. Xiong, S. Liu, A robot path planning method based on improved genetic algorithm and improved dynamic window approach, *Sustainability* 15 (5) (2023) 4656.
- [77] V.M. Vidyarthi, D. Mishra, A. Mittal, K. Bandyopadhyay, S. Pundhir, N. Varshney, A non linear system controlling using constrained based genetic algorithm, in: 2024 5th International Conference on Smart Electronics and Communication, ICOSEC, 2024, pp. 86–91.
- [78] W. Tang, Optimization of urban terminal delivery routes using fuzzy genetic algorithm and its practical application, *J. Comput. Methods Sci. Eng.* (2025) 14727978251323125.
- [79] H. Kim, S. Choi, J. Son, J. Park, C. Kwon, Neural genetic search in discrete spaces, 2025, arXiv preprint arXiv:2502.10433.
- [80] R. Yao, J. Wang, Y. Lv, D. Wang, The time-impact optimal trajectory planning for multi-joint robot based on GA-PSO, in: 2024 4th International Conference on Computer, Control and Robotics, ICCCR, 2024, pp. 284–288.
- [81] Y. Tang, S. Qi, L. Zhu, X. Zhuo, Y. Zhang, F. Meng, Obstacle avoidance motion in mobile robotics, *J. Syst. Simul.* 36 (1) (2024) 1–26.
- [82] K.S. Suresh, R. Venkatesan, S. Venugopal, Mobile robot path planning using multi-objective genetic algorithm in industrial automation, *Soft Comput.* 26 (15) (2022) 7387–7400.
- [83] K. Hao, J. Zhao, Z. Li, Y. Liu, L. Zhao, Dynamic path planning of a three-dimensional underwater AUV based on an adaptive genetic algorithm, *Ocean Eng.* 263 (2022) 112421.
- [84] D. Li, L. Wang, J. Cai, A. Wang, T. Tan, J. Gui, Research on path planning of mobile robot based on improved genetic algorithm, *Int. J. Model. Simul. Sci. Comput.* 14 (06) (2023) 2341030.
- [85] B.J. Yu, *A Genetic Algorithm Framework using Variable Length Chromosomes for Vehicle Maneuver Planning* (PhD diss.), Massachusetts Institute of Technology, 2022.
- [86] K. Hao, J. Zhao, Z. Li, Y. Liu, L. Zhao, Dynamic path planning of a three-dimensional underwater AUV based on an adaptive genetic algorithm, *Ocean Eng.* 263 (2022) 112421.
- [87] S.R. Bassolillo, G. Raspaolo, L. Blasi, E. D'Amato, I. Notaro, Path planning for fixed-wing unmanned aerial vehicles: An integrated approach with theta\* and clothoids, *Drones* 8 (2) (2024) 62.
- [88] J. Luo, Y. Tian, Z. Wang, Research on unmanned aerial vehicle path planning, *Drones* 8 (2) (2024) 51.
- [89] T. Wu, Z. Zhang, Y. Zeng, Y. Zhang, Mixed-integer programming model and hybrid local search genetic algorithm for human–robot collaborative disassembly line balancing problem, *Int. J. Prod. Res.* 62 (5) (2024) 1758–1782.
- [90] S. Xu, W. Bi, A. Zhang, Y. Wang, A deep reinforcement learning approach incorporating genetic algorithm for missile path planning, *Int. J. Mach. Learn. Cybern.* 15 (5) (2024) 1795–1814.
- [91] J. Zhang, Y. Zhan, Y. Cao, Research on the adaptability of unmanned aerial vehicle route planning based on genetic algorithm in complex environments, in: 2024 IEEE 2nd International Conference on Electrical, Automation and Computer Engineering, ICEACE, 2024, pp. 105–112.
- [92] L.A. Zadeh, Fuzzy sets, *Inf. Control* 8 (3) (1965) 338–353.
- [93] A.K. Abbas, Y. Al Mashhadany, M.J. Hameed, S. Algburi, Review of intelligent control systems with robotics, *Indones. J. Electr. Eng. Inf. (IJEEI)* 10 (4) (2022) 734–753.
- [94] G. Castellano, G. Attolico, A. Distanto, Automatic generation of fuzzy rules for reactive robot controllers, *Robot. Auton. Syst.* 22 (2) (1997) 133–149.
- [95] V. Kumar, A. Mistri, Fuzzy logic-based synchronization of trajectory planning and obstacle avoidance for RRP SCARA robot, *Int. J. Interact. Des. Manuf. (IJIDeM)* (2025) 1–14.
- [96] A. Hentout, A. Maoudj, M. Aouache, A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots, *Artif. Intell. Rev.* 56 (4) (2023) 3369–3444.
- [97] P.M. Tuan, N.D. Tai, T.Q. Huy, N.T. Thinh, Flexible path planning of mobile robot for avoiding the dynamic obstacles using fuzzy controllers, *Int. J. Mech. Eng. Robot. Res.* 13 (1) (2024) 126–132.
- [98] J.G. Pérez-Juárez, J.R. García-Martínez, O.A. Barra-Vázquez, E.E. Cruz-Miguel, L.F. Olmedo-García, A. Medina Santiago, Fuzzy logic controller for a terrestrial mobile robot based on the global positioning system, in: 2024 IEEE International Conference on Engineering Veracruz, ICEV, 2024, pp. 1–7.
- [99] A. Hentout, A. Maoudj, A. Kouider, Shortest path planning and efficient fuzzy logic control of mobile robots in indoor static and dynamic environments, *Sci. Technol.* 27 (1) (2024) 21–36.
- [100] A. Rout, G.B. Mahanta, B.B. Biswal, S.V. Raj, D. BBVL, Application of fuzzy logic in multi-sensor-based health service robot for condition monitoring during pandemic situations, *Robot. Intell. Autom.* 44 (1) (2024) 96–107.
- [101] S. Siva, M. Wigness, RIDER: Reinforcement-based inferred dynamics via emulating rehearsals for robot navigation in unstructured environments, in: 2024 IEEE International Conference on Robotics and Automation, ICRA, 2024, pp. 10566–10573.
- [102] A. Mostafanasab, M.B. Menhaj, M. Shamshirsaz, R. Fesharakifard, A novel mobile robot path planning method based on neuro-fuzzy controller, *AUT J. Math. Comput.* 6 (1) (2025) 41–53.
- [103] O. Rodríguez-Abreo, J. Rodríguez-Reséndiz, A. García-Cerezo, J.R. García-Martínez, Fuzzy logic controller for UAV with gains optimized via genetic algorithm, *Heliyon* 10 (4) (2024).
- [104] Y. Mhanni, Y. Lagmich, Enhanced obstacle avoidance and intelligent navigation for mobile robots: An integrated approach using fuzzy logic and an optimized APF method, *Math. Model. Eng. Probl.* 10 (6) (2023).
- [105] J. Song, Y. Pu, X. Xu, Adaptive ant colony optimization with sub-population and fuzzy logic for 3D laser scanning path planning, *Sensors* 24 (4) (2024) 1098.
- [106] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.
- [107] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Robot. Res.* 5 (1) (1986) 90–98.
- [108] S.M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.
- [109] S. Hacoheh, S. Shoval, N. Shvalb, Probability navigation function for stochastic static environments, *Int. J. Control. Autom. Syst.* 17 (2019) 2097–2113.
- [110] A. Pal, R. Tiwari, A. Shukla, A focused wave front algorithm for mobile robot path planning, in: *Hybrid Artificial Intelligent Systems: 6th International Conference, HAIS 2011, Wroclaw, Poland, May (2011) 23–25, Proceedings, Part I* 6, 2011, pp. 190–197.
- [111] A. Lazarowska, Comparison of discrete artificial potential field algorithm and wave-front algorithm for autonomous ship trajectory planning, *IEEE Access* 8 (2020) 221013–221026.
- [112] M. Planning, *Robot Motion Planning*, Boston, 1991.
- [113] P.C. Iloh, A comprehensive and comparative study of DFS, BFS, and a\* search algorithms in a solving the maze transversal problem, *Int. J. Soc. Sci. Sci. Stud.* 2 (2) (2022) 482–490.
- [114] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction To Algorithms*, third ed., MIT Press, 2009.
- [115] S. Hacoheh, S. Shoval, N. Shvalb, Probability navigation function for stochastic static environments, *Int. J. Control. Autom. Syst.* 17 (2019) 2097–2113.
- [116] E. Rimón, *Exact Robot Navigation using Artificial Potential Functions*, Yale University, 1990.
- [117] D.E. Koditschek, Robot planning and control via potential functions, *Robot. Rev.* (1989) 349.
- [118] J. Yang, H. Zhang, P. Ning, Path planning and trajectory optimization based on improved APF and multi-target, *IEEE Access* 11 (2023) 139121–139132.

- [119] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
- [120] X. Bian, W. Zhao, L. Tang, H. Zhao, X. Mei, FastSLAM-MO-PSO: A robust method for simultaneous localization and mapping in mobile robots navigating unknown environments, *Appl. Sci.* 14 (22) (2024) 10268.
- [121] B.K. Patle, S.-L. Chen, A. Singh, S.K. Kashyap, Optimal trajectory planning of the industrial robot using hybrid S-curve-PSO approach, *Robot. Intell. Autom.* 43 (2) (2023) 153–174.
- [122] T.-K. Dao, T.-G. Ngo, J.-S. Pan, T.-T.-T. Nguyen, T.-T. Nguyen, Enhancing path planning capabilities of automated guided vehicles in dynamic environments: Multi-objective PSO and dynamic-window approach, *Biomimetics* 9 (1) (2024) 35.
- [123] C.J. Sullivan, Low-Thrust Trajectory Design in Multi-Body Systems Via Multi-Objective Reinforcement Learning (PhD diss.), University of Colorado at Boulder, 2022.
- [124] G.S. Kashyap, D. Mahajan, O.C. Phukan, A. Kumar, A.E.I. Brownlee, J. Gao, From simulations to reality: Enhancing multi-robot exploration for urban search and rescue, 2023, arXiv preprint arXiv:2311.16958.
- [125] J. Fusic, R. Sitharthan, Self-adaptive learning particle swarm optimization-based path planning of mobile robot using 2D lidar environment, *Robotica* 42 (4) (2024) 977–1000.
- [126] P.K. Das, H.S. Behera, P.K. Jena, B.K. Panigrahi, An intelligent multi-robot path planning in a dynamic environment using improved gravitational search algorithm, *Int. J. Autom. Comput.* 18 (2021) 1032–1044.
- [127] W. Gan, L. Su, Z. Chu, A PSO-enhanced Gauss pseudospectral method to solve trajectory planning for autonomous underwater vehicles, *Math. Biosci. Eng.* 20 (7) (2023) 11713–11731.
- [128] Z. Wang, G. Sun, K. Zhou, L. Zhu, A parallel particle swarm optimization and enhanced sparrow search algorithm for unmanned aerial vehicle path planning, *Heliyon* 9 (4) (2023).
- [129] H.S. Radeaf, M.Z. Al-Faiz, Inverse kinematics optimization for humanoid robotic legs based on particle swarm optimization, in: 2023 15th International Conference on Developments in ESystems Engineering (DeSE), 2023, pp. 94–99.
- [130] B.K. Patle, S.-L. Chen, A. Singh, S.K. Kashyap, Optimal trajectory planning of the industrial robot using hybrid S-curve-PSO approach, *Robot. Intell. Autom.* 43 (2) (2023) 153–174.
- [131] S. Lin, J. Wang, B. Huang, X. Kong, H. Yang, Bio particle swarm optimization and reinforcement learning algorithm for path planning of automated guided vehicles in dynamic industrial environments, *Sci. Rep.* 15 (1) (2025) 463.
- [132] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Trans. Evol. Comput.* 1 (1) (1997) 53–66.
- [133] J.K.C. Revanna, N.Y.B. Al-Nakash, Impact of ACO intelligent vehicle real-time software in finding shortest path, *Softw. Impacts* 19 (2024) 100625.
- [134] S. Liu, L. Mao, J. Yu, Path planning based on ant colony algorithm and distributed local navigation for multi-robot systems, in: 2006 International Conference on Mechatronics and Automation, Luoyang, China, 2006, pp. 1733–1738.
- [135] O. Castillo, H. Neyoy, J. Soria, P. Melin, F. Valdez, A new approach for dynamic fuzzy logic parameter tuning in ant colony optimization and its application in fuzzy control of a mobile robot, *Appl. Soft Comput.* 28 (2015) 150–159.
- [136] P.B. Kumar, C. Sahu, D.R. Parhi, A hybridized regression-adaptive ant colony optimization approach for navigation of humanoids in a cluttered environment, *Appl. Soft Comput.* 68 (2018) 565–585.
- [137] H. Sun, C. Ji, Research on static path planning for mobile robot based on improved ant colony algorithm, in: 2024 5th International Conference on Management Science and Engineering Management (ICMSEM 2024), Atlantis Press, 2024, pp. 1071–1077.
- [138] Y. Yao, A.J. Wang, F.M. Shang, Dynamic obstacle avoidance path planning method for autonomous driving based on quantum ant colony algorithm, *Adv. Transp. Stud.* (2024).
- [139] J. Song, Y. Pu, X. Xu, Adaptive ant colony optimization with sub-population and fuzzy logic for 3D laser scanning path planning, *Sensors* 24 (4) (2024) 1098.
- [140] M. Brand, M. Masuda, N. Wehner, X.-H. Yu, Ant colony optimization algorithm for robot path planning, in: 2010 International Conference on Computer Design and Applications, vol. 3, 2010, V3-436.
- [141] X. Yuan, J. Wang, C. Tong, H. Wang, Integrated path planning for AUV communication efficiency and obstacle avoidance based on ant colony optimization and three-dimensional dynamic window algorithm, *Appl. Ocean Res.* 156 (2025) 104465.
- [142] L. Zhao, F. Li, D. Sun, Z. Zhao, An improved ant colony algorithm based on Q-learning for route planning of autonomous vehicle, *Int. J. Comput. Commun. Control.* 19 (3) (2024).
- [143] P. Wu, J. Huang, X. Li, B. Chen, Multi-missile routes co-planning method based on improved ant colony algorithm, in: 2023 42nd Chinese Control Conference, CCC, 2023, pp. 5750–5755.
- [144] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report TR06, Erciyes University, 2005, pp. 1–10.
- [145] M.A. Contreras-Cruz, V. Ayala-Ramirez, U.H. Hernandez-Belmonte, Mobile robot path planning using artificial bee colony and evolutionary programming, *Appl. Soft Comput.* 30 (2015) 319–328.
- [146] M. Saffari, M.J. Mahjoob, Bee colony algorithm for real-time optimal path planning of mobile robots, in: 2009 Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009, pp. 1–4.
- [147] Z. Han, M. Chen, H. Zhu, Q. Wu, Ground threat prediction-based path planning of unmanned autonomous helicopter using hybrid enhanced artificial bee colony algorithm, *Def. Technol.* 32 (2024) 1–22.
- [148] S. Kumar, D.R. Parhi, M.K. Muni, Path planning and obstacle avoidance of multi-robotic system in static and dynamic environments, *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* 237 (9) (2023) 1376–1390.
- [149] Q. Cui, P. Liu, H. Du, H. Wang, X. Ma, Improved multi-objective artificial bee colony algorithm-based path planning for mobile robots, *Front. Neurobot.* 17 (2023) 1196683.
- [150] Z. Han, M. Chen, H. Zhu, Q. Wu, Ground threat prediction-based path planning of unmanned autonomous helicopter using hybrid enhanced artificial bee colony algorithm, *Def. Technol.* 32 (2024) 1–22.
- [151] W. Fan, W. Luo, M. Wang, Y. Su, Y. Chen, K. Deng, A 3D path planning method based on improved artificial bee colony algorithm, in: 2022 37th Youth Academic Annual Conference of Chinese Association of Automation, YAC, 2022, pp. 561–566.
- [152] Y.A. Sarumaha, D.R. Firdaus, I. Moridu, The application of artificial bee colony algorithm to optimizing vehicle routes problem, *J. Inf. Syst. Technol. Eng.* 1 (1) (2023) 11–15.
- [153] S. Aslan, T. Oktay, Path planning of an unmanned combat aerial vehicle with an extended-treatment-approach-based immune plasma algorithm, *Aerospace* 10 (5) (2023) 487.
- [154] L. Ding, H. Wu, Y. Yao, et al., Chaotic artificial bee colony algorithm for system identification of a small-scale unmanned helicopter, *Int. J. Aerosp. Eng.* 2015 (2015).
- [155] M. Yasin, A. Khan, M.A. Qayyum, H.M.B. Yousuf, A. Mehfooz, D. Hunter, Biological control of locusts and grasshoppers: A review, *J. Orthoptera Res.* 33 (2) (2024) 289–304.
- [156] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: Theory and application, *Adv. Eng. Softw.* 105 (2017) 30–47.
- [157] B. Patel, V. Dubey, N. Sharma, Mobile robot navigational planning using grasshopper algorithm, in: 2023 OITS International Conference on Information Technology, OCIT, 2023, pp. 944–949.
- [158] A. Shareef, S. Al-Darraj, Grasshopper optimization algorithm based path planning for autonomous mobile robot, *Bull. Electr. Eng. Inf.* 11 (6) (2022) 3551–3561.
- [159] Z. Elmi, M.Ö. Efe, Online path planning of mobile robot using grasshopper algorithm in a dynamic and unknown environment, *J. Exp. Theor. Artif. Intell.* 33 (3) (2021) 467–485.
- [160] K.M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Syst. Mag.* 22 (3) (2002) 52–67.
- [161] L. dos Santos Coelho, C.A. Sierakowski, Bacteria colony approaches with variable velocity applied to path optimization of mobile robots, 2005.
- [162] J.A. Abdulsahab, D.J. Kadhim, Classical and heuristic approaches for mobile robot path planning: A survey, *Robotics* 12 (4) (2023) 93.
- [163] M.K. Muni, D.R. Parhi, P.B. Kumar, Improved motion planning of humanoid robots using bacterial foraging optimization, *Robotica* 39 (1) (2021) 123–136.
- [164] Y. Long, S. Liu, D. Qiu, C. Li, X. Guo, B. Shi, M.S. AbouOmar, Local path planning with multiple constraints for USV based on improved bacterial foraging optimization algorithm, *J. Mar. Sci. Eng.* 11 (3) (2023) 489.
- [165] A.N. Kasruddin Nasir, M.A. Ahmad, M.O. Tokhi, Hybrid spiral-bacterial foraging algorithm for a fuzzy control design of a flexible manipulator, *J. Low Freq. Noise Vib. Act. Control.* 41 (1) (2022) 340–358.
- [166] Z. Li, Z. Wang, Bacterial foraging algorithm with potential field guidance mechanism, 2021.
- [167] B. Tao, Y. He, X. Wang, Improvement of bacterial foraging algorithm for heading control of unmanned surface vessel, in: International Conference on Autonomous Unmanned Systems, Springer Nature Singapore, 2023, pp. 59–68.
- [168] Z. Wang, J. Peng, S. Ding, A bio-inspired trajectory planning method for robotic manipulators based on improved bacteria foraging optimization algorithm and tau theory, *Math. Biosci. Eng.* 19 (1) (2022) 643–662.
- [169] B. Tao, Y. He, X. Wang, Improvement of bacterial foraging algorithm for heading control of unmanned surface vessel, in: International Conference on Autonomous Unmanned Systems, Springer Nature Singapore, 2023, pp. 59–68.
- [170] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2010.
- [171] P. Bhanaria, Maneesha, P.K. Pandey, A multi-objective path-planning based on firefly algorithm for mobile robots, in: International Conference on Advances in IoT and Security with AI, Springer Nature Singapore, 2023, pp. 127–136.
- [172] O. Zedadra, A. Guerrieri, H. Seridi, A. Benzaid, G. Fortino, Inverse firefly-based search algorithms for multi-target search problem, *Big Data Cogn. Comput.* 8 (2) (2024) 18.
- [173] C. Shen, Automatic assembly system based on industrial robot using improved firefly optimization algorithm, in: 2024 International Conference on Data Science and Network Security, ICDNSNS, 2024, pp. 1–4.

- [174] B.K. Patle, D.R. Parhi, A. Jagadeesh, S.K. Kashyap, On firefly algorithm: Optimization and application in mobile robot navigation, *World J. Eng.* 14 (1) (2023) 65–76.
- [175] P.B. Kumar, D.R. Parhi, M. Sethy, Humanoid navigation: A firefly based approach, *Comput. Anim. Virtual Worlds* 32 (5) (2021).
- [176] K.R. Pratiba, S. Ridhanya, J. Ridhisha, P. Hemashree, Integrated Q-learning with firefly algorithm for transportation problems, *EAI Endorsed Trans. Energy Web* 11 (1) (2024).
- [177] T. Tian, Z. Liang, Y. Wei, Q. Luo, Y. Zhou, Hybrid whale optimization with a firefly algorithm for function optimization and mobile robot path planning, *Biomimetics* 9 (1) (2024).
- [178] A.K. Kashyap, A. Pandey, D.R. Parhi, S.S. Gour, Trajectory tracking of single and multiple humanoid robots in cluttered environment, *Mater. Today Proc.* 56 (2022) 650–654.
- [179] M. Zhang, Y. Zhao, Research on three-dimensional path planning of UAV-agv cooperative electric power patrol based on improved firefly algorithm, in: *Fourth International Conference on Mechanical, Electronics, and Electrical and Automation Control (METMS 2024)*, vol. 13163, SPIE, 2024, pp. 1994–1999.
- [180] N.H. Singh, A. Laishram, K. Thongam, Optimal path planning for mobile robot navigation using FA-TPM in cluttered dynamic environments, *Procedia Comput. Sci.* 218 (2023) 612–620.
- [181] M.M. Eusuff, K.E. Lansey, Optimization of water distribution network design using the shuffled frog leaping algorithm, *J. Water Resour. Plan. Manag.* 129 (3) (2003) 210–225.
- [182] J. Ni, X. Yin, J. Chen, X. Li, An improved shuffled frog leaping algorithm for robot path planning, in: *2014 10th International Conference on Natural Computation, ICNC, 2014*, pp. 545–549.
- [183] A. Hidalgo-Paniagua, M.A. Vega-Rodríguez, J. Ferruz, N. Pavón, MOSFLA-MRPP: Multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning, *Eng. Appl. Artif. Intell.* 44 (2015) 123–136.
- [184] K. Shubhasri, D.R. Parhi, Navigation based on adaptive shuffled frog-leaping algorithm for underwater mobile robot, in: *Intelligent Computing, Communication and Devices: Proceedings of ICCD 2014, Volume 1, 2015*, pp. 651–659.
- [185] Z. Chen, W. Zhang, Z. Li, W. Wu, B. Chen, T. Han, Research on autonomous inspection path of unmanned aerial vehicles based on shuffled frog leaping algorithm, in: *2024 6th International Conference on Electronic Engineering and Informatics, EEI, 2024*, pp. 255–260.
- [186] S.N. MohanRajan, A. Loganathan, P. Manoharan, Fuzzy shuffled frog leaping optimization-based enhanced convlstm for land use/land cover prediction, *Earth Sci. Inf.* 18 (2) (2025) 207.
- [187] H. Gao, L. Wei, Improved shuffled frog leading algorithm-based optimal trajectory path for robot control system, in: *2024 4th International Conference on Mobile Networks and Wireless Communications, ICMNWC, 2024*, pp. 1–5.
- [188] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Softw.* 69 (2014) 46–61.
- [189] U. Yüzgeç, Robot path planning using gray wolf optimizer, in: *International Conference on Advanced Technologies, Computer Engineering and Science, ICATCES'18, 2018*, pp. 69–74.
- [190] M. Panda, B. Das, B. Pati, Global path planning for multiple AUVs using GWO, *Arch. Contol. Sci.* 30 (1) (2020) 77–100.
- [191] L. Dong, X. Yuan, B. Yan, Y. Song, Q. Xu, X. Yang, An improved grey wolf optimization with multi-strategy ensemble for robot path planning, *Sensors* 22 (18) (2022) 6843.
- [192] R. Kumar, L. Singh, R. Tiwari, Path planning for the autonomous robots using modified grey wolf optimization approach, *J. Intell. Fuzzy Syst.* 40 (5) (2021) 9453–9470.
- [193] F. Gul, W. Rahiman, S.S.N. Alhady, A. Ali, I. Mir, A. Jalil, Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO-GWO optimization algorithm with evolutionary programming, *J. Ambient. Intell. Humaniz. Comput.* 12 (2021) 7873–7890.
- [194] F. Hason, S. Al-Darraj, A review: On bio-inspired optimization methods for path planning of mobile robot, *Iraqi J. Intell. Comput. Inf. (IJICI)* 2 (1) (2023) 1–10.
- [195] D. Polak, M. Woźniak, Red fox optimization algorithm, *Expert Syst. Appl.* 166 (2021) 114107.
- [196] M. Luneckas, et al., Hexapod robot gait switching for energy consumption and cost of transport management using heuristic algorithms, *Appl. Sci.* 11 (3) (2021) 1339.
- [197] R. Natarajan, G. Megharaj, A. Marchewka, P.B. Divakarachari, M.R. Hans, Energy and distance based multi-objective red fox optimization algorithm in wireless sensor network, *Sensors* 22 (10) (2022) 3761.
- [198] S. Al-Ansarry, S. Al-Darraj, D.G. Honi, An efficient path planning in uncertainty environments using dynamic grid-based and potential field methods, *Iraqi J. Electr. Electron. Eng.* 19 (2) (2023).
- [199] S. Al-Ansarry, S. Al-Darraj, A. Shareef, D.G. Honi, F. Fallucchi, Bi-directional adaptive probabilistic method with a triangular segmented interpolation for robot path planning in complex dynamic-environments, *IEEE Access* 11 (2023) 87747–87759.
- [200] B. Vaisi, A review of optimization models and applications in robotic manufacturing systems: Industry 4.0 and beyond, *Decis. Anal. J.* 2 (2022) 100031.
- [201] F.H. Ketafa, S. Al-Darraj, Path planning for an autonomous mobile robot using red fox optimization algorithm, in: *2022 International Conference on Data Science and Intelligent Computing, ICDSIC, 2022*, pp. 218–223.
- [202] H.A. Neamah, E. Donát, P. Korondi, Educational approach to trajectory prediction for dynamic obstacle avoidance using curve fitting methods, in: *2024 IEEE 11th International Conference on E-Learning in Industrial Electronics (ICELIE 2024)*, Chicago, IL, USA, 3–6, 2024, 2024, pp. 1–6.
- [203] L. Keczan, B. Örsi, K. Katona, R. Mikuska, H.A. Neamah, C. Csukonyi, P. Korondi, Technical limitations of organic human–robot interaction (o-HRI) for mobile robots moving amongst humans, in: *2024 IEEE 21st International Power Electronics and Motion Control Conference (PEMC 2024)*, Pilsen, Czech Republic, 30 Sep–3 2024, 2024.
- [204] H.A. Neamah, O.A. Mayorga Mayorga, Optimized TD3 algorithm for robust autonomous navigation in crowded and dynamic human-interaction environments, *Results Eng.* 24 (2024) 102874.
- [205] H.A. Neamah, E. Donát, P. Korondi, Optimizing autonomous navigation in unknown environments: A novel trap avoiding vector field histogram algorithm VFH+T, *Results Eng.* 23 (2024) 102625.
- [206] Y. Cao, N. Mohamad Nor, An improved dynamic window approach algorithm for dynamic obstacle avoidance in mobile robot formation, *Decis. Anal. J.* 11 (2024) 100471.
- [207] E.S. Low, P. Ong, C.Y. Low, An empirical evaluation of Q-learning in autonomous mobile robots in static and dynamic environments using simulation, *Decis. Anal. J.* 8 (2023) 100314.
- [208] R. Singh, J. Ren, X. Lin, A review of deep reinforcement learning algorithms for mobile robot path planning, *Vehicles* 5 (4) (2023) 1423–1451.
- [209] Z. Zhang, Y. Wang, H. Chen, L. Liu, Deep reinforcement learning for path planning of autonomous mobile robots, *Appl. Intell.* 55 (2025) 1–15.
- [210] Y. Zhu, X. Li, M. Zhang, J. Wang, Deep reinforcement learning of mobile robot navigation, *Sensors* 25 (12) (2025) 3847.
- [211] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, *Nature* 518 (7540) (2015) 529–533.
- [212] J.C.S. da Rosa Filho, R.S. Ferreira, A.L. Martinotto, Comparative analysis of PPO and DQN for UAV obstacle avoidance, *IFAC-PapersOnLine* 58 (1) (2025) 1020–1029.
- [213] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
- [214] M.R. Khan, S. Ahmed, T. Hassan, Advancing mobile robot navigation with DRL and heuristic approaches, *Neurocomputing* 601 (2025) 127708.
- [215] S. Yang, H. Liu, J. Chen, W. Zhang, Dual-attention proximal policy optimization for efficient autonomous navigation in narrow channels using deep reinforcement learning, *Ocean Eng.* 312 (2025) 118422.
- [216] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.
- [217] S.M. Nasti, P. Martinelli, R. Carli, Adaptive mapless mobile robot navigation using deep reinforcement learning based on TD3, *Front. Robot. AI* 12 (2025) 1625968.
- [218] R. Alharthi, M. Alshamrani, A. Alqahtani, Novel deep reinforcement learning based collision avoidance for mobile robots, *PLoS ONE* 20 (3) (2025) e0312559.
- [219] T. Zheng, Y. Wang, L. Chen, Deep Reinforcement Learning Based Coverage Path Planning using TD3, *Preprints.org*, 2025, 2025030300.
- [220] A. Khan, A. Ribeiro, V. Kumar, A.G. Francis, Graph neural networks for motion planning, 2020, arXiv preprint arXiv:2006.06248.
- [221] Q. Li, F. Gama, A. Ribeiro, A. Prorok, Graph neural networks for decentralized multi-robot path planning, in: *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., IROS, Las Vegas, NV, USA, 2020*, pp. 11833–11840.
- [222] Q. Li, W. Lin, Z. Liu, A. Prorok, Message-aware graph attention networks for large-scale multi-robot path planning, *IEEE Robot. Autom. Lett.* 6 (3) (2021) 5533–5540.
- [223] Z. Xiao, P. Li, C. Liu, H. Gao, X. Wang, MACNS: A generic graph neural network integrated deep reinforcement learning based multi-agent collaborative navigation system for dynamic trajectory planning, *Inf. Fusion* 103 (2024) 102128.
- [224] W. Lu, Q. Lin, L. Meng, C. Li, Collaborative navigation of multiple autonomous mobile robots via geometric graph neural network, *IEEE Trans. Ind. Electron.* (2025) (in press).
- [225] Z. Liu, Y. Zhai, J. Li, G. Wang, Y. Miao, Graph relational reinforcement learning for mobile robot navigation in large-scale crowded environments, *IEEE Trans. Neural Netw. Learn. Syst.* 34 (12) (2023) 9944–9958.
- [226] L. Zhao, X. Deng, R. Li, X. Gui, J. Sun, Graph-based robust localization of object-level map for mobile robotic navigation, *IEEE Trans. Instrum. Meas.* 72 (2023) 1–12.
- [227] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 30, Long Beach, CA, USA, 2017, pp. 5998–6008.

- [228] J. Liu, X. Chen, Y. Wang, M. Zhang, ViT-A\*: Legged robot path planning using vision transformer a\*, in: Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Detroit, MI, USA, 2023, pp. 1–8.
- [229] K. Lee, S. Park, J. Kim, H. Choi, Mission-conditioned path planning with transformer, *Electronics* 13 (13) (2024) 2437.
- [230] L. Chen, Y. Zhang, H. Wang, X. Liu, Transformer-based imitative reinforcement learning for multirobot path planning, *IEEE Trans. Neural Netw. Learn. Syst.* 34 (9) (2023) 6521–6534.
- [231] H. Luo, W. Chen, J. Zhang, Intelligent logistics management robot path planning algorithm integrating transformer and GCN network, 2025, arXiv preprint arXiv:2501.02749.
- [232] F. AlMahamid, K. Grolinger, Agile DQN: Adaptive deep recurrent attention for autonomous robot navigation, *Sci. Rep.* 15 (2025) 3287.
- [233] Y. Chen, H. Liu, J. Wang, Z. Zhang, A hierarchical reinforcement learning algorithm based on attention mechanism for UAV autonomous navigation, *IEEE Trans. Intell. Transp. Syst.* 23 (12) (2022) 25092–25106.
- [234] A.K. Kashyap, D.R. Parhi, Modified type-2 fuzzy controller for intercollision avoidance of single and multi-humanoid robots in complex terrains, *Intell. Serv. Robot.* 16 (1) (2023) 87–108.
- [235] T. Ren, H. Jebelli, Efficient 3D robotic mapping and navigation method in complex construction environments, *Comput.-Aided Civ. Infrastruct. Eng.* (2024).
- [236] F. Kamil, F.H. Gburi, M.A. Kadhon, B.A. Kalaf, Fuzzy logic-based control for intelligent vehicles: A survey, in: *AIP Conference Proceedings*, vol. 3092, (1) AIP Publishing, 2024.
- [237] V. Rajasekhar, A.G. Sreenatha, Fuzzy logic implementation of proportional navigation guidance, *Acta Astronaut.* 46 (1) (2000) 17–24.