INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2024, VOL. 70, NO. 1, PP. 13–22
Manuscript received October 27, 2023; revised March, 2024. doi: 10.24425/ijet.2023.147709

Evaluation of popular path planning algorithms

Mehmet Kara

Abstract—**The navigation of mobile robots is a key element of autonomous systems, which allows robots to move effectively and securely in changing environments with greater autonomy and precision. This study aims to provide researchers with a comprehensive guide for selecting the best path-planning methods for their particular projects. We evaluate some popular algorithms that are regularly used in mobile robot navigation, in order to demonstrate their specifications and determine where they are most effective. For example, one algorithm is used to model the problem as a standard graph, and another algorithm is found to be the most suitable for highly dynamic and highly dimensional environments, due to its robust path-planning capabilities and efficient route construction. We also filter high-performance algorithms in terms of computational complexity, accuracy, and robustness. In conclusion, this study provides valuable information on its individual strengths and weaknesses, helping robotics and engineers make informed decisions when selecting the most appropriate algorithm for their specific applications.**

Keywords—**mobile robot; navigation; algorithms; evaluation; robotics**

I. Introduction

MOBILE robot navigation is a rapidly expanding field that has a wide range of uses, such as automation in manufacturing and logistics, search and rescue operations, and exploration of dangerous environments [1]. The goal is to find the best way to help robots move around in new environments and complete their tasks successfully [2]. To achieve robot navigation, many different methods have been developed over the years. Some of these methods are older and more traditional, while others use new techniques like machine learning.

Conventional approaches to mobile robot navigation encompass fuzzy logic-based controllers and potential field methods [3], [4]. While these methods find widespread application due to their simplicity, they often have limitations. Challenges such as grappling with intricate scenarios, as well as grappling with susceptibility to sensor noise and uncertainty, have emerged as recurring issues [5].

Recent advances in machine learning and artificial intelligence have yielded elaborate and robust algorithms tailored for mobile robot navigation. Among these innovations are methodologies such as neuroevolution, imitation learning, and deep reinforcement learning [6]. These cutting-edge approaches demonstrate promising capabilities in addressing intricate and dynamic scenarios. Nevertheless, they often involve substantial prerequisites of extensive training data and computational resources.

Mehmet Kara is with AGH University of Science and Technology, Krakow, Poland (e-mail: kara@agh.edu.pl, orcidID 0000-0001-9045-2614).

Central to the development and deployment of mobile robot navigation algorithms is the critical evaluation phase [7]. The effectiveness of navigation algorithms is gauged through various metrics that include success rate, navigation time, path length, and collision frequency. These evaluative benchmarks stand as recurrent benchmarks, used effectively to dissect the efficacies and limitations of different algorithms [8].

Algorithms can exhibit varying superiority depending on the context [9]. For example, while algorithm A might yield enhanced solutions over algorithm B, the trade-off lies within the temporal dimension, where algorithm A demands a longer processing duration. Here, the system's overarching priority necessitates scrutiny to effectuate the optimal algorithmic choice. Should time emerge as a critical factor, resulting in swift outcomes, the discerning choice would favor algorithm A.

This research seeks to identify the most effective algorithms, exploring their advantages and drawbacks. To compare them, the algorithms were tested under the same conditions, and their performance was evaluated in terms of how close they got to the optimal path and how efficient they were computationally. The following section provides an in-depth review of the literature on the most common path-planning algorithms. It then goes on to discuss the selection of algorithms, the methodology used, and the tests conducted, culminating in a comprehensive assessment of the results.

II. Background of Path Planning

The navigation of mobile robots is a key element in robotics and autonomous systems [10]. It involves the planning and control of the robot's motion in order to reach a desired goal in its environment. The aim of navigation is to allow the robot to move safely, efficiently and autonomously, while avoiding obstacles and reaching its destination.

The utilization of mobile robots has grown significantly in recent years, with applications in a variety of domains. These include autonomous vehicles, UAVs, warehouse robotics, agricultural robots, and search and rescue operations [11]. The development of algorithms, sensors, and computing power has enabled robots to navigate complex and ever-changing environments with greater autonomy and accuracy.

Path planning related to mobile robot navigation is the process of finding an optimal or feasible path for a mobile robot to move from its initial position to a desired goal position while avoiding obstacles or navigating through complex environments [12].

The increasing number of path-planning algorithms requires a systematic evaluation [13]. The performance of these algorithms can vary greatly depending on the complexity of the environment, the task at hand, and other contextual elements. Therefore, it is essential to evaluate algorithms in a variety of scenarios to determine their adaptability and effectiveness. This evaluation helps researchers, engineers, and practitioners make informed decisions when selecting algorithms for real-world applications.

Choosing an algorithm is not a one-size-fits-all situation. Depending on the robotic task, different priorities must be taken into account, such as reducing the path length, optimizing the computation time, or making sure the algorithm is reliable under uncertain conditions [14]. An algorithm that works well in one area may not be as successful in another. Therefore, it is essential to match the algorithm to the task objectives. This highlights the importance of having a thorough understanding of the advantages and disadvantages of each algorithm, making the evaluation of algorithms a key factor in making informed decisions.

The path planning journey for mobile robots conventionally involves two fundamental steps: creating an environment representation and devising a path that either optimizes a cost function or fulfills specific constraints. This intricate process is navigated through the application of a diverse array of algorithms. In pursuit of understanding this dynamic landscape, this study focuses on six prominent algorithms that have garnered substantial utilization and recognition. The six algorithms chosen encompass:

1) A Star Algorithm
2) Genetic Algorithm (GA)
3) Fuzzy Logic-Based Algorithm
4) Artificial Potential Field (APF)
5) Probabilistic Road Map (PRM)
6) Rapidly Discovering Random Trees (RRT)

The following section of the article explains the reasons for choosing these algorithms.

III. SELECTION OF ALGORITHMS

The navigation of mobile robots is a key element of autonomous systems, which allows robots to move effectively and securely in changing environments. To ensure optimal performance and successful path planning, it is essential to select the right algorithm. In this article, we will evaluate some popular algorithms that are regularly used in mobile robot navigation. Our aim is to objectively compare their advantages and disadvantages, aiding robotics and engineers in making informed decisions when selecting the most suitable algorithm for their particular applications.

We looked at the performance of path-planning algorithms in terms of computational complexity, precision, and robustness. We employed a methodical approach when selecting the path-planning algorithms for our paper. We began by specifying the requirements of the path planning problem, including the complexity of the environment, the presence of obstacles, and the dimensionality of the configuration space. We then evaluated the suitability of each algorithm to meet these requirements. Finally, we filter high-performance algorithms in terms of computational complexity, accuracy, and robustness.

We objectively evaluated the performance of the algorithms by selecting relevant performance metrics such as path length, computation time, success rate, and scalability. These metrics were essential to quantify the efficiency and effectiveness of each algorithm in our particular situation.

The efficiency and speed of our application were of the utmost importance, so we carefully studied the computational complexity of each algorithm, taking into account our need for real-time operations and the computational resources available. To guarantee reliability, we tested the algorithms in difficult situations, such as narrow passages, complex obstacles, and static environments. This assessment allowed us to determine how well each algorithm adapted and worked under difficult conditions.

We also looked at the learning curve associated with the use and adjustment of each algorithm. To facilitate quick development and deployment, we favored algorithms with extensive libraries and simpler implementations. By taking into account these restrictions, we ensured that the algorithms chosen would be effective within the boundaries of our robotic system.

In conclusion, after a thorough evaluation and examination, we made informed decisions about our path-planning algorithms. For the situation where the shortest route was essential, we chose the A star algorithm because of its effectiveness and capacity to guarantee the best outcome with an appropriate heuristic. For optimization tasks with multiple objectives, the GA provided superior performance in finding near-optimal solutions and was thus selected. In uncertain and imprecise environments, we opted for the Fuzzy Logic-Based Algorithm, taking advantage of its decision-making abilities based on linguistic rules and membership functions. For real-time navigation in static environments, the APF algorithm was our preference due to its straightforwardness and computational efficiency. The PRM algorithm was found to be the most suitable for highly dynamic and highly dimensional environments, due to its robust path planning capabilities and efficient route construction. The RRT algorithm was also chosen for its ability to rapidly explore and adapt in real time to high-dimensional spaces, making it ideal for scenarios that require quick responses. Using this comprehensive methodology, we were able to select the best path-planning algorithms for our robotic navigation application, guaranteeing efficient and intelligent autonomous motion in a variety of complex environments.

IV. Experimental Setup and Methodology

We evaluated each algorithm against established criteria in order to better understand its individual benefits and drawbacks for the navigation of mobile robots. Every navigation algorithm has its own set of features and strengths. Although we looked at many criteria when we filter mentioned six algorithms among others, when it comes to evaluating and

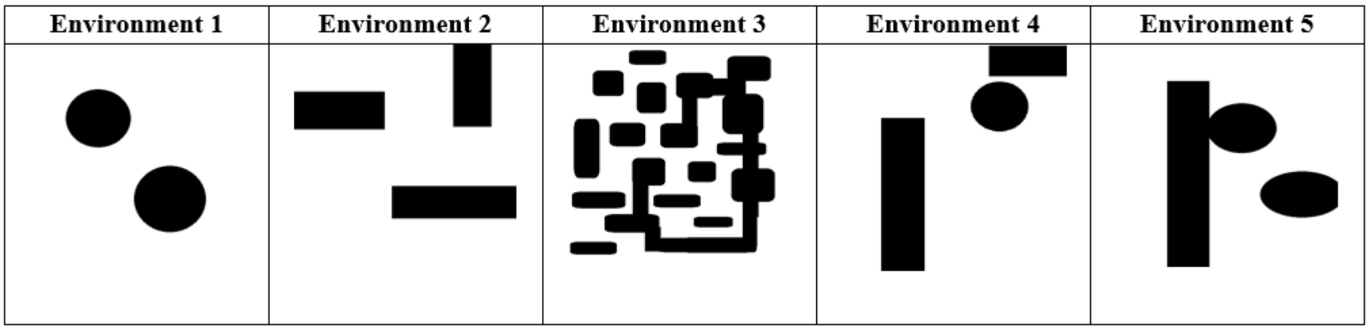

Fig. 1. Test Environments.

comparing, two key factors stand out as universally applicable metrics: path length and process time. Path length is essential as it determines the optimality of the route, while process time is critical to making real-time decisions during navigation. By basing our evaluation criteria around these two fundamental aspects, we can effectively compare algorithms on an equal footing, gaining valuable insights into their efficiency and performance in various real-world scenarios. This standardized approach will help us make informed decisions and select the most suitable algorithm for particular robotic tasks and optimization problems.

The success of an approach is largely dependent on the performance of its algorithm. This algorithm is the main source of the solution's value, as well as the amount of time it takes to complete, the number of parameters it requires, and its suitability for the problem. Path planning is a crucial issue for mobile robots and there are many solutions available. However, each is suitable for different scenarios, depending on the complexity of the environment, the time it takes to complete, and the length of the path. This paper seeks to compare popular path-planning algorithms in order to demonstrate their specifications and determine where they are most effective.

The methodology consists of some basic steps. First, we have selected popular algorithms, as we explained in the previous section. Second, five different static environments are created, which can be seen in Fig. 1. Those environments are created randomly with different levels of complexity. Because the comparison would be made in a certain environment, it is focused on how different the outputs of algorithms are in the same environment. Each algorithm runs in these environments separately by considering start point is top-left corner and end point is bottom-right corner. The results are stated as Processing Time and Path Length. In the final step, the results are compared according to the same criteria for the same environment, and the results are analyzed in the conclusion section.

The creation of these environments allowed researchers to conduct controlled experiments, conferring on them complete dominion over the attributes that characterize each scenario. This meticulous control ensured a harmonious and replicable framework for their investigations, affording the luxury of isolating and scrutinizing algorithmic performance without the convolutions of the real-world intricacies.

Our aim was to scrutinize the adaptability and flexibility of the algorithms through exposure to diverse environments. Each of these environments was meticulously tailored to embody a distinct navigation challenge, thereby facilitating a comparative analysis of algorithmic prowess across a spectrum of complexities. This systematic evaluation unveiled the relative efficiency of the algorithms within specific environmental contexts, enabling the identification of optimal performers for particular scenarios. Ultimately, this endeavor provided us with a holistic understanding of the intrinsic merits and limitations that characterize each methodology.

Computer-based environments are increasingly popular for testing mobile robot navigation algorithms. One of the main advantages of using a simulated environment is the ability to control and replicate various scenarios that are difficult to achieve in real-world environments. It allows people to test and validate their algorithms in a safe and controlled environment without the risk of damaging expensive equipment or causing harm to individuals. Another advantage of using such an environment is the ability to manipulate various environmental factors, such as obstacles, to create different scenarios for testing. So, algorithms could be tested under a range of different conditions, which is difficult to achieve in real-world environments. Computer-based environments will be used to perform path planning.

A. A Star Algorithm

The A star algorithm, commonly referred to as A star, is a widely-utilized path-finding technique used in graph traversal and search problems. This algorithm, which was first published in 1968 [15], is favored for its ability to quickly identify the shortest route between two nodes on a weighted graph. A star is a search algorithm that combines the benefits of Dijkstra's algorithm and the Greedy Best-First Search. It does this using a heuristic to calculate the cost from the current node to the target [16]. The algorithm keeps track of two values for each node: the actual cost of getting to the node from the starting point and the estimated cost from the node to the goal, which is determined by the heuristic. When considering both of these costs, A star is able to explore the most promising paths first, thus reducing the search space and making it more efficient than other search algorithms.

It is a highly effective and commonly used pathfinding technique to find the shortest route in a graph [17]. It

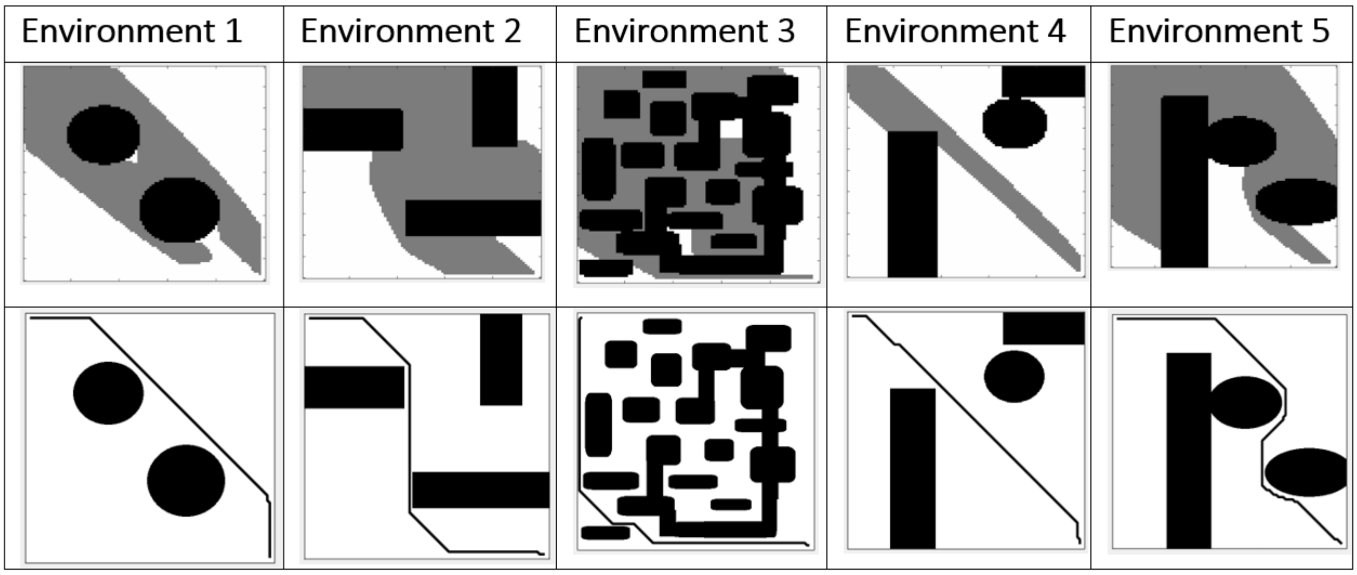

Fig. 2. A Star path for each environment

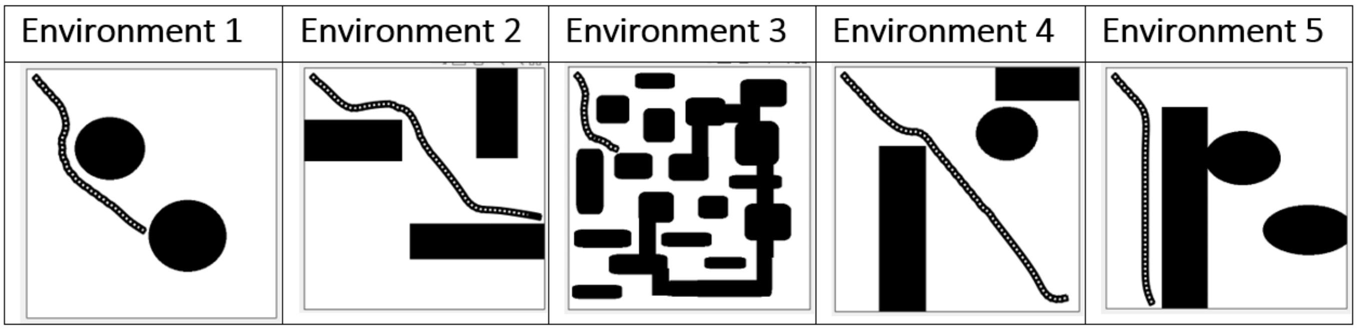

Fig. 3. Fuzzy Logic-Based Algorithm paths for each environment

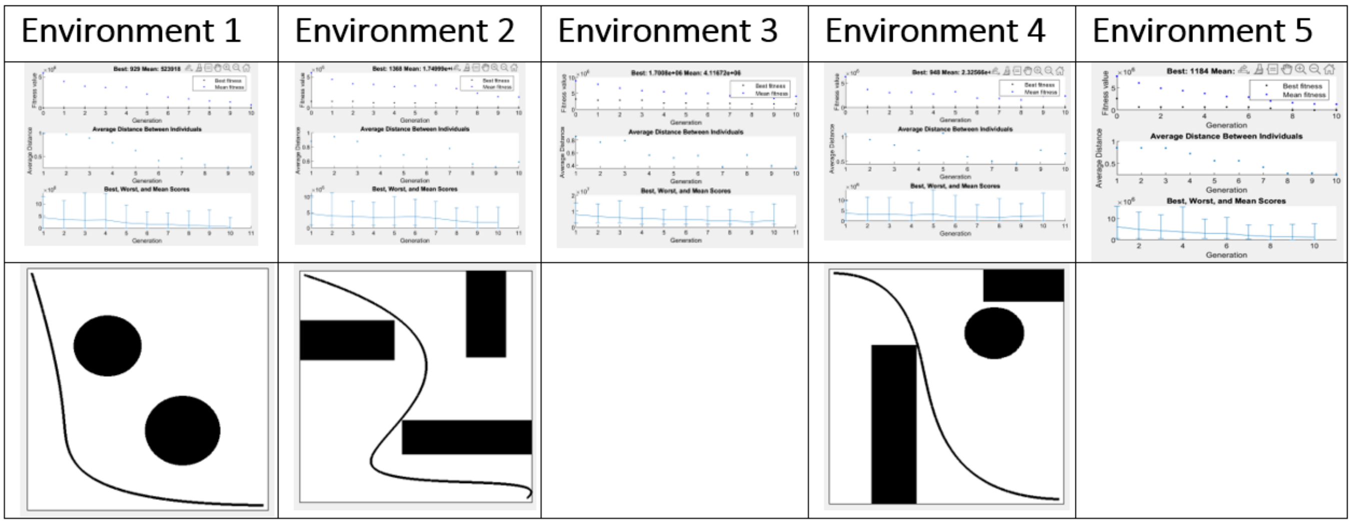

Fig. 4. Genetic Algorithm paths for each environment

guarantees the best possible outcome if the heuristic used is accurate. A star is suitable for search problems with a clearly defined goal and can quickly explore the most promising paths first, significantly reducing the search area. Balances the completeness of Dijkstra's algorithm and the speed of Greedy Best-First Search, making it suitable for a variety of applications, such as route planning, robotics, and video games.

Despite its effectiveness, A star can traverse a large area of the search space in certain situations, particularly when the heuristic is not precise or the graph is highly interconnected [18]. In some scenarios, the algorithm could be hindered by its reliance on exact heuristics. Furthermore, A star may not be the most suitable option when dealing with rapidly changing environments, as it does not manage dynamic obstacles efficiently and regular changes to the path may be computationally costly.

The A star algorithm is used to model the problem as a standard graph for the navigation of mobile robots. It seeks to find the shortest path by exploring each part of the graph one by one, with the input being the graph of the given image. The algorithm begins by scanning for nearby regions and then gradually converging on further points. To apply this algorithm to discrete spaces, it is essential to create the discrete space of the map, which is done by circulating the pixels of each received image separately. However, the downside of the A Star algorithm is that it takes a long time to compute high-resolution images. To reduce this time, pixels can be clustered, or the resolution can be decreased. It is necessary to find a balance between better results and computational time. The A Star algorithm produces better results in high-resolution images, but requires more work time.

As shown in Fig. 2, the A Star algorithm successfully navigated five different environments, following the path explained. However, the long run-time of this algorithm is a disadvantage which can be mitigated by clustering pixels or reducing the resolution. Although this may compromise the quality of the results, a balance between better results and computation time can be achieved. Nevertheless, the A Star algorithm produces smooth and short paths, making it a promising option for mobile robot navigation.

B. *Fuzzy Logic-Based Algorithm*

Fuzzy logic, which attempts to concrete future cases and was introduced by Lotfi Zadeh in 1965 with a proposal for fuzzy set theory [19]. Fuzzy logic-based algorithms use fuzzy logic, a mathematical system that deals with uncertainty and imprecision, to make decisions and control systems when there are incomplete or unclear data [20]. This concept allows for the representation of vague or nonbinary concepts by using membership functions that determine the degree of belonging to a set. What should be noted here is that there is no all-or-nothing logic, like a situation is right or wrong [21].

Fuzzy logic algorithms are often used in a variety of fields, including control systems, pattern recognition, and decision making. These algorithms are advantageous in dealing with real-world issues where exact logic may not be suitable, as they take into account uncertain data and linguistic variables. It has also been applied to many fields, from control theory to artificial intelligence. For example, it is used with Ant Colony Optimization (ACO), as a cost function for obstacles [22]. That hybrid approach is 10% better than ACO in terms of speed and route of the found road.

Fuzzy logic algorithms are adept at handling data that are imprecise, uncertain, or not clearly defined [23]. They are suitable for decision-making and control systems in cases where exact binary logic may not be applicable. These algorithms enable the integration of human expertise and domain knowledge in the form of linguistic variables and membership functions. They can model complex relationships between input and output variables, allowing the system to make informed decisions even when there are incomplete data.

Fuzzy Logic-Based Algorithms can be complex to implement and tune due to the requirement of defining membership functions and linguistic rules. Furthermore, these algorithms can be hard to interpret, as it can be challenging to comprehend and explain the rationale behind certain decisions. Additionally, they may not be as effective as other approaches in tasks with clearly defined and precise data, where classical control systems or crisp logic are sufficient.

The algorithm outputs two things: instantaneous return speed and direction, allowing the robot to approach the target while avoiding obstacles. To achieve this, a Fuzzy System is created by manually setting rules for different scenarios. Although this approach is effective in finding a way in some environments, it cannot guarantee success in all scenarios, as seen in Fig. 3.

C. *Genetic Algorithm*

The GA is a search and optimization technique that was inspired by the principles of natural selection and genetics. GA is designed to replicate the evolution process in living organisms to solve complex problems [24]. The algorithm works by creating a population of potential solutions, representing them as chromosomes, and evaluating their fitness based on a given objective function. Individuals with the highest fitness have a greater chance of reproducing and passing on their genetic information to the next generation through cross-replication and mutation operations. This cycle of selection, crossover, and mutation is repeated until satisfactory solutions are found. GA is especially useful for optimization problems with large solution spaces, as they can effectively explore different regions of the search space to find globally optimal or near-optimal solutions.

GA has been shown to be very successful in tackling complex optimization issues that involve a vast search area and multiple, possibly conflicting, objectives [25]. Through crossover and mutation operations, they are able to explore different parts of the solution space, thus allowing them to discover global or near-optimal solutions in a relatively short period of time. GA can be applied to both continuous and discrete parameter spaces, making them suitable for a wide range of problems. Additionally, they are well-suited for parallel processing, which allows for quicker convergence in large-scale optimization tasks.

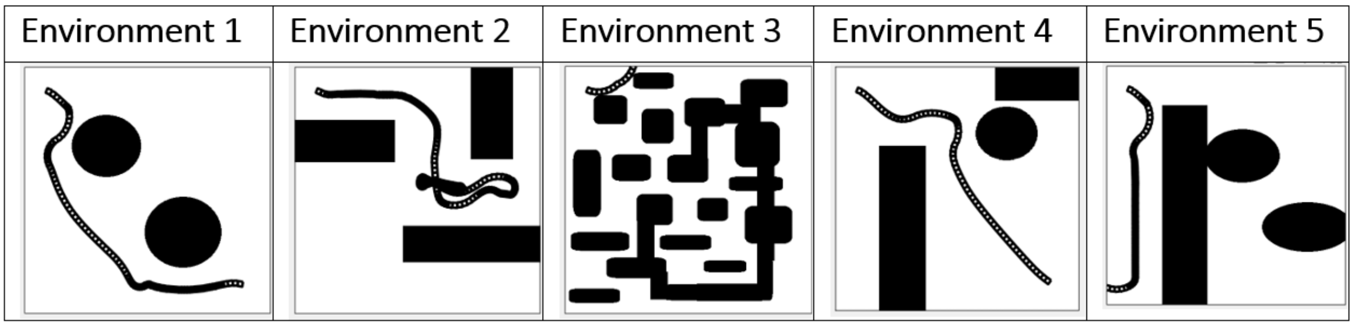

Fig. 5. APF paths for each environment

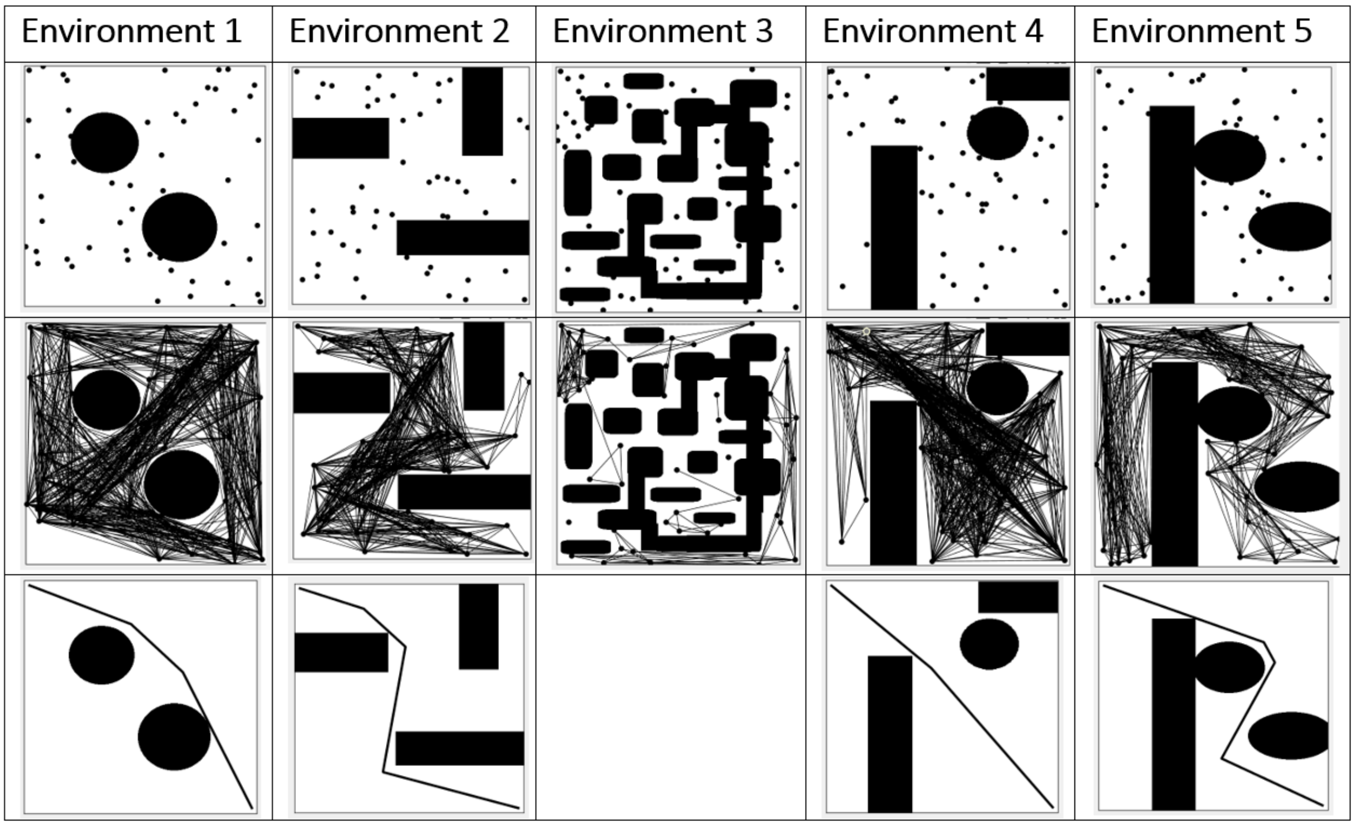

Fig. 6. Probabilistic Roadmap paths for each environment

GA is capable of effectively exploring the solution space, but it does not guarantee the discovery of the most optimal solution [26]. The quality of the result is highly dependent on the initial population and the parameters set. For complex issues, GA may require a lot of computing power and a long time to arrive at a satisfactory solution. Furthermore, the way the problem is represented and the design of the genetic operators can have a major effect on the algorithm's performance.

The GA seeks to find the optimal result rather than the best one [27]. The problem is first modeled, and the GA's parameters and working logic are determined accordingly. To do this, an objective function and the necessary variables for the GA should be defined. For example, if the distance between the target and the exit point consists of multiple stops, the distance between each stop can be calculated using the Euclidean distance and the objective function can be defined as the sum of these distances. In cases where some of the paths may pass through obstacles, these parts can be accepted as negative distances.

As seen in Fig. 4, GA cannot find a path in environments 3 and 5. In this case, it is necessary to make some changes to the GA parameters because it cannot keep up with the complexity of the environment.

D. Artifical Potential Fields

The robot then follows the gradient of the potential field, moving towards the goal while avoiding obstacles. The APF

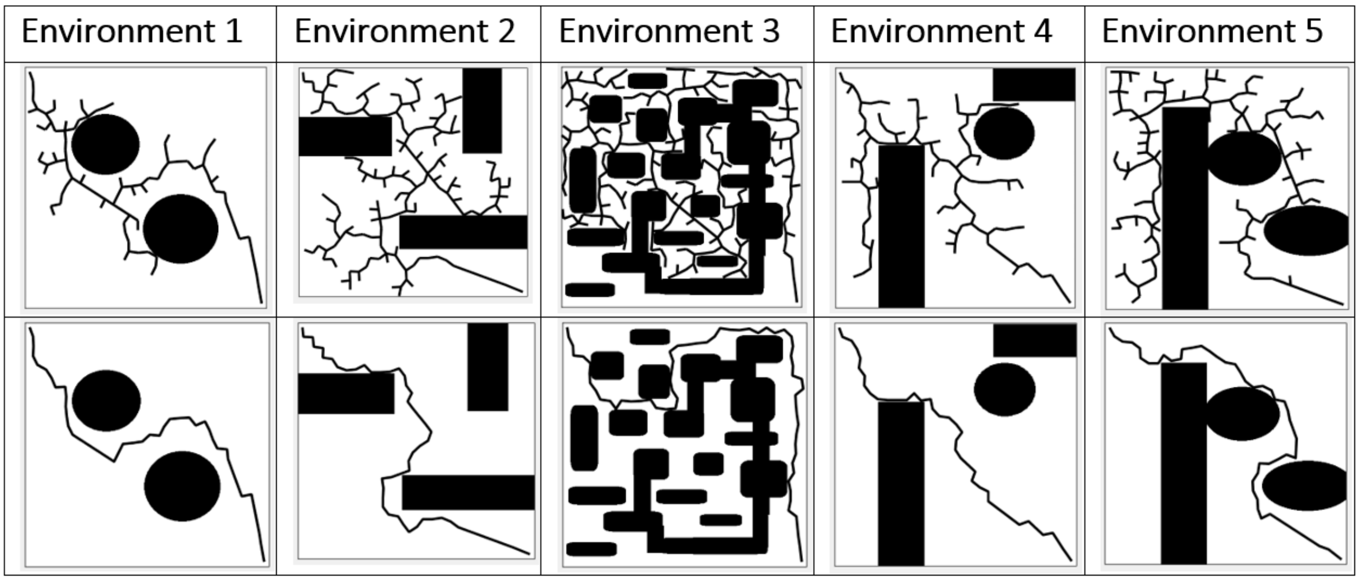

Fig. 7. Rapidly Exploring Random Trees paths for each environment

algorithm is a path-planning approach used in robotics and autonomous systems. It is designed to guide a robot or agent from a starting point to a goal while avoiding any obstacles on its way [28]. The algorithm assigns an artificial potential to both the goal and any obstacles, with attractive potentials around the goal and repulsive potentials around the obstacles. The robot then follows the gradient of the potential field, moving towards the goal while avoiding any obstacles. The robot then follows the gradient of the potential field, progressing from regions of high potential, namely obstacles, to areas of low potential, goal. This approach is relatively straightforward to execute and can effectively manage dynamic environments. Nevertheless, APF algorithms may be prone to local minima and may not always discover the most optimal path.

The APF algorithm is a simple, yet effective way to guide a robot or agent through a complex environment. It requires only local information to make decisions, making it computationally efficient [29]. Additionally, it is capable of real-time planning and is effective in static environments, making it suitable for a variety of robotic applications, such as path planning, autonomous navigation, and robotic swarm control. Implementing APF algorithms is relatively straightforward.

APF algorithms have a major flaw in that they can become trapped in local minima, resulting in the robot getting stuck or taking a suboptimal route [30]. When faced with narrow passages or complex surroundings, APF algorithms may struggle to identify a viable path. Additionally, APF algorithms are not able to effectively manage dynamic environments, as they lack the ability to plan for changes in real time, and constantly refreshing the potential field can lead to computational inefficiencies [31].

The efficacy of the APF algorithm was evaluated on five distinct forces, including left, right, forward, left, and right crossing over the boundaries and the front side of the point. However, as demonstrated in Fig. 5, the Potential Field algorithm is not successful in intricate regions, particularly in mazes where there are no paths.

E. Probabilistic Road Map

PRM algorithm is a popular motion planning technique used in robotics and computer graphics. PRM algorithm operates in two stages. First, it generates an off-line road plan and then queries this plan during runtime. The aim of the first stage is to create a random path and generate input for the second stage [32]. PRM selects points on a random plane that are not on obstacles. The number of selected vertices affects the execution time of the algorithm, where more vertices result in a better result but a longer execution time [33].

The second stage of the algorithm aims to use the path created in the first stage. The resulting graph from the first stage is suitable for any search algorithm and is formed by connecting the vertices. An A Star algorithm can be used in this stage, where each edge length is considered as edge weight and the shortest distance between the source and the target is found. As shown in Fig. 6, the algorithm was unsuccessful in finding a way in the third environment, but it showed successful results in other environments.

PRM algorithm is a reliable approach for motion planning in complex and high-dimensional configuration spaces [34]. It can generate a roadmap of valid configurations and paths that can be used for multiple planning queries. PRM is especially advantageous for scenarios with dynamic obstacles, as the roadmap is precalculated and only the start and goal configurations need to be checked for collisions. Furthermore, PRM is reliable and can handle non-holonomic constraints and complex robot geometries.

The performance of PRM is highly dependent on the quality and amount of randomly chosen configurations [35]. In areas of the configuration space that are not densely populated or well-explored, PRM may not be able to discover feasible paths.

Furthermore, the generation of the initial roadmap can be computationally costly, especially in high-dimensional spaces. The success of the algorithm is contingent upon the correct selection of parameters, such as the number of samples and the radius of connection, which may need to be adjusted.

F. Rapidly Exploring Random Trees

The RRT algorithm connects the random samples to the tree, and the tree grows until it reaches the target configuration. The algorithm is especially suitable for problems with high-dimensional configuration spaces and non-holonomic constraints. RRT algorithm produces solutions that look like a tree, with each solution point being a node and the branches resembling the branches of a tree. This algorithm is designed to explore the area, with the branches leaving the nodal point to explore the region they are heading towards. Each branch is randomly chosen and moves randomly towards the target through that branch [36]. As such, RRT is a heuristic algorithm that improves the random results it produces. The key factor here is the length of the randomly generated branch. Longer branches reduce the algorithm's running time, but result in a longer path length, whereas shorter branches provide better results, but take longer to execute. Fig. 7 displays the results of this algorithm in five different environments.

The tree structure of RRT is extended to sampled configurations, enabling a quick exploration of the space [37]. This algorithm is designed to cover the configuration space in an efficient manner while favoring unexplored regions. Its probabilistic nature makes it suitable for complex and dynamic environments, and it has been successfully used in various robotic applications, such as path planning for autonomous vehicles and robotic manipulators.

RRT is a widely used and successful algorithm for motion planning in high-dimensional configuration spaces [7]. It is capable of quickly exploring the space and has a good chance of finding a solution if one exists, given enough time. This algorithm is especially suitable for robotic systems with non-holonomic constraints and complex geometries. Furthermore, it is effective in dynamic environments since it focuses on discovering feasible paths rather than constructing a roadmap for the entire configuration space [38].

RRT is not always able to identify the most efficient route, as it is not guaranteed to discover the most advantageous overall solution [33]. The success of the algorithm is largely dependent on the sampling technique and may not be successful in areas with limited or confined space. For certain problems, RRT can be computationally expensive, and modifications of the algorithm, such as RRT* and RRTConnect, have been developed to address some of these issues.

V. Results and Analysis

The assessment of algorithms for the guidance of mobile robots has yielded useful data on their effectiveness and suitability in different situations. Each algorithm showed distinct advantages and disadvantages depending on the established criteria and performance measurements.

Table I displays the results obtained by evaluating six different algorithms in five different environments. Generally, these algorithms perform better when the solution space is very large. In this study, we compared the algorithm results based on two criteria: path length and processing time. In route planning, these criteria are inversely proportional to the algorithm parameters. Therefore, we adjusted the parameters of each algorithm based on the suitability of the environment, the desired results, and the computation time.

TABLE I
Results of 6 different algorithms for 5 different environments

		Environments				
Algorithms	**Criteria**	1	2	3	4	5
A Star	**Time**	5.3	7.7	10.6	12.0	12.3
	Length	749	854	899	702	869
GA	**Time**	5.7	2.4	X	4.0	X
	Length	931	1710	X	913	X
Fuzzy Logic B.A.	**Time**	X	X	X	1.9	X
	Length	X	X	X	903	X
APF	**Time**	3.9	X	X	1.4	X
	Length	784	X	X	780	X
PRM	**Time**	3.4	3.7	X	3.3	4.2
	Length	711	902	X	682	980
RRT	**Time**	2.6	2.9	6.4	2.4	2.9
	Length	881	1000	1057	758	982

X means that Path is not Found.

Upon meticulous analysis of the outcomes, a conspicuous observation emerges: the A Star algorithm emerges as remarkably versatile, exhibiting adeptness across diverse environments. Although its runtime is extended due to its preemptive scanning of the entire field before path planning, the resultant performance is notably commendable. The efficacy of this algorithm is particularly pronounced in expansive terrains, where its coverage prowess becomes evident. Notably, as envisaged, the runtime escalates in proportion to environmental complexity, a consequence of the algorithm's comprehensive area scanning.

In sharp contrast, the GA takes a divergent route. Operating with a roster of 10 random solutions as individual parameters, GA yields promising results in simpler settings. However, it falters in complex environments, necessitating a substantial increase in the number of random solutions to navigate intricate spaces effectively. This predicament is exemplified in the third environment, which resembles a maze, where GA was unable to chart a viable path. To enhance the algorithm's success rate in such demanding settings, a surge in the count of random solutions becomes imperative, albeit at the cost of protracted processing times.

The comparative assessment underscores the relatively modest performance of the Fuzzy Logic-Based Algorithm in relation to its algorithmic counterparts. Evidently, it faltered in yielding valid paths across four of the five examined environments, suggestive of its limitations in grappling with intricate settings. However, a glimmer of promise emerged in the case of Environment 4, where notable achievements were showcased by efficiently finding a viable path within a concise timeframe.

Conversely, the APF algorithm, while encountering pathfinding impasses in environments 2, 3, and 5, exhibited a conspicuous superiority over the Fuzzy Logic algorithm within the confines of environment 4. In particular, in this setting, the path charted by the APF algorithm was not only shorter, but also boasted smoother turns, as visually depicted in Fig. 5. These findings conclusively illustrate the pronounced efficacy of the APF algorithm in domains of increased complexity, as exemplified by environment 4.

Despite boasting a runtime more expedient than that of A Star, the PRM algorithm generates paths of relatively abbreviated length. Notably, in the 4th environment, the APF algorithm surpassed the Fuzzy Logic approach; however, juxtaposing APF against PRM highlights the latter's need for a longer processing interval to ascertain shorter paths. It is worthy of mention that, across all environments, PRM consistently yields paths briefer than those generated by APF. This distinctive characteristic positions PRM as an apt selection for endeavors that prioritize path length over computational speed.

The final algorithm within this discourse is the RRT, contrasting the PRM in both runtime and path length. According to the chosen parameters, RRT expeditiously uncovers paths, albeit at the expense of their elongated lengths. This trade-off between runtime and path length echoes a recurrent theme observed across myriad algorithms. While augmenting the iteration count in RRT could potentially yield improved paths, this enhancement comes at the trade-off of extended processing duration. Consequently, RRT emerges as a judicious choice for applications where expedited computational times and acceptance of moderately optimal paths align with the requisite criteria.

VI. Conclusion

In conclusion, the evaluation of algorithms for mobile robot navigation has provided valuable information on their individual strengths and weaknesses. Selecting the most appropriate algorithm depends on the specific requirements and constraints of the navigation problem.

For scenarios where finding the optimal path is essential, the A star algorithm offers a robust solution. When dealing with complex optimization tasks, the GA can provide high-quality solutions, although its convergence speed should be considered.

The fuzzy logic-based algorithm has the potential to manage uncertain and inexact data, especially when domain expertise is included. For navigating in static settings in real-time, the APF algorithm is a straightforward and effective option.

In complex and changing environments, the PRM algorithm is particularly useful, offering potential solutions to multiple queries. For rapid exploration and adaptation on the fly in high-dimensional spaces, the RRT algorithm is advantageous.

Further studies could explore the possibility of combining the A star with other search algorithms to benefit from their respective strengths and reduce their weaknesses. Refinement of the heuristics of A star could also be beneficial in more intricate situations. Additionally, the integration of fuzzy logic into other algorithms may improve decision-making under uncertain conditions.

In addition, creating more sophisticated versions of existing algorithms, such as the RRT and GA variants, could lead to faster convergence and better results. To address concerns about the use of fuzzy logic systems in essential applications, it is important to focus on making them more understandable and interpretable.

Robotics navigation is becoming increasingly important in many areas, and the further development and improvement of these algorithms will lead to more reliable and intelligent mobile robot systems, which will improve their performance and safety in actual use.

References

[1] E. Oztemel and S. Gursev, "Literature review of industry 4.0 and related technologies," *Journal of Intelligent Manufacturing*, vol. 31, pp. 127–182, 1 2020. [Online]. Available: doi:10.1007/S10845-018-1433-8

[2] B. Siciliano and O. Khatib, "Springer handbook of robotics," *Springer Handbook of Robotics*, pp. 1–2227, 1 2016. [Online]. Available: doi:10.1007/978-3-319-32552-1

[3] H. Y. Hsueh, A. I. Toma, H. A. Jaafar, E. Stow, R. Murai, P. H. Kelly, and S. Saeedi, "Systematic comparison of path planning algorithms using pathbench," *Advanced Robotics*, vol. 36, pp. 566–581, 2022. [Online]. Available: doi:10.1080/01691864.2022.2062259

[4] Y. Rasekhipour, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 1255–1267, 2017. [Online]. Available: doi:10.1109/TITS.2016.2604240

[5] Y. Zhao, Z. Zheng, and Y. Liu, "Survey on computational-intelligence-based uav path planning," *Knowledge-Based Systems*, vol. 158, pp. 54–64, 10 2018. [Online]. Available: doi:10.1016/J.KNOSYS.2018.05.033

[6] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, pp. 362–386, 4 2020. [Online]. Available: doi:10.1002/ROB.21918

[7] B. K. Patle, G. B. L, A. Pandey, D. R. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, pp. 582–606, 8 2019. [Online]. Available: doi:10.1016/J.DT.2019.04.011

[8] M. Korkmaz and A. Durdu, "Comparison of optimal path planning algorithms," *14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings*, vol. 2018-April, pp. 255–258, 4 2018. [Online]. Available: doi:10.1109/TCSET.2018.8336197

[9] A. Belfodil, A. Belfodil, A. Bendimerad, P. Lamarre, C. Robardet, M. Kaytoue, and M. Plantevit, "Fssd - a fast and efficient algorithm for subgroup set discovery," *Proceedings - 2019 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2019*, pp. 91–99, 10 2019. [Online]. Available: doi:10.1109/DSAA.2019.00023

[10] P. G. Luan, "Real-time hybrid navigation system-based path planning and obstacle avoidance for mobile robots," *Applied Sciences*, 2020. [Online]. Available: doi:10.3390/app10103355

[11] M. B. Alatise, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39 830–39 846, 2020. [Online]. Available: doi:10.1109/ACCESS.2020.2975643

[12] Q. Song, Q. Zhao, S. Wang, Q. Liu, and X. Chen, "Dynamic path planning for unmanned vehicles based on fuzzy logic and improved ant colony optimization," *IEEE Access*, vol. 8, pp. 62 107–62 115, 2020. [Online]. Available: doi:10.1109/ACCESS.2020.2984695

[13] F. Kiani, "Adapted-rrt: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms," *Neural Computing and Applications*, vol. 33, pp. 15 569 – 15 599, 2021. [Online]. Available: doi:10.1007/s00521-021-06179-0

[14] B. Cherkassky, "Shortest paths algorithms: Theory and experimental evaluation," *Mathematical Programming*, vol. 73, pp. 129–174, 1994. [Online]. Available: doi:10.1007/BF02592101

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968. [Online]. Available: doi:10.1109/TSSC.1968.300136

[16] 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. 59 196–59 210, 2021. [Online]. Available: doi:10.1109/ACCESS.2021.3070054

[17] N. S. Yerramilli, N. Johnson, O. S. Y. Reddy, and S. Prajwal, “Navigation systems using a*,” *2021 International Conference on Recent Trends on Electronics, Information, Communication and Technology (RTEICT)*, pp. 708–712, 2021. [Online]. Available: doi:10.1109/RTEICT52294.2021.9573801

[18] Y. Li, R. Jin, X. Xu, Y. yuan Qian, H. Wang, S.-S. Xu, and Z. Wang, “A mobile robot path planning algorithm based on improved a* algorithm and dynamic window approach,” *IEEE Access*, vol. PP, pp. 1–1, 2022. [Online]. Available: doi:10.1109/ACCESS.2022.3179397

[19] L. A. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965, cited By :62358. [Online]. Available: doi:10.1016/S0019-9958(65)90241-X

[20] J. C. Mohanta and A. Keshari, “A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation,” *Applied Soft Computing Journal*, vol. 79, pp. 391–409, 6 2019. [Online]. Available: doi:10.1016/J.ASOC.2019.03.055

[21] C. Punriboon, C. So-In, P. Aimtongkham, and N. Leelathakul, “Fuzzy logic-based path planning for data gathering mobile sinks in wsns,” *IEEE Access*, vol. 9, pp. 96 002–96 020, 2021. [Online]. Available: doi:10.1109/ACCESS.2021.3094541

[22] S. Aggarwal and N. Kumar, “Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges,” *Computer Communications*, vol. 149, pp. 270–299, 1 2020. [Online]. Available: doi:10.1016/J.COMCOM.2019.10.014

[23] S. Wang, “Mobile robot path planning based on fuzzy logic algorithm in dynamic environment,” *2022 International Conference on Artificial Intelligence in Everything (AIE)*, pp. 106–110, 2022. [Online]. Available: doi:10.1109/AIE57029.2022.00027

[24] I. Zubeiri, Y. E. Morabit, and F. Mrabti, “Genetic algorithm for vertical handover (gafvh) in a heterogeneous networks,” *International Journal of Electrical and Computer Engineering*, vol. 9, pp. 2534–2540, 8 2019. [Online]. Available: doi:10.11591/IJECE.V9I4.PP2534-2540

[25] S. Katoch, S. S. Chauhan, and V. Kumar, “A review on genetic algorithm: past, present, and future,” *Multimedia Tools and Applications*, vol. 80, pp. 8091–8126, 2 2021. [Online]. Available: doi:10.1007/S11042-020-10139-6

[26] M. Nazarahari, E. Khanmirza, and S. Doostie, “Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm,” *Expert Systems with Applications*, vol. 115, pp. 106–120, 1 2019. [Online]. Available: doi:10.1016/J.ESWA.2018.08.008

[27] A. N. Alfiyatin, W. F. Mahmudy, and Y. P. Anggodo, “K-means clustering and genetic algorithm to solve vehicle routing problem with time windows problem,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 11, pp. 462–468, 8 2018. [Online]. Available: doi:10.11591/IJEECS.V11.I2.PP462-468

[28] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, “Mobile robot path planning using membrane evolutionary artificial potential field,” *Applied Soft Computing*, vol. 77, pp. 236–251, 4 2019. [Online]. Available: doi:10.1016/J.ASOC.2019.01.036

[29] D. A. V. Veldhuizen and G. B. Lamont, “Multiobjective evolutionary algorithms: analyzing the state-of-the-art.” *Evolutionary computation*, vol. 8, pp. 125–147, 2000. [Online]. Available: doi:10.1162/106365600568158

[30] J. R. Sánchez-Ibáñez, C. J. Pérez-Del-pulgar, and A. García-Cerezo, “Path planning for autonomous mobile robots: A review,” *Sensors*, vol. 21, 12 2021. [Online]. Available: doi:10.3390/S21237898

[31] S. Hacohen, S. Shoval, and N. Shvalb, “Probability navigation function for stochastic static environments,” *International Journal of Control, Automation and Systems*, vol. 17, pp. 2097–2113, 8 2019. [Online]. Available: doi:10.1007/S12555-018-0563-2

[32] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, “Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5113–5120, 9 2018. [Online]. Available: doi:10.1109/ICRA.2018.8461096

[33] Z. Wang and J. Cai, “Probabilistic roadmap method for path-planning in radioactive environment of nuclear facilities,” *Progress in Nuclear Energy*, vol. 109, pp. 113–120, 11 2018. [Online]. Available: doi:10.1016/J.PNUCENE.2018.08.006

[34] A. A. Ravankar, A. Ravankar, T. Emaru, and Y. Kobayashi, “Hpprm: Hybrid potential based probabilistic roadmap algorithm for improved dynamic path planning of mobile robots,” *IEEE Access*, vol. 8, pp. 221 743–221 766, 2020. [Online]. Available: doi:10.1109/ACCESS.2020.3043333

[35] W. Khaksar, K. S. B. M. Sahari, F. B. Ismail, M. Yousefi, and M. A. Ali, “Runtime reduction in optimal multi-query sampling-based motion planning,” *2014 IEEE International Symposium on Robotics and Manufacturing Automation, IEEE-ROMA2014*, pp. 52–56, 10 2015. [Online]. Available: doi:10.1109/ROMA.2014.7295861

[36] L. Chen, Y. Shan, W. Tian, B. Li, and D. Cao, “A fast and efficient double-tree rrt star-like sampling-based planner applying on mobile robotic systems,” *IEEE/ASME Transactions on Mechatronics*, vol. 23, pp. 2568–2578, 12 2018. [Online]. Available: doi:10.1109/TMECH.2018.2821767

[37] J. Denny, R. Sandström, A. Bregger, and N. M. Amato, “Dynamic region-biased rapidly-exploring random trees,” *Springer Proceedings in Advanced Robotics*, vol. 13, pp. 640–655, 2020. [Online]. Available: doi:10.1007/978-3-030-43089-4_41

[38] Z. Tahir, A. H. Qureshi, Y. Ayaz, and R. Nawaz, “Potentially guided bidirectionalized rrt* for fast optimal path planning in cluttered environments,” *Robotics and Autonomous Systems*, vol. 108, pp. 13–27, 10 2018. [Online]. Available: doi:10.1016/J.ROBOT.2018.06.013