E-ISSN: 2997-9382
American Journal of Technology Advancement
https://semantjournals.org/index.php/AJTA

| Research Article

Check for updates

Critical Evaluations of Path Planning Algorithms: A Comprehensive Review

Abbas Nadhim Kadhim, Muhammed Sabri Salim
Department of Electronics and Communication Engineering, Al-Nahrain University, Baghdad-Iraq

Annotation
Over the past few years, there has been a significant amount of interest in the research of path planning strategies, particularly in situations that remained static. This is because it is pertinent to the operations of mobile robots that are successful and autonomous. The reason for this is that it is relevant. The purpose of this study is to investigate and evaluate the algorithms and methodologies that are currently being utilized for the purpose of path planning in static settings. This study was specifically constructed for the purpose of achieving this goal. Graph-based algorithms and sampling-based algorithms are the two kinds that will be the focus of our subsequent discussion in this section. Both of these categories are subjected to a comprehensive investigation, with a particular emphasis placed on the underlying ideas, problems, and opportunities that lay beneath them. A special emphasis is being placed on the specific factors that need to be taken into consideration when applying graph-based and sampling-based tactics in static settings. In addition, various trajectories for future study are being evaluated.

Keywords: Path planning, Mobile robots, Review, Static environment, Sampling methods, Graph methods.

1. Introduction

Path planning is regarded as the primary and most important task for all robots in terms of autonomy for obstacle avoidance and navigation in their surroundings. It is a prerequisite for designing safe and collision-free paths in a crowded surroundings. The necessity of human participation could be greatly lowered by autonomous systems. The autonomous robot must thus be aware of all surrounding obstacles and determine a safe, practical, and best path depending on the environmental variables.[1],[2] Combining environmental data with the designed trip model creates a map. Hence, either partial or complete environmental data is required to create the path. [1], [3] Therefore, the design of an autonomous robot in systems meant to carry out entirely independent activities depends critically on path planning. The constructed path planning technique helps the mobile robot to identify the optimal path in the search space. The most important demand is that the produced road be safe and efficient under several environmental

conditions. [4] Path planning mostly aims to create a route free of obstacles so avoiding collisions with them. Path planning is essential to guarantee that the autonomous robot can achieve the goal along a path that respects several operational constraints, such power consumption and safety, including path planning. [5] Furthermore, by knowing the possible path, the mobile robot can start the control schema meant to lead it in the chosen direction. Path planning, then, simply creates a list of configurations based on environmental guidelines, beginning with the starting configuration and working through the goal configuration.[6] In static situations, path planning solves the problem of finding a workable path for mobile robots between a starting point and a destination. Guidance, surveillance, inspection, and assembly are among the several independent chores that depend on this procedure. [7], [8] The main goal is to find a motion path that complies with the particular limitations connected to the current mission, including the dimensions, kinematic characteristics, and any environmental impediment of the robot.[9] Path planning is a subset of motion planning, which also addresses actuator control management and collision avoidance techniques in addition to path determination. [10], [11] This work will give an analytical study of algorithms applied in stationary environments. Following an examination of several methods, their applications, and performance metrics relevant to these non-dynamic environments, one can assess the effectiveness and efficiency of these algorithms. Also include recommendations for more research and possible developments in this sector in this paper.

2. Static Environments

In mobile robotics, a stationary environment is one in which the physical components remain unmodified across time. This covers walls, furniture, and stationary monuments as well as elements. Using pre-existing maps and geographical data, robots negotiating a stationary environment may effectively pathfind and avoid obstacles. [8] Still, immovable things give mobile robots special challenges. These difficulties can limit mobility, create dead ends, or need careful navigation to cross safely. see Figure 1. For instance, a robot has to properly identify these specified obstacles and change its direction to prevent collisions. [11] Moreover, variations in sensor readings—such as those produced by surface textures or lighting conditions—may distort the robot's perspective of these obstacles even in case the surroundings are stationary. Strong sensor integration and effective algorithms are therefore essential for precisely spotting and overcoming stationary obstacles. [11],[12] Usually, the effective development of mobile robots in many applications rely on a knowledge of both the type of stationary obstacles and the traits of stationary surroundings.

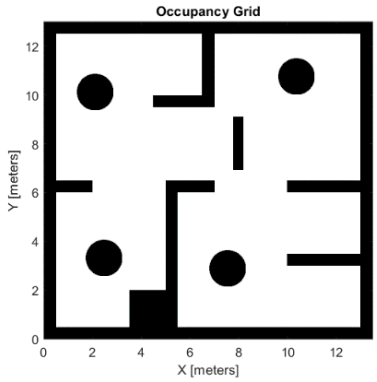


Figure (1): Static environment, the black area represent obstacles.

3. Taxonomy of Path Planning Algorithms

To be aware of the several subcategories in this field, the researcher has to examine the several classifications of static environment route planning algorithms. Graph-based and sampling-based techniques are two main forms of static environment path planning systems.

3.1. Graph-Based Algorithms

Graph search techniques govern how we examine connected data, from determining routes in transportation networks to discovering links in social media. Graph search techniques investigate the configuration space by examining a collection of configurations and their interactions. that build a configuration-space network composed of nodes representing robot configurations and edges connecting those nodes. The graph is then explored to determine the robot route leading to the destination. The graph format enables the convenient inclusion of feasibility knowledge and limitations across the robot's configuration space. In addition, the nonlinearity and complexity of the configuration space have no direct impact on the scalability of these methods. Furthermore, global graph search methods often provide coverage of all available free space with an acceptable computational cost as long as the original graph's structure is suitable.[1] Several graph search methods have been suggested for mobile robot path discovery in two and three dimensions, including Dijkstra, A*, and others.

3.1.1. Dijkstra's Algorithm

Dijkstra's algorithm, developed by Dutch computer scientist Edsger W. Dijkstra in 1956, is a foundational algorithm in computer science used for solving the single-source shortest path problem in graph theory. Originating in Dijkstra's work on network routing optimization, the method has clear uses in sectors including telecommunications and transportation. [13] Looking one by one through all the paths in a weighted graph, Dijkstra's algorithm finds the shortest path from a given beginning point to all the other points. [14],[15] It runs by keeping a priority queue of vertices starting with the source vertex at zero and setting all others to infinity. [16] The program chooses the vertex with the lowest tentative distance periodically, changes the distances to its adjacent vertices, and finally notes it as "visited." This method keeps on until all reachable vertices have been handled, therefore guaranteeing that the shortest path to every vertex is precisely found [17], [18]. The Dijkstra's algorithm flowchart is shown in Fig. 2.

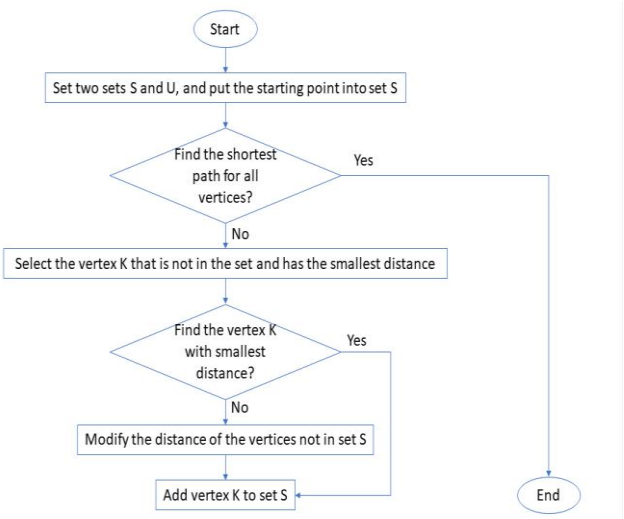


Figure (2): Flow chart of the Dijkstra's algorithm [14]

Dijkstra's method ensures an optimal solution for determining the shortest paths, so one of its main advantages in graphs with non-negative weights is its efficiency.[19] Working with several kinds of data structures, such as binary heaps and Fibonacci heaps, it takes between O(V2) and O(E + V log V) time to run, where V is the number of vertices and E is the number of edges [18]. The simplicity and methodical approach of the algorithm also help one to grasp and use it. Dijkstra's method has restrictions, though. Graphs with negative edge weights are not handled; should negative weights exist, the method may yield erroneous answers. [20] Furthermore, although it is ideal for determining the shortest path from a single source, it could not be as

effective when you have to identify the shortest paths from several sources [21],[22] in which case an algorithm such as A* could be more suited. Dijkstra's method is still a useful tool for path planning generally, particularly in cases of non-negative weights since it always finds the optimal solution and runs consistently.

Wijaya et al. 2024 [18] has made it feasible for service robots to negotiate well while avoiding obstacles by using of Dijkstra's algorithm for path planning optimization. The system comprised of a Lidar sensor-equipped service robot interfaced with a Jetson Nano driven by an ESP32 microcontroller, which was mapped and obstacle detected using Control and communication came from ROS. The study showed that Dijkstra's method is efficient in producing effective path planning solutions with an average movement speed of 0.23 meters per second and minimal positioning mistakes of 0.021 meters on the x-axis and 0.017 meters on the y-axis. This was achieved by letting the efficient negotiations around three separate hurdles go without collision.

Al shammrei et al. 2022 [23] provide an improved Dijkstra method to construct an optimal path planning for mobile robots (MRs), thereby permitting the shortest paths by modelling the surroundings of the robot as a directed graph. This effectively navigates around obstacles. Following a pre-defined path, the MR dynamically alters the graph by removing nodes upon obstacle identification and reevaluating the path in response using an ultrasonic sensor. Simulations on a hand-built MR navigation environment with 9 nodes, 19 edges and 2 obstacles were ran to demonstrate the efficiency and adaptability of our algorithm in real-time obstacle avoidance. The improved Dijkstra algorithm generated, based on the outcomes, an optimal path. The efficacy of the suggested path planning and obstacle avoidance algorithm is demonstrated in a case study including a mobile robot. The robot first discovered an optimal path of Q' = {1, 2, 3, 5, 8} with a distance of 460 cm; but, it adjusted its path by deleting node 2 when it came against obstacles at that node. The robot's meticulous halting at every intersection to evaluate prospective threats in surrounding ones made real-time path adjustments conceivable. Path planning and obstacle avoidance diagraphs are shown in Fig. 3.

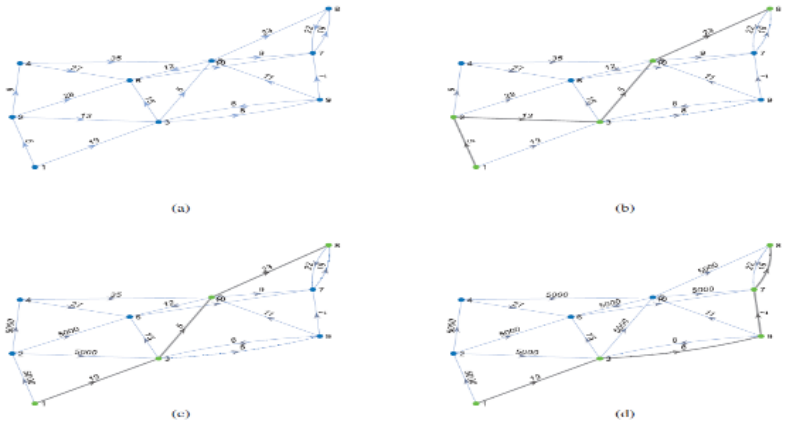


Figure (3): Illustration of the diagraphs of the path planning and obstacle avoidance implementation.[23]

ÇELİK et al. 2023 [21] Their aim was to improve Autonomous Mobile Robots' (AMRs') path planning capacity by means of a modification to the conventional Dijkstra's algorithm, widely applied for navigation tasks. Designing a low-cost AMR with a Raspberry Pi as the primary processing unit and integrating a LIDAR sensor via the Robot Operating System (ROS) for environment mapping and localization comprised the implementation. In both real-time and Gazebo simulations, the updated Dijkstra method was compared against the original one. Improving both navigation efficiency and decision-making time, results showed that the new method provided pathways roughly 1% to 3% shorter than those produced by the conventional Dijkstra algorithm. The navigation strategy created for an autonomous mobile robot (AMR)

utilizing the modified Dijkstra's algorithm is shown in Fig. 4. The figure shows a map where the expected robot trajectory is shown by a thin red line signifying the navigation path. A green dot marks the beginning of the motion; a purple dot marks the target position. The robot moves in a thick red curving line to indicate its direction toward the objective. Because the improved approach computes not only the distances to neighboring nodes but also to distant neighbors, therefore enabling a possibly shorter path and smoother direction shifts that show a more optimistic navigation than the original Dijkstra's algorithm. The image also shows minor variations from the intended path, however, which are ascribed to real-world component tolerances and non-ideal PID control tuning.

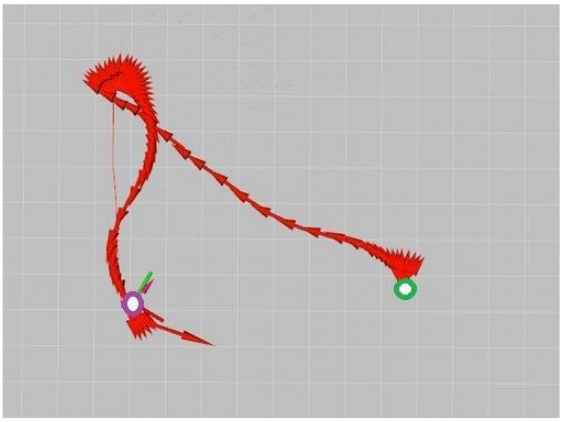


Figure (4): Modified Dijkstra's algorithm-based navigation plan and motion trail on the map.[21]

3.1.2. A* Algorithm

The A* algorithm is a fundamental pathfinding and graph traversal technique that originated in the 1960s, specifically developed by Peter Hart, Nils Nilsson, and Bertram Raphael. It was designed to find the shortest path from a starting node to a goal node while efficiently navigating a weighted graph, making it useful in various fields, including artificial intelligence, robotics, and computer games.[24] At its core, the A* algorithm operates by utilizing a best-first search strategy. It combines features from Dijkstra's algorithm and Greedy Best-First Search by maintaining a priority queue of nodes to explore, evaluated by a cost function denoted as

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

Here, g(n) represents the cost to reach the node from the start point, while h(n) is a heuristic estimate of the cost to reach the goal from that node.[25] This dual approach allows A* to efficiently zero in on the optimum path, ensuring that the search is guided toward the goal while accounting for the simplest route available. [26] The path planning program structure of traditional A* algorithm is shown in Fig. (5). One of the primary strengths of the A* algorithm is its efficiency, especially when a well-designed heuristic is employed. [28] A properly defined heuristic can significantly reduce the number of nodes explored, leading to quicker pathfinding compared to uninformed search methods.[29] Additionally, A* is complete and optimal, meaning it will always find the shortest path if one exists.[27] However, the A* algorithm does have some weaknesses. The accuracy and performance of A* heavily depend on the chosen heuristic. If the heuristic is poor, the search may become inefficient, exploring far more nodes than necessary.[30] Additionally, while A* performs well in many scenarios, its memory usage can be substantial, particularly in large search spaces, as it requires storing all explored nodes. This can lead to inefficiencies in environments where memory is constrained.[24], [31]

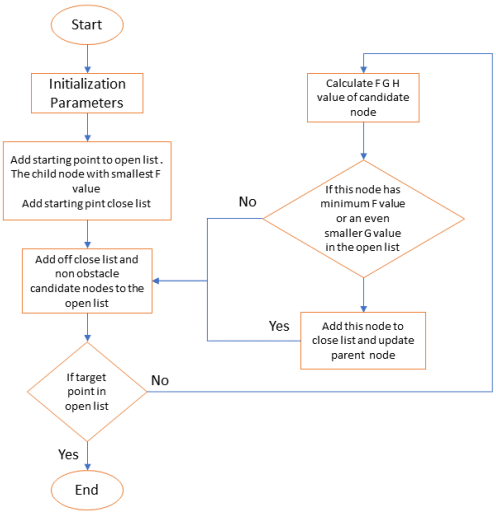


Figure (5): Traditional A* algorithm path planning block diagram.[27]

XiangRong et al. 2021 [32] suggest an improved A-star algorithm to make the traditional A-star algorithm better for planning robot paths in static environments by fixing the problem of how much memory it needs. The implementation involves introducing three new concepts: bidirectional search, a guideline, and a key point list, which optimize the algorithm's function and reduce the search area. Based on MATLAB simulations in an indoor 40x40 grid, the results show that the improved algorithm reduces the memory footprint by over 60%, the search area by up to 62.04% in different situations, and overall efficiency, getting better step length and search time than the original A-star algorithm.

Zhang et al. 2021 [24] offer an enhanced A* algorithm for mobile robot path planning that rectifies the deficiencies of the conventional A* method, including the generation of suboptimal paths characterized by excessive inflection points and insufficient smoothness. The execution entails segmenting the distance between neighboring nodes, employing a bidirectional optimization technique to eliminate superfluous nodes, and deploying a cubic spline function to refine the trajectory. The findings indicate substantial enhancements, as the refined A* algorithm reduces path length from 63.038 to 53.616 and decreases inflection points from 9 to 3, thereby improving the efficiency and smoothness of planned trajectories for mobile robots in intricate environments. Figure 6 depicts the trajectory produced by the enhanced A* algorithm.

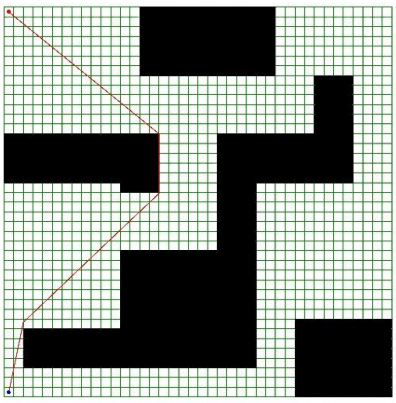


Figure (6): Improved A* algorithm path.[24]

Xin et al. 2019 [27] augment the path planning capabilities of mobile robots by refining the A* algorithm, with the objective of minimizing path length and enhancing smoothness for superior navigation. The implementation incorporates a threshold value into the open list of the A* algorithm to prioritize nodes, thereby considerably reducing search time. Additionally, it incorporates Floyd's technique to remove superfluous inflection points and implements a smoothing procedure to render the trajectory less abrupt, hence enhancing its suitability for robotic motion. The MATLAB simulation results indicate that the enhanced A* algorithm produces a more concise and smoother path than the conventional A* method, with a negligible increase in planning time. Figure 7 depicts the path simulation outcomes derived from the enhanced A* method, which integrates a threshold value in the open list and incorporates Floyd's technique alongside a smoothing procedure. This graphic illustrates the best path from the beginning point to the destination location, characterized by fewer inflection points and a more streamlined trajectory than the conventional A* algorithm. The route seems more straight and efficient, demonstrating the algorithm's capability to circumvent barriers successfully, thus optimizing both path length and the robot's movement dynamics. This enhanced graphic illustrates the algorithm's efficacy in producing a more viable and pragmatic route for mobile robots in practical applications.

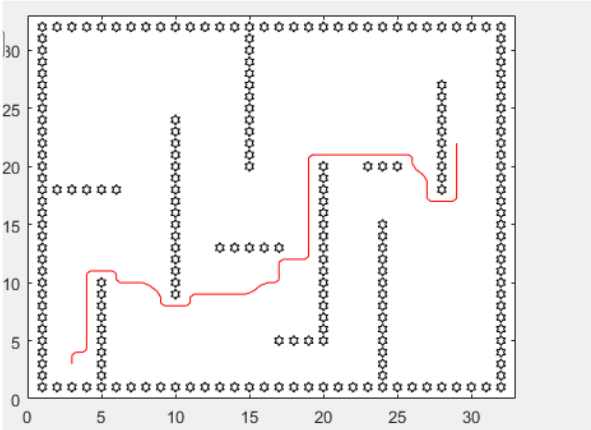


Figure (7): Improved A* algorithm path planning.[27]

3.2. Sampling-Based Algorithms

Sampling-based techniques are particularly significant in the fields of robotics and motion planning for navigating immobile environments. These strategies facilitate the construction of a representation of a specified space by generating random samples within that space. The Probabilistic Roadmap (PRM) is a recognized technique in which edges represent potential paths between nodes derived from samples. Rapidly-exploring Random Trees (RRT) represent a significant sampling-based technique. In RRT, branches extending to randomly selected locations in the space systematically construct a tree structure. RRT and PRM both rely on the assumption that the environment is static, ensuring that pathways and obstacles remain constant throughout the planning process. The stability of these algorithms allows them to optimize performance by focusing on the creation of efficient paths without the need for constant adjustment. [33]

3.2.1. Probabilistic Roadmaps (PRMs)

One prominent path planning technique that emerged from the nexus of computational geometry and robotics is the Probabilistic Roadmap (PRM) algorithm. The PRM algorithm was first developed in the 1990s to address the challenges of motion planning in high-dimensional environments. Its foundation is the establishment of a probabilistic framework for navigating complex environments, which is occasionally impractical for conventional deterministic methods. [34],[35] The learning phase and the inquiry phase are the two main stages in which the PRM

technique operates. The method chooses locations at random from the environment's empty configuration space during the learning phase. It then creates a roadmap that captures the spatial connectivity by connecting these samples along local paths free of obstacles. The algorithm uses the set roadmap to traverse the network of connections in search of a feasible path after being given a start point and an end point during the query phase. This approach makes it easier to navigate through intricate environmental formations. [36],[37] A typical PRM algorithm's learning process is depicted in Figure 8, and the PRM approach is shown in Figure 9. [38]

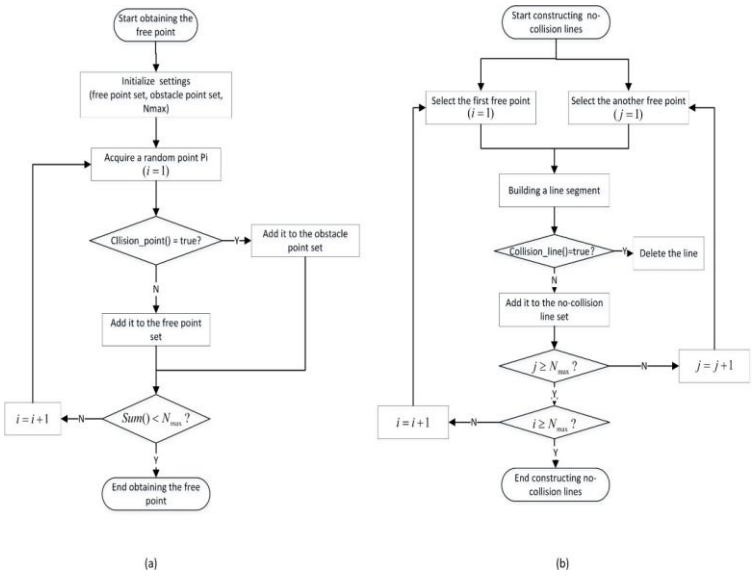


Figure (8): The learning process of typical PRM algorithm: (a) Obtaining free points; and (b) constructing collision-free line segments.[38]

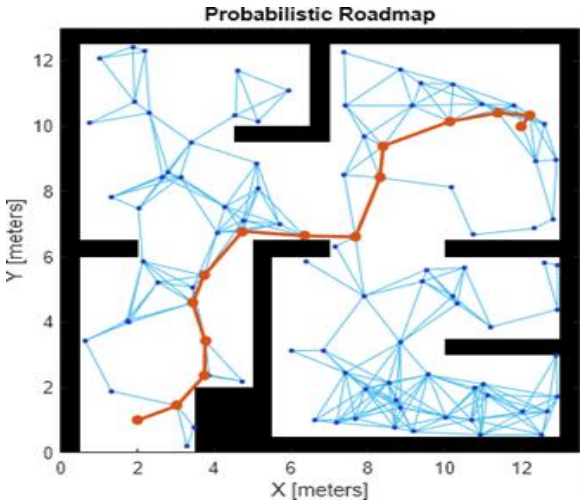


Figure (9): Demonstration of PRM algorithm

The PRM technique's capacity to manage high-dimensional spaces and its adaptability to various contexts through modifications in sample density and roadmap connections delineate its advantages. Moreover, it is particularly advantageous when the configuration space is excessively intricate for traditional methods, as it explores multiple directions through randomness. [38], [40] Nevertheless, the PRM approach possesses numerous shortcomings as well. The quality of the generated roadmap primarily hinges on the sampling technique; insufficient sampling may yield disjointed lines or suboptimal pathways. [41] Moreover, particularly in scenarios with fluctuating obstacles or during real-time operations, the initial plan formulation may incur significant computational expenses. [36] [42]

Li et al. 2022 [36] propose a revised form of the Probabilistic Roadmap Method (PRM) to improve route planning for mobile smart cars. The writers address various PRM problems, including its inefficiencies and inadequate information for choosing sampling sites, thereby that this The authors modified the PRM technique to create a road point distance criteria. They also applied a two-way incremental collision detection technique that reduced computing time and a pseudo-random sampling technique emphasizing the primary spatial axis. Furthermore, Bessel curves are used to smooth the produced pathways, so they are more suited for vehicle navigation. The improved PRM method performs better in a larger spectrum of planning scenarios than the conventional PRM algorithm, reduces pathways and collision detection calls, and makes roadmaps far faster. Tests on a ROS platform and MATLAB simulations backed these results. This raises the general standard and efficiency of the paths. Figure 10 shows the intended course produced by the adjusted Probabilistic Roadmap method. This figure shows the road map produced under the assumption that N, the number of sampling points, is set to 60 by use of the modified method.

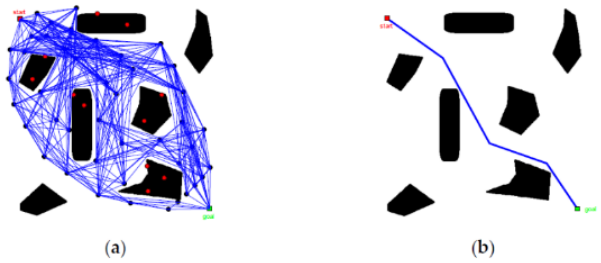


Figure (10): The modified PRM algorithm: (a) roadmap and (b) planned path.[36]

Ma et al. 2022 [42] proposing an improved bidirectional PRM algorithm that incorporates cubic spline interpolation for enhanced path smoothness. The implementation involves modifying the search strategy to alternately explore paths from both the starting and target nodes, effectively reducing unnecessary node connections and improving computational efficiency, while integrating cubic spline techniques to ensure smoother trajectory paths. Experimental results demonstrated that the bidirectional PRM significantly decreased average search times and achieved shorter path lengths compared to traditional PRM methods, with improvements of 24% in search efficiency and a 4% enhancement in path length, ultimately validating the superiority of the proposed approach in practical mobile robot navigation tasks. Fig. 11 illustrates the path generated by the cubic spline bidirectional PRM algorithm. It visually represents how the integration of cubic spline interpolation has transformed the originally segmented and potentially abrupt path into a smooth trajectory for the mobile robot.

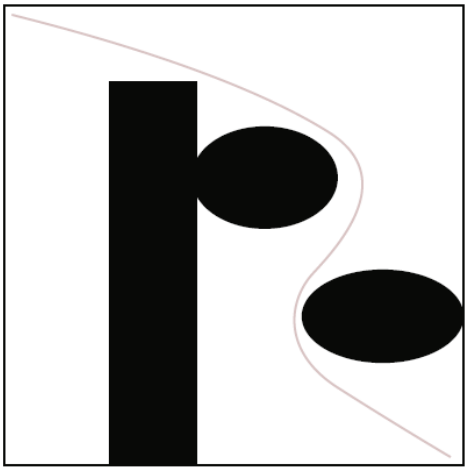


Figure (11): Cubic spline bidirectional PRM.[42]

3.2.2. Rapidly-exploring Random Trees (RRTs)

The Rapidly-exploring Random Tree (RRT) algorithm originated in the mid-1990s, developed by Steven M. LaValle as a solution for high-dimensional path planning issues.[43] Its primary purpose is to efficiently explore complex and high-dimensional spaces to find a feasible path from a start point to a goal point, particularly in robotics and artificial intelligence applications. [44] The core concept of RRT involves incrementally building a tree in the configuration space by randomly sampling points and extending the tree towards these points.[45] Initially, the tree starts at the initial configuration, and for each iteration, the algorithm randomly selects a position in the space. It then determines the nearest node in the current tree and creates a new node by moving a small step toward the random sample. This expansion continues until the generated tree reaches the target configuration, forming a continuous path.[46] Fig. (12) and Fig. (13) show the block diagram and the demonstration of the RRT algorithm. One of the significant strengths of RRT is its capability to efficiently handle complex environments with obstacles and high dimensionality, such as those found in robotic motion planning. Its ability to quickly explore the space makes it suitable for real-time applications.[47] Additionally, one can easily adapt and extend RRT to various variations, such as RRT*, which offer probabilistic completeness and optimality. [48], [49] Still, the RRT method has many flaws as well. Sometimes the created paths are unduly complicated and lack smoothness, which would not be appropriate for uses when a more exact path is needed.[50] In particular in complex or constrained environments where it can be difficult to plan because there aren't enough appropriate samples, randomness in the sampling process can also lead to inconsistent performance. Longer planning periods or failure to identify workable routes can follow from this. [46], [49], [51] Although RRT is a useful tool for path planning, its limits should be taken into account and possible improvement techniques should be considered to increase its relevance in different situations.

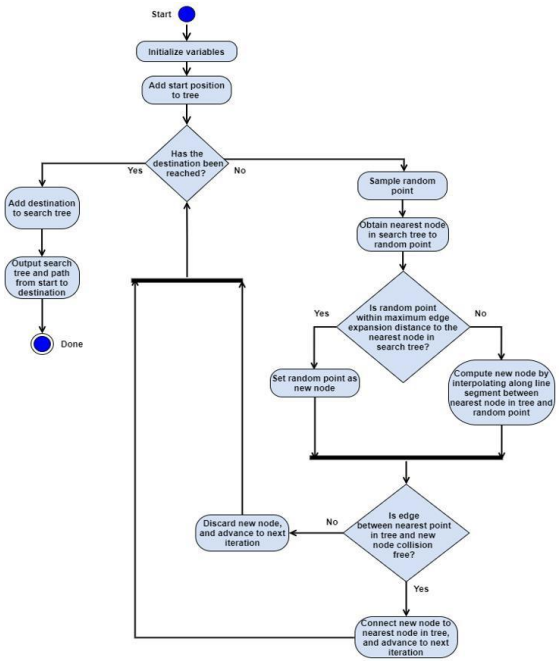


Figure (12): Block diagram of RRT algorithm.[33]

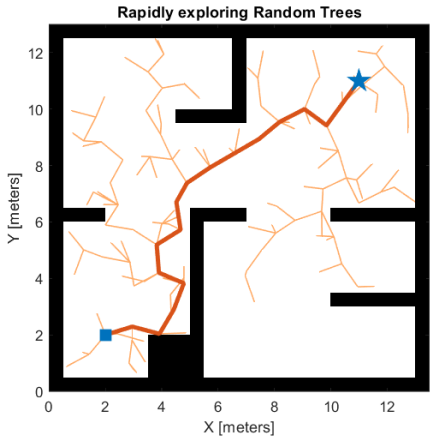


Figure (13): Demonstration of RRT algorithm

Chen et al. 2021 [52] suggests an enhanced RRT-Connect method (IRRT-Connect) to overrule the restrictions of conventional algorithms, including local minima trapping and longer search periods, so improving the efficiency of path planning for mobile robots. The implementation consists in creating a third node in the configuration space to support a quadtree expansion and in including a guiding mechanism that biassed the search for the goal point during node growth. Following its implementation on a ROS mobile robot, experimental results show that the IRRT-Connect algorithm substantially outperforms existing algorithms (RRT, RRT-Connect, and RRT*) in many environments, achieving improvements of up to 91% in the number of iterations, 88% in planning time, and 13% in path length for simpler environments, while maintaining efficiency in more complex settings, so validating its effectiveness in real-world applications. Fig. 14 shows a difficult environment marked by different obstacles the path produced by the enhanced RRT-Connect algorithm (IRRT-Connect). Emphasizing how effectively the algorithm negotiates around barriers, the figure illustrates a straight and smooth path from the beginning point to the goal. The path seems to be straight, proving the efficiency of the method in lowering the total path length while fast adjusting to challenges found along the road.

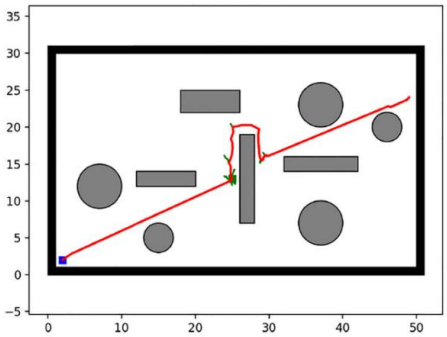
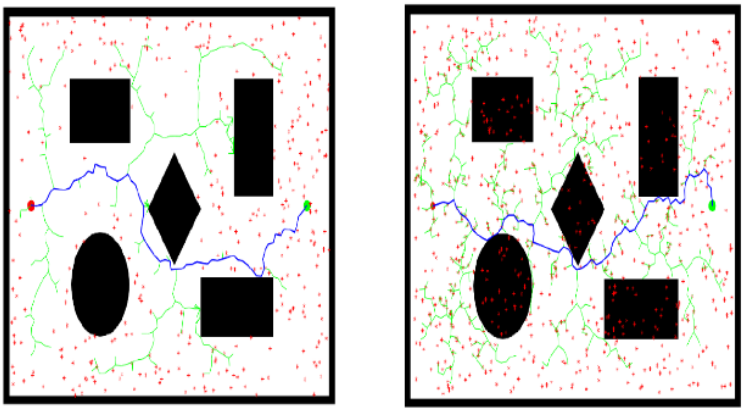


Figure (14): Path generated by the improved RRT-Connect algorithm.[52]

Wu et al. 2021 [46] Suggest an improved RRT algorithm called Fast-RRT algorithm, that aims to improve the speed and stability of path planning in motion planning tasks by addressing the limitations of the traditional RRT algorithm, which includes high search time variance and inefficiency in narrow passages. Its implementation consists of two key modules: the Improved RRT, which quickly finds an initial feasible path using a Fast-Sampling strategy that focuses on unexplored areas and a Random Steering method that enhances performance in narrow spaces; and the Fast-Optimal module, which fuses multiple paths to derive a near-optimal trajectory. The results highlight the efficacy of Fast-RRT, demonstrating that it operates 20 times faster than the RRT* algorithm with significantly lower search time variance. Fig. 15 illustrates the results of the Fast-RRT algorithm compared to the traditional RRT algorithm during a test scenario. In this

figure, the sampling states generated by the algorithms are represented by red points, while the green line depicts the random tree constructed by each algorithm. The blue line shows the feasible path resulting from the search. The distribution of sampling points indicates that the RRT algorithm spreads its points randomly throughout the state space, whereas the Fast-RRT algorithm's points are more strategically located, being sparse around the random tree and densely clustered in areas far from it. This strategic sampling approach is beneficial for guiding the random tree toward unexplored areas efficiently, thereby facilitating a quicker path to the target.



(a) **(b)**

Figure (15): Comparison of the operation results of (a) the improved RRT algorithm and (b) the RRT algorithm.[46]

4. Summary

Dijkstra's and A* are deterministic graph search algorithms focused on finding the shortest paths in defined graphs. In scenarios where heuristics can effectively guide the search, people generally prefer A* for pathfinding. PRM and RRT are both sampling-based methods that work well for robotics and high-dimensional spaces. RRT is especially good for environments that are complex or changeable, but it needs extra work to make sure the paths are smooth. While PRM shines in answering many questions against a fixed road map, it could find difficulties with dynamic changes. Table 1 summarizes the several techniques covered in this paper.

Table (1): Summary of the algorithms in static environments.

Algorithm	Dijkstra's Algorithm	A* Algorithm	Probabilistic Roadmaps (PRM)	Rapidly-exploring Random Trees (RRT)
Strengths	guarantees an optimal solution for finding the shortest paths ability to work well with various data structures easy to implement and understand.	complete and optimal Adapts well to different heuristic functions based on specific environments.	ability to handle high-dimensional spaces and its adaptability to various environments particularly beneficial in situations where the configuration space is too complicated for traditional methods	Handles complex environments and high-dimensional spaces effectively. Capable of real-time exploration and can be adapted with variations like RRT* for optimal solutions.
Limitations	Inefficient for	The accuracy and	quality of the generated	Paths can be

	finding shortest paths from multiple sources. Does not handle graphs with negative edge weights	performance heavily depend on the chosen heuristic memory usage can be substantial	roadmap heavily depends on the sampling strategy can be computationally expensive	jagged or suboptimal; subsequently, refinement strategies may be needed Performance may deteriorate in tightly constrained environments without proper adjustments.
Optimality	Yes	Yes	No (depends on roadmap quality)	No (but can be adapted to RRT* for optimality)

5. Conclusion

This study emphasizes the critical need of path planning in the realm of autonomous robotics by giving a thorough summary of many techniques and their applications in navigation and obstacle avoidance. According to the study, good path design guarantees safety in challenging environments and boosts robot navigation's performance. Among the algorithms—Dijkstra, A*, and Rapidly-exploring Random Trees (RRT)—there are advantages and drawbacks. This emphasizes the need of continually adjusting to match various circumstances. Including creative ideas and improving current techniques will be vital as the discipline develops to increase robot autonomy, lower human intervention, and maximize performance in practical uses. Future study should improve current models and generate hybrid algorithms to handle these problems. At last, this will provide more sophisticated and practical self-driving systems. The results of the study greatly progress the present conversation on robotic navigation. They underline how path planning helps to provide consistent and efficient autonomous operations.

REFERENCES

1. L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, “Path planning techniques for mobile robots: Review and prospect,” Oct. 01, 2023, Elsevier Ltd. doi: 10.1016/j.eswa.2023.120254.
2. R. Raj and A. Kos, “A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives,” Jul. 01, 2022, MDPI. doi: 10.3390/app12146951.
3. J. R. Sánchez-Ibáñez, C. J. Pérez-Del-pulgar, and A. García-Cerezo, “Path planning for autonomous mobile robots: A review,” Dec. 01, 2021, MDPI. doi: 10.3390/s21237898.
4. B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, “A review: On path planning strategies for navigation of mobile robot,” Aug. 01, 2019, China Ordnance Society. doi: 10.1016/j.dt.2019.04.011.
5. L. Yang et al., “Path Planning Technique for Mobile Robots: A Review,” Oct. 01, 2023, Multidisciplinary Digital Publishing Institute (MDPI). doi: 10.3390/machines11100980.
6. C. Zhou, B. Huang, and P. Fränti, “A review of motion planning algorithms for intelligent robots,” Feb. 01, 2022, Springer. doi: 10.1007/s10845-021-01867-z.
7. B. K. Jogeshwar and K. Lochan, “Algorithms for Path Planning on Mobile Robots,” in IFAC-PapersOnLine, Elsevier B.V., 2022, pp. 94–100. doi: 10.1016/j.ifacol.2022.04.016.

8. J. R. Sánchez-Ibáñez, C. J. Pérez-Del-pulgar, and A. García-Cerezo, “Path planning for autonomous mobile robots: A review,” Dec. 01, 2021, MDPI. doi: 10.3390/s21237898.
9. S. Lin, A. Liu, J. Wang, and X. Kong, “A Review of Path-Planning Approaches for Multiple Mobile Robots,” Sep. 01, 2022, MDPI. doi: 10.3390/machines10090773.
10. F. Gul, W. Rahiman, and S. S. Nazli Alhady, “A comprehensive study for robot navigation techniques,” Jan. 01, 2019, Cogent OA. doi: 10.1080/23311916.2019.1632046.
11. K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, “A Survey of Path Planning Algorithms for Mobile Robots,” Vehicles, vol. 3, no. 3, pp. 448–468, Sep. 2021, doi: 10.3390/vehicles3030027.
12. B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, “A review: On path planning strategies for navigation of mobile robot,” Aug. 01, 2019, China Ordnance Society. doi: 10.1016/j.dt.2019.04.011.
13. X. Zhou, J. Yan, M. Yan, K. Mao, R. Yang, and W. Liu, “Path Planning of Rail-Mounted Logistics Robots Based on the Improved Dijkstra Algorithm,” Applied Sciences (Switzerland), vol. 13, no. 17, Sep. 2023, doi: 10.3390/app13179955.
14. L. S. Liu et al., “Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach,” Wirel Commun Mob Comput, vol. 2021, 2021, doi: 10.1155/2021/8881684.
15. X. Li, “Path planning of intelligent mobile robot based on Dijkstra algorithm,” in Journal of Physics: Conference Series, IOP Publishing Ltd, Dec. 2021. doi: 10.1088/1742-6596/2083/4/042034.
16. R. Szczepanski and T. Tarczewski, “Global path planning for mobile robot based on Artificial Bee Colony and Dijkstra’s algorithms,” in Proceedings - 2021 IEEE 19th International Power Electronics and Motion Control Conference, PEMC 2021, Institute of Electrical and Electronics Engineers Inc., Apr. 2021, pp. 724–730. doi: 10.1109/PEMC48073.2021.9432570.
17. A. Alyasin, E. I. Abbas, and S. D. Hasan, “An Efficient Optimal Path Finding for Mobile Robot Based on Dijkstra Method,” in 4th Scientific International Conference Najaf, SICN 2019, Institute of Electrical and Electronics Engineers Inc., Apr. 2019, pp. 11–14. doi: 10.1109/SICN47020.2019.9019345.
18. R. S. Wijaya, A. Mahendra, S. Prayoga, and A. Wibisana, “Path Planning Application using Dijkstra Algorithm on Service Robot,” 2024, pp. 262–271. doi: 10.2991/978-94-6463-620-8_20.
19. D. Rachmawati and L. Gustin, “Analysis of Dijkstra’s Algorithm and A∗ Algorithm in Shortest Path Problem,” in Journal of Physics: Conference Series, Institute of Physics Publishing, Jul. 2020. doi: 10.1088/1742-6596/1566/1/012061.
20. M. Luo, X. Hou, and J. Yang, “Surface Optimal Path Planning Using an Extended Dijkstra Algorithm,” IEEE Access, vol. 8, pp. 147827–147838, 2020, doi: 10.1109/ACCESS.2020.3015976.
21. O. M. ÇELİK and M. KÖSEOĞLU, “A Modified Dijkstra Algorithm for ROS Based Autonomous Mobile Robots,” Journal of Advanced Research in Natural and Applied Sciences, vol. 9, no. 1, pp. 205–217, Mar. 2023, doi: 10.28979/jarnas.1119957.
22. J. Pak, J. Kim, Y. Park, and H. Il Son, “Field Evaluation of Path-Planning Algorithms for Autonomous Mobile Robot in Smart Farms,” IEEE Access, vol. 10, pp. 60253–60266, 2022, doi: 10.1109/ACCESS.2022.3181131.

23. S. Alshammrei, S. Boubaker, and L. Kolsi, “Improved Dijkstra Algorithm for Mobile Robot Path Planning and Obstacle Avoidance,” Computers, Materials and Continua, vol. 72, no. 3, pp. 5939–5954, 2022, doi: 10.32604/cmc.2022.028165.
24. L. Zhang and Y. Li, “Mobile Robot Path Planning Algorithm Based on Improved A Star,” in Journal of Physics: Conference Series, IOP Publishing Ltd, Apr. 2021. doi: 10.1088/1742-6596/1848/1/012013.
25. G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, “Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment,” IEEE Access, vol. 9, pp. 59196–59210, 2021, doi: 10.1109/ACCESS.2021.3070054.
26. H. Liu and Y. Zhang, “ASL-DWA: An Improved A-Star Algorithm for Indoor Cleaning Robots,” IEEE Access, vol. 10, pp. 99498–99515, 2022, doi: 10.1109/ACCESS.2022.3206356.
27. W. Xin, L. Wanlin, F. Chao, and H. Likai, “Path Planning Research Based on An Improved A∗ Algorithmfor Mobile Robot,” in IOP Conference Series: Materials Science and Engineering, Institute of Physics Publishing, Aug. 2019. doi: 10.1088/1757-899X/569/5/052044.
28. F. Duchon et al., “Path planning with modified A star algorithm for a mobile robot,” in Procedia Engineering, Elsevier Ltd, 2014, pp. 59–69. doi: 10.1016/j.proeng.2014.12.098.
29. D. Xiang, H. Lin, J. Ouyang, and D. Huang, “Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot,” Sci Rep, vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-17684-0.
30. X. Dai, S. Long, Z. Zhang, and D. Gong, “Mobile robot path planning based on ant colony algorithm with a∗ heuristic method,” Front Neurorobot, vol. 13, Apr. 2019, doi: 10.3389/fnbot.2019.00015.
31. O. O. Martins, A. A. Adekunle, O. M. Olaniyan, and B. O. Bolaji, “An Improved multi-objective a-star algorithm for path planning in a large workspace: Design, Implementation, and Evaluation,” Sci Afr, vol. 15, Mar. 2022, doi: 10.1016/j.sciaf.2021.e01068.
32. T. XiangRong, Z. Yukun, and J. XinXin, “Improved a-star algorithm for robot path planning in static environment,” in Journal of Physics: Conference Series, IOP Publishing Ltd, Feb. 2021. doi: 10.1088/1742-6596/1792/1/012067.
33. J. Jermyn, “A Comparison of the Effectiveness of the RRT, PRM, and Novel Hybrid RRT-PRM Path Planners,” Int J Res Appl Sci Eng Technol, vol. 9, no. 12, pp. 600–611, Dec. 2021, doi: 10.22214/ijraset.2021.39297.
34. S. Alarabi, C. Luo, and M. Santora, “A PRM Approach to Path Planning with Obstacle Avoidance of an Autonomous Robot,” in 2022 8th International Conference on Automation, Robotics and Applications, ICARA 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 76–80. doi: 10.1109/ICARA55094.2022.9738559.
35. G. Song, S. Thomas, and N. M. Amato, “A general framework for PRM motion planning,” in 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), 2003, pp. 4445–4450.
36. Q. Li, Y. Xu, S. Bu, and J. Yang, “Smart Vehicle Path Planning Based on Modified PRM Algorithm,” Sensors, vol. 22, no. 17, Sep. 2022, doi: 10.3390/s22176581.
37. L. Changan, C. Jingang, and L. Chunyang, “Path Planning for Mobile Robot Based on an Improved Probabilistic Roadmap Method *,” 2009.

38. L. Qiao, X. Luo, and Q. Luo, “An Optimized Probabilistic Roadmap Algorithm for Path Planning of Mobile Robots in Complex Environments with Narrow Channels,” Sensors, vol. 22, no. 22, Nov. 2022, doi: 10.3390/s22228983.
39. L. Jaillet and T. Siméon, “A PRM-based motion planner for dynamically changing environments,” in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2004, pp. 1606–1611. doi: 10.1109/iros.2004.1389625.
40. A. MSaeed, K. Shaaban Rijab, A. M. Saeed, and K. S. Rijab, “Enhancing Performance of Path Planning PRM Algorithm for Automated Boat Using PID Controller,” Journal of Global Scientific Research in Electrical and Electronic Engineering, vol. 9, pp. 3678–3689, 2024, doi: 10.5281/jgsr.2024.14039138.
41. R. Bohlin and L. E. Kavraki, “Path planning using Lazy PRM,” Proceedings-IEEE International Conference on Robotics and Automation, vol. 1, pp. 521–528, 2000, doi: 10.1109/ROBOT.2000.844107.
42. X. Ma, R. Gong, Y. Tan, H. Mei, and C. Li, “Path Planning of Mobile Robot Based on Improved PRM Based on Cubic Spline,” Wirel Commun Mob Comput, vol. 2022, 2022, doi: 10.1155/2022/1632698.
43. L. Garrote, C. Premebida, M. Silva, and U. Nunes, “An RRT-based navigation approach for mobile robots and automated vehicles,” in Proceedings - 2014 12th IEEE International Conference on Industrial Informatics, INDIN 2014, Institute of Electrical and Electronics Engineers Inc., Nov. 2014, pp. 326–331. doi: 10.1109/INDIN.2014.6945533.
44. J. G. Kang, D. W. Lim, Y. S. Choi, W. J. Jang, and J. W. Jung, “Improved RRT-connect algorithm based on triangular inequality for robot path planning,” Sensors (Switzerland), vol. 21, no. 2, pp. 1–34, Jan. 2021, doi: 10.3390/s21020333.
45. I. Noreen, A. Khan, and Z. Habib, “A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms,” 2016.
46. Z. Wu, Z. Meng, W. Zhao, and Z. Wu, “Fast-RRT: A RRT-based optimal path finding method,” Applied Sciences (Switzerland), vol. 11, no. 24, Dec. 2021, doi: 10.3390/app112411777.
47. B. Liu and C. Liu, “Path planning of mobile robots based on improved RRT algorithm,” in Journal of Physics: Conference Series, IOP Publishing Ltd, Apr. 2022. doi: 10.1088/1742-6596/2216/1/012020.
48. R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy, and I. Ali, “Informed RRT∗-Connect: An Asymptotically Optimal Single-Query Path Planning Method,” IEEE Access, vol. 8, pp. 19842–19852, 2020, doi: 10.1109/ACCESS.2020.2969316.
49. P. Xin, X. Wang, X. Liu, Y. Wang, Z. Zhai, and X. Ma, “Improved Bidirectional RRT* Algorithm for Robot Path Planning,” Sensors, vol. 23, no. 2, Jan. 2023, doi: 10.3390/s23021041.
50. K. Hao, Y. Yang, Z. Li, Y. Liu, and X. Zhao, “CERRT: A Mobile Robot Path Planning Algorithm Based on RRT in Complex Environments,” Applied Sciences (Switzerland), vol. 13, no. 17, Sep. 2023, doi: 10.3390/app13179666.
51. Feraco, S.; Luciani, S.; Bonfitto, and A.; Amati, “A local trajectory planning and control method for autonomous vehicles based on the RRT algorithm,” 2021.
52. J. Chen, Y. Zhao, and X. Xu, “Improved RRT-Connect Based Path Planning Algorithm for Mobile Robots,” IEEE Access, vol. 9, pp. 145988–145999, 2021, doi: 10.1109/ACCESS.2021.3123622.