Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Comparison Study of Path Planning and Collision Avoidance Algorithms for UAV Operations

Dr. SWARNALATHA ANUMULA

swarnalatha.aero@kcgcollege.com

KCG COLLEGE OF TECHNOLOGY

Mr. Adarsh Pratap Singh

KCG COLLEGE OF TECHNOLOGY

Ms. Angelin Anani A.S

KCG COLLEGE OF TECHNOLOGY

Research Article

Keywords: Path Planning, Collision Avoidance, Swarm Drones, A* Algorithm, Dijkstra's Algorithm, Rapidly-exploring Random Tree (RRT), Probabilistic Roadmap (PRM), Dynamic Window Approach (DWA), Back-and-Forth Algorithm

Posted Date: April 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9242272/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Comparison Study of Path Planning and Collision Avoidance Algorithms for UAV Operations

Dr. Swarnalatha Anumula[1], Mr. Adarsh Pratap Singh[2], Ms. Angelin Anani A.S[3]

[1] Associate Professor, Department of Aeronautical Engineering, KCG College of Technology

[2] Department of Aerospace Engineering, KCG College of Technology

[3,]Department of Aerospace Engineering, KCG College of Technology

Corresponding author: swarnalatha.aero@kcgcollege.com

ABSTRACT:

In this work, Path planning and collision avoidance algorithms were studied, and the performance of multiple algorithms - A*, Dijkstra, Rapidly-exploring Random Tree (RRT) and Back and Forth were analysed. A comparative study of these algorithms has been done in the research by considering the parameters execution time and path length. The research leads to work on four different scenarios of obstacles under three cases with varying waypoints, efficient distribution of waypoints and a certain minimum waypoint common for all other scenarios. The research concludes that A star and Dijkstra's algorithm give an optimal path but take high computational time, whereas RRT gives solutions quickly but the path is not optimal. The conclusion assists the selection of an optimal path-planning algorithm to achieve safe and efficient operation in dynamically changing environments.

KEYWORDS:

Path Planning, Collision Avoidance, Swarm Drones, A* Algorithm, Dijkstra's Algorithm, Rapidly-exploring Random Tree (RRT), Probabilistic Roadmap (PRM), Dynamic Window Approach (DWA), Back-and-Forth Algorithm.

1 INTRODUCTION:

Unmanned Aerial Vehicles (UAVs), often known as drones, have revolutionised numerous industries, from logistics and surveillance to disaster response and precision agriculture. A major challenge in deploying UAVs effectively lies in enabling autonomous navigation, particularly through environments that are filled with obstacles and dynamic challenges. Achieving reliable path planning and collision avoidance is essential for ensuring safe, energy-efficient, and goal-oriented flight operations. As the demand for intelligent aerial systems grows, the development and refinement of navigation algorithms have become increasingly critical.

Traditional methods such as A* and Dijkstra's algorithm have long provided deterministic and optimal pathfinding solutions; however, they often struggle with computational burdens in dynamic or large-scale environments. Sampling-based strategies like Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM) offer greater flexibility in navigating

complex spaces but typically require enhancements to meet the demands of real-time responsiveness. To address dynamic and cluttered environments, local optimisation techniques such as the Dynamic Window Approach (DWA) have gained traction, offering improved manoeuvrability and adaptability. Meanwhile, systematic sweep patterns, embodied by the Back-and-Forth algorithm, have shown immense value in missions requiring complete area coverage, such as agricultural monitoring, search and rescue operations, and environmental mapping.

This study is centered around analysing and comparing a diverse set of path planning strategies tailored for swarm drone navigation. Special emphasis is placed on evaluating each method's computational efficiency, path smoothness, execution speed, and obstacle avoidance capability under varying environmental conditions. Through systematic simulation and testing, the research explores the real-world applicability of A*, Dijkstra, RRT, PRM, DWA, and Back-and-Forth algorithms. In addition, hybrid approaches that blend the strengths of multiple methods are considered as potential solutions for achieving superior navigation performance.

The main objectives of this investigation are to assess algorithmic performance across different obstacle configurations, enhance efficiency for real-time swarm drone operations, compare pathfinding metrics such as execution time and path length, and explore integration strategies for optimised results. Ultimately, this research aims to advance UAV autonomy by offering in-depth insights into the capabilities and limitations of contemporary path planning methodologies, contributing to the development of drones that can operate reliably and intelligently in a wide range of mission-critical environments.

2 RELATED WORK:

Path planning is the backbone of autonomous navigation wherein A^* algorithm has been proven to be important in many applications. One of the remarkable improvements was the use of the Predictive A* approach, which enhanced the anticipation of the obstacle dynamics within the algorithm [1]. Additionally, with the use of hash tables in A*, computational time has been reduced significantly making navigation in real-time applications to be optimized [2]. Such improvements as quick path generation schemes decreased the drone-based navigation system latency of A strategies even more, simplifying navigation systems [3]. Comparative analysis of A by D. Mandoi shows that D. Mandoi is not the best in path optimality, but algorithms such as Dijkstra and RRT can be better than A in certain terrain conditions [4]. Recent adaptive heuristic changes have introduced to A have improved its capacity to adapt to the changing environment making it more robust [5]. Referring to Dijkstra algorithm, other researchers such as D. Karve have demonstrated that the algorithm has remained unmatched in determining the actual shortest paths especially when dealing with unpredictable conditions such as turbulence [6]. It has an extension to its 3D applications by E. Dhulkefl in which high-fidelity path planning in aerial vehicles was possible [7]. Additional changes, including methods of aligning the matrices, would speed up the planning of a trajectory without compromising accuracy [8]. Combination with metaheuristic algorithms such as Particle Swarm Optimization has enabled new directions of hybrid Dijkstra schemes [9], and multi-angle scanning benefits offered by Y. Sun makes UAV routing much more efficient [10].

In the meantime, the Probabilistic Roadmap (PRM) approach has been developed much further, and the pseudo-random sampling is optimized by Q. Li to achieve a better coverage of the nodes [11]. PRM has been strengthened against uncertain environments through risk-sensitive extensions based on Monte Carlo simulations, investigated by F. R. Bana [12]. The dimensionality-reduction methods were very useful in terms of Indoor UAV applications because they maximized the efficiency of roadmap generation [13]. G. The strategy of optimizing node connections by Li also improved the overall viability of PRM [14], and dynamic recalculation of boundaries by S. Aviram made PRM planning even more reactive to changes in real time [15].A search algorithm that is currently catching the eye of many is the Rapidly-exploring Random Tree (RRT) algorithm, although some enhancements have been made to the algorithm by F. Yang that employs the Ant Colony Optimization techniques and thus, has enhanced the efficiency of the algorithm considerably [16]. There are two-node expansion strategies suggested by J. Chen that reduced significantly the time required to discover viable paths [17]. The quest to minimize redundant sampling [18] and the comparative performance study between different forms of RRT [19] have gotten to define the best practices currently. The efforts of J. Fang were aimed at removing redundant nodes and path smoothing, which improved RRT-based planning systems to a greater extent [20].

Along with these classic strategies of path planning, there are Back-and-Forth algorithms that proved to be incredibly potential, especially when it comes to area coverage missions. A study conducted by L. Jintao focused on covering the sweep efficiently and not diminishing the accuracy of the overlap [21]. The applications that Young-Jo Lee conducted on the use of UAVs in agriculture modified these patterns to make sure that the whole field is scanned with a limited amount of energy consumption [22]. Meanwhile, Yuxin Wang also proposed adaptable back-and-forth sweeps that could cope with dynamic, moving obstacles in real time [23], and M. Ortiz also added obstacle avoidance control to the conventional sweep techniques to guarantee both safety and efficiency when performing coverage tasks [24].

Lastly, the Dynamic Window Approach (DWA) has been a great enhancement of collision avoidance in dynamic movements. DWA, which was first introduced by Fox et al. [25], was a game changer in robot navigation by optimising the velocity commands under dynamic constraint. Subsequent Zhang had refined velocity sampling methods that made obstacle avoidance more precise [26]. Wu has made further improvements in the flexibility of DWA by predictive integrations of moving obstacle behaviour [27]. Recent activities such as Jang have been working on cost optimalisation of cost functions in DWA to provide smooth and efficient routes [28], and Chen modified the strategy to high-speed UAVs, which guarantee real-time responsiveness despite high flight-condition demands [29].

This research will aim at learning and applying various path planning algorithms to navigate in the world of obstacles. Each drone must: Determine an effective and optimal collision free path between the origin point and the target point. Assess and optimize computer expense in order that we are able to apply it to real life applications. Keep a balance of optimum path, execution time and waypoint efficiency. This research has been carried out with the intention of analyzing the performance of each of the algorithms, including A*, Dijkstra, RRT, DWA, PRM, Back-and-Forth.

3.PATH PLANING ALGORITHMS:

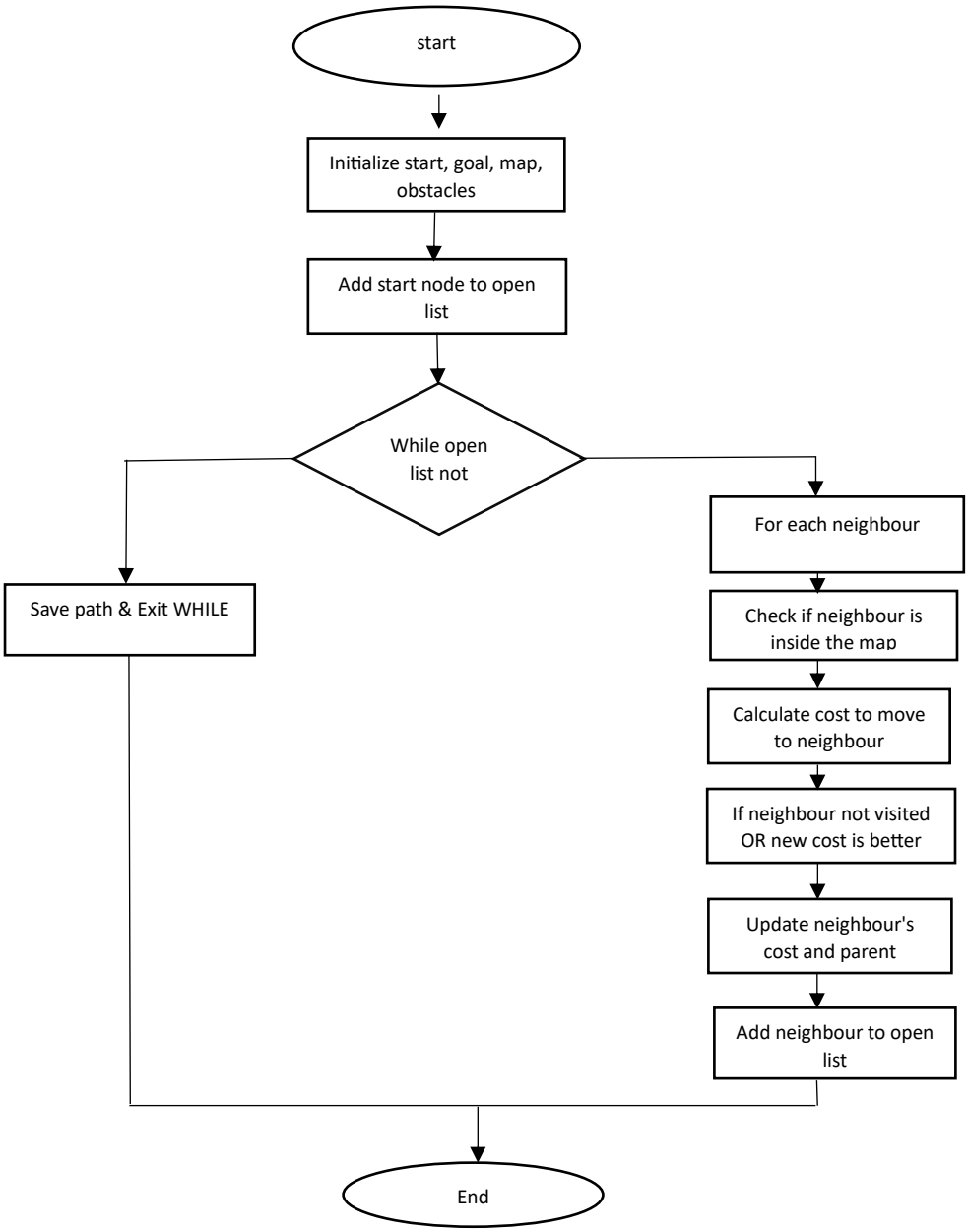
The research evaluates and observes six different algorithms viz., A*, Dijkstra, RRT, DWA, PRM, and Back-and-Forth.

3.1 A* Algorithm

The A* algorithm is a well-established path planning technique widely used for its ability to reliably find the shortest and most efficient path between a start and a goal position. It combines the strengths of Dijkstra's algorithm and Greedy Best-First Search, using a cost function that balances the actual cost to reach a node and an estimated cost to the goal. The cost function, denoted as $f(n)=g(n)+h(n)f(n) = g(n) + h(n)$, where $g(n)g(n)$ is the accumulated cost from the start to the current node and $h(n)h(n)$ is a heuristic estimate of the remaining distance to the goal, allows the algorithm to navigate intelligently through the search space.

In this study, A* was implemented in a two-dimensional grid environment, where free spaces and obstacles were explicitly mapped.

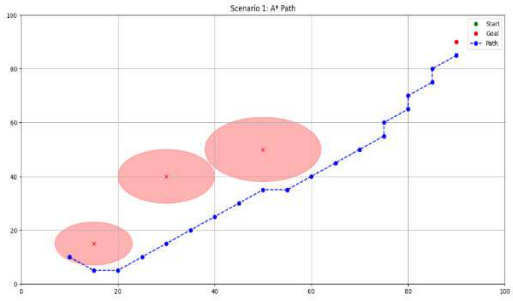
3.1.1 Flow Chart



The simulation has been done using python in a pre-determined 2D obstacle environment of size 100 x 100 m, where drone starts from pre-defined starting to point to target point. The obstacle environment includes obstacle represented as circle and its radius as threat range i.e. the dangerous zone where drones cannot enter. The movement of drone is possible in 8 directions and allowing diagonal and straight-line traversal. The threat range accounts for safety precautions to simulate real world requirements such as drone size and turning radius, navigation sensor uncertainty, reaction time margin and dynamic error in control systems. The simulation has been done under four obstacle scenarios with predefined obstacle coordinates as well as threat range (radius). Scenario 1 has three obstacles with coordinates [(15, 15), (30, 40), (50, 50)] and threat range [8, 10, 12] respectively in metres. Scenario 2 has eight obstacles with coordinates [(12, 48), (24, 33), (27, 58), (30, 70), (55, 80), (59, 52), (70, 34), (70, 65)] and threat range [12, 9, 9, 10, 9, 10, 12, 7] respectively in metres. It is moderately complex. Scenario 3 has nine obstacles with coordinates [(20, 70), (25, 19), (25, 39), (45, 20), (47, 41), (50, 61), (70, 53), (75, 74), (78, 20)] and threat range [15, 8, 8, 9, 8, 8, 10, 9, 15] respectively in metres. It is more complex than other previous scenarios. Scenario 4 has nine obstacles with coordinates [(10, 50), (20, 20), (30, 42), (30, 80), (50, 55), (60, 10), (60, 80), (65, 38), (75, 65)] and threat range [10, 8, 7, 9, 9, 8, 9, 9, 10] respectively in metres as shown in figures 1 to 12.

All the four scenarios are tested under three cases where case 1 randomly chooses waypoints and finds the path whereas in case 2 waypoints are distributed in order to optimize the path and case 3 uses minimum number of waypoints required for most complex obstacle scenario as reference for other scenarios as shown in Figures 1to 12.

CASE 1:



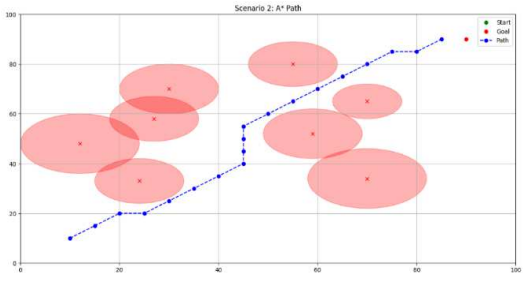
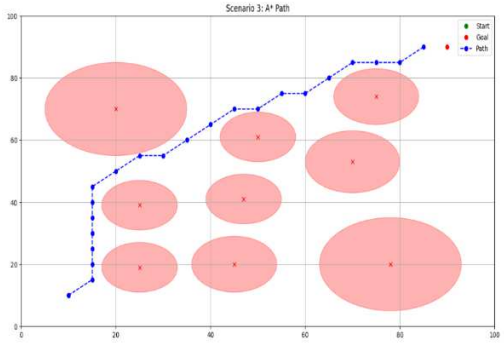
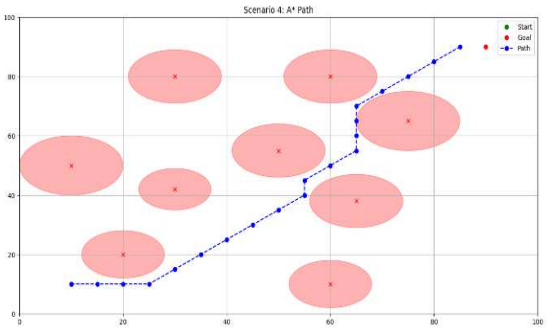


Figure1:a.Scenario1 *b.Scenario2*





c.Scenario3 *d. Scenario 4*

CASE 2:

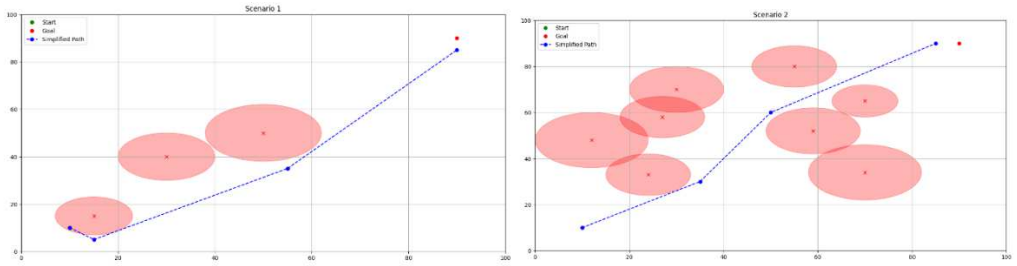
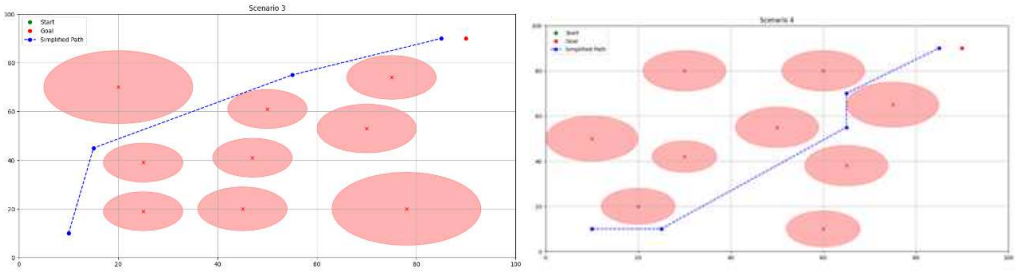


Figure 2: a.Scenario 1 *b.Scenario 2*



c.Scenario3 *d Scenario 4*

CASE 3

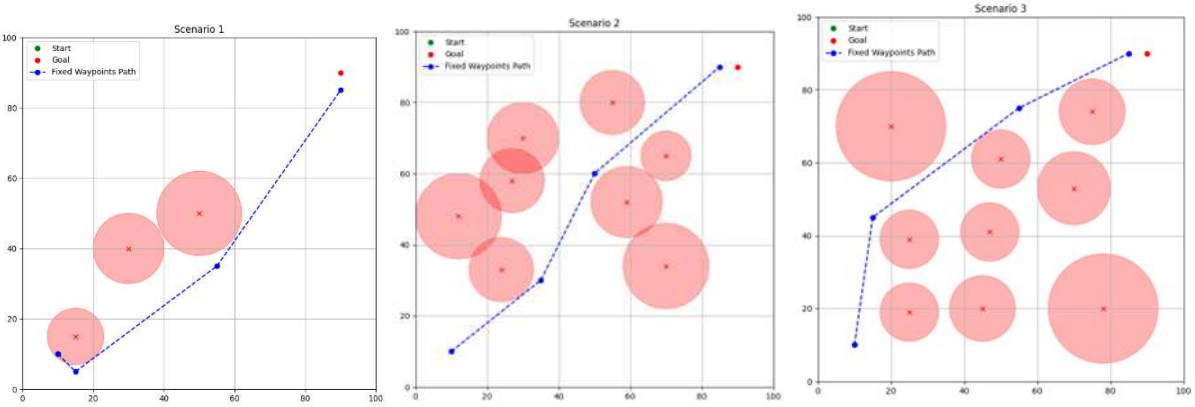
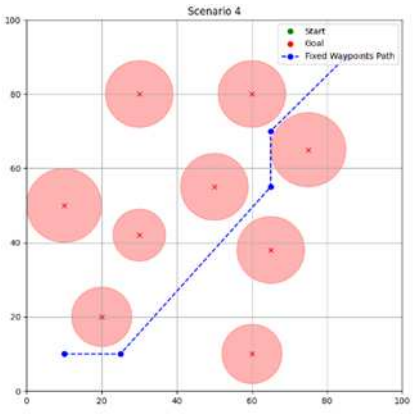


Figure 3: a.Scenario 1 *b.Scenario 2* *c.Scenario*



d.Scenario

3.2 DIJKSTRA'S ALGORITHM

The algorithm by Dijkstra is a traditional path planning technique that is known to be optimally guaranteed, that of finding the shortest path between a single source node and all other nodes in a graph. It works by finding the closest unexplored node and moving outwards in an equal manner until the target node is hit. The algorithm only takes into account the cumulative cost between the initial node and a specific point, omitting any form of heuristic direction, making it accurate but usually costly in large settings in terms of computational power.In this work, the algorithm by Dijkstra was applied on a two-dimensional grid map, and free spaces and obstacles were explicitly represented. The design of the movement model enabled movement transitions in the four cardinal directions up, down, left and right to suit realistic movement capabilities that UAVs had in structured environments. The cost between two adjacent nodes was set to be equal, that is, the same amount of effort is used to travel between neighboring grid cells. The list of frontier nodes was maintained using a priority queue structure, which was sorted in terms of cumulative distance of start node. The node that had the minimum costs was picked to grow at every iteration. Should there be a neighboring node that can be reached at a lower accumulated cost, the cost was updated and when again put back in the queue to be explored. This was repeated until the goal node was expanded and then a path was reconstructed by following the way through parent nodes.

In this study, Dijkstra algorithm ensures the optimal solution in case all costs of the edges are non-negative, which was the case in this study. Nonetheless, its consistent exploration approach can frequently cause superfluous expansions in areas distant to the objective, which cause bigger computational requirements than the heuristic-based algorithms such as A. Under the conditions of the static environment, when path optimality is the key factor, and there are enough computational resources to solve the problem, Dijkstra algorithm is a reliable path planner. However, with real-time application and UAV swarms, especially in dynamic or large scale setting, scalability is a blocker unless further optimisations are made.

3.2.1 Algorithm:

1. Initialize Dijkstra Open List:

Open List = Priority queue with (cost, position).

cost_so_far[start] = 0

came_from[start] = None

2. Expand Nodes:

Pop the lowest cost node.

For each 8-connected neighbor:

If neighbor within bounds [0,100] and collision-free with obstacles:

Calculate movement cost (Euclidean distance).

If neighbor not visited or new lower cost found:

Update cost_so_far and came_from.

Add neighbor to Open List.

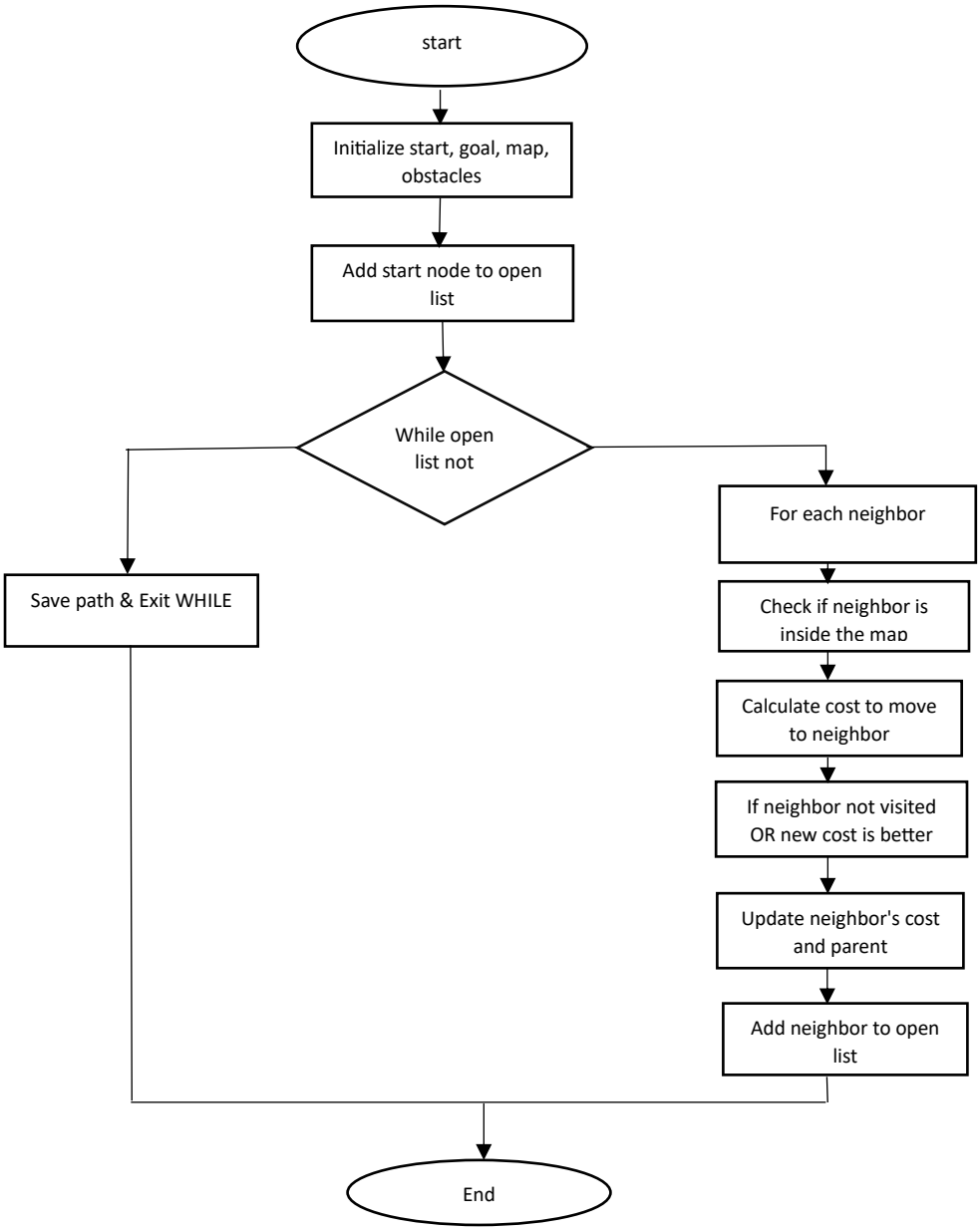
3. Goal Reached Check:

If node is within grid_size distance from the goal:

Backtrack using came_from to reconstruct path.

Proceed to Step 3

3.2.2 Flowchart:



CASE 1

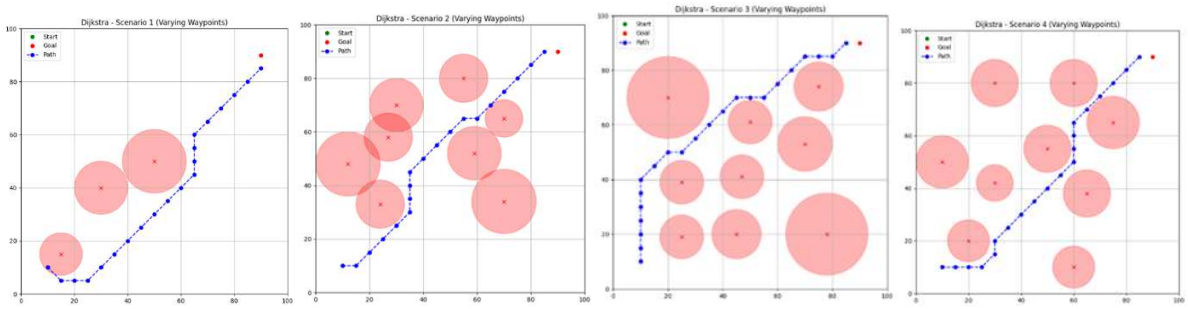


Figure 4:a. Scenario 1 b. Scenario 2 c. Scenario 3 d. Scenario 4

CASE 2

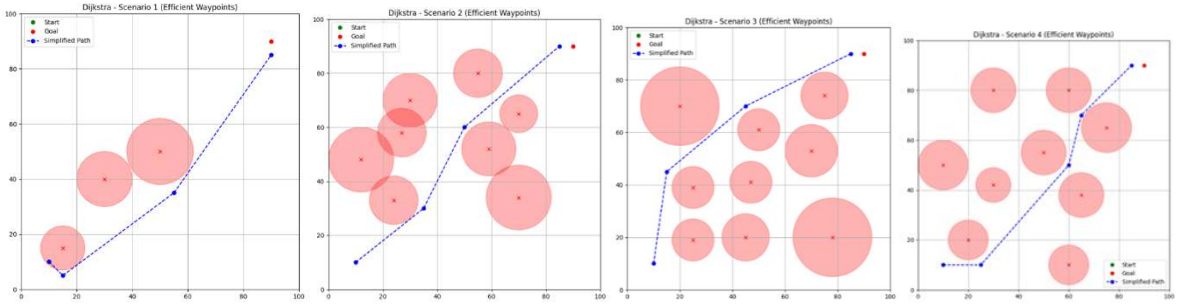


Figure5 :a.Scenario1 b.Scenario2 c.Scenario 3 d Scenario 4

CASE 3

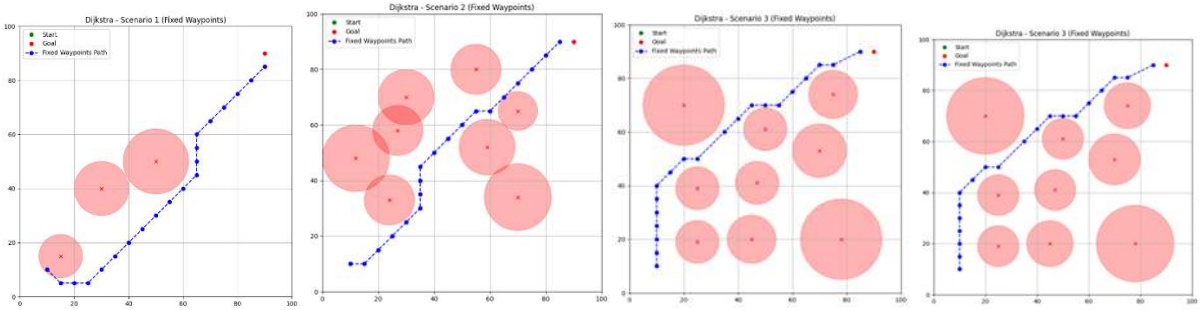


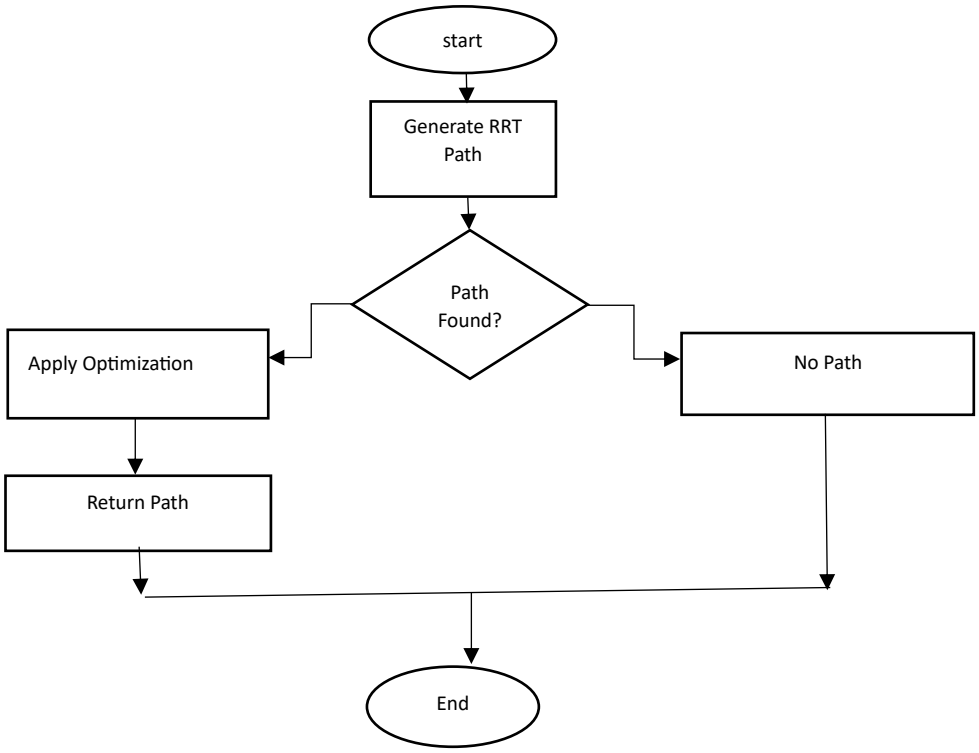
Figure 6 : a. Scenario 1 b.Scenario 2 c.Scenario3 d.Scenario 4

3.3 RRT **ALGORITHM**

The Rapidly-Exploring Random Tree (RRT) algorithm is a widely used sampling-based method for path planning, particularly suited for high-dimensional or complex environments where deterministic algorithms struggle. RRT operates by incrementally building a tree rooted at the starting position, randomly exploring the space by extending branches toward sampled points. This stochastic nature allows RRT to efficiently explore large, unknown, or intricate search spaces without the need for complete environmental knowledge beforehand.

In this research, RRT was implemented within a bounded two-dimensional workspace populated with circular obstacles of varying sizes. The tree growth was guided by randomly generating sample points across the environment and identifying the nearest existing node in the tree structure. From the nearest node, an extension was attempted towards the random sample, with the step size constrained to a predefined limit to maintain manageable and feasible movements suitable for UAV navigation.

3.3.1 Flowchart:



Case 1

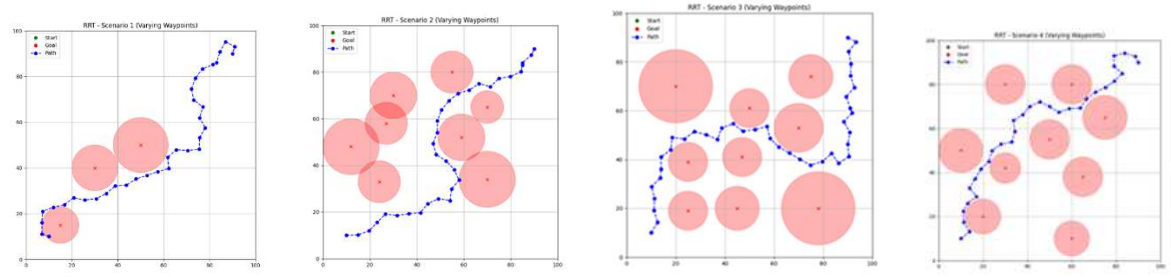


Figure 7: a.Scenario1 b.Scenario 2 c. Scenario 3 d. Scenario 4

CASE 2

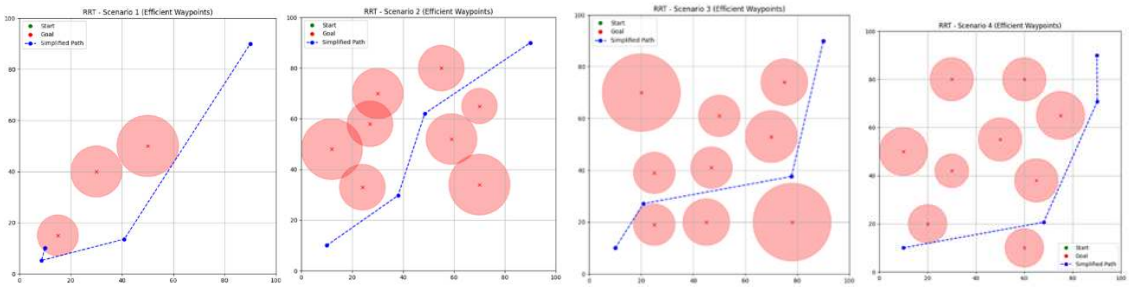


Figure 8: a.Scenario 1 b. Scenario 2 c. Scenario 3 d.Scenario 4

CASE 3

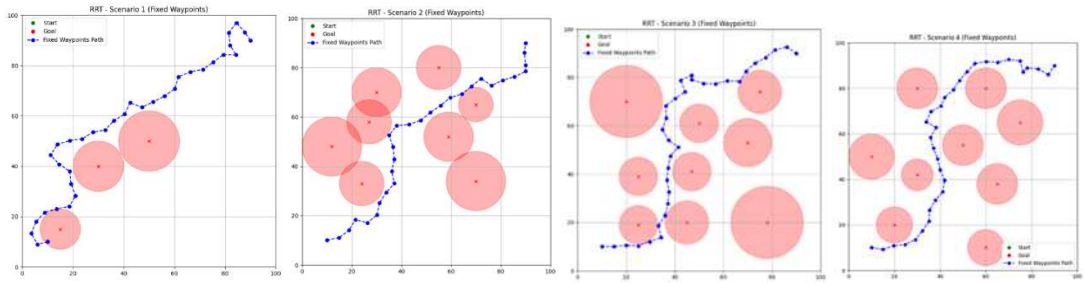


Figure 9: a.Scenario 1 b.Scenario 2 c.Scenario 3 d .Scenario 4

3.4 BACK AND FORTH ALGORITHM

Back-and-Forth algorithm, sometimes known as a coverage path planning strategy, is especially useful when a UAV has to cover an area in a systematic manner and guarantee that all parts of it have been covered, like in a search and rescue operation, surveying an agricultural field, or mapping trip. In contrast with conventional goal-based path planning algorithms, Back-and-Forth is based on complete area coverage and does not only aim at a destination point.

This paper modified the Back-and-Forth approach to allow swarming drones to cover delimited areas by making sure they circumvented circular barriers cautiously. The operational space was partitioned using parallel lanes depending on the range of sensors on the drone and its flight dynamics. The drones would move fully over one lane and then switch to the other side of the road creating a regular and orderly pattern of coverage that looks like a lawnmower.

3.4.1 Algorithm:

Forward Pathfinding:

Start from the start point and generate a path towards the goal using basic_pathfinding().

If pathfinding fails, return None.

Backward Pathfinding:

Start from the goal point and generate a path back towards the start.

If pathfinding fails, return None.

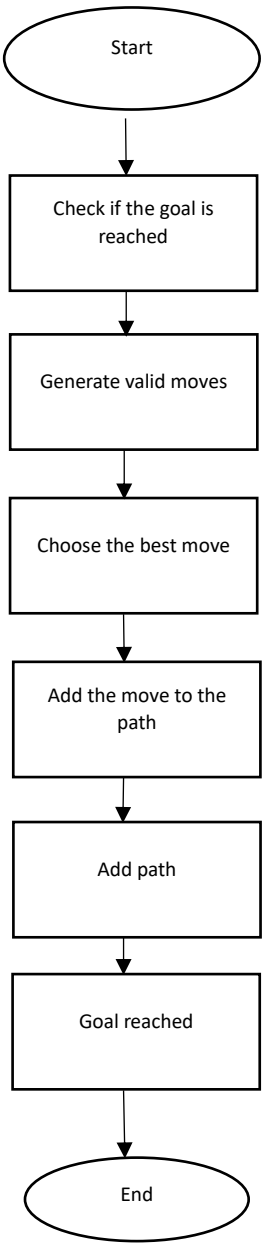
Combine the paths:

Reverse the backward path to bring it from goal to start.

Merge the forward path and the reversed backward path, ensuring no overlap.

Return the combined path.

3.4.1.1 Flowchart:



CASE 1

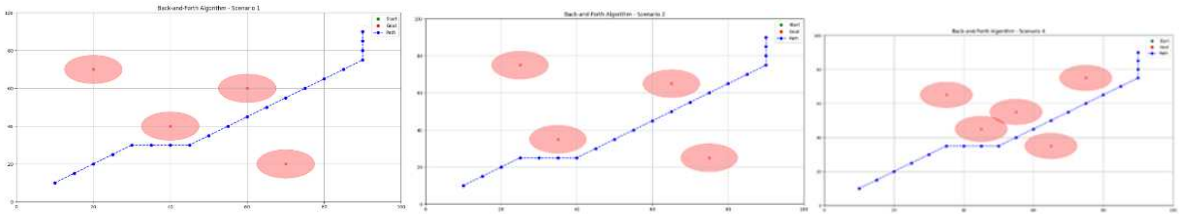


Figure 10: a.Scenario 1 *b.Scenario 2* c.Scenario 3

CASE 2:

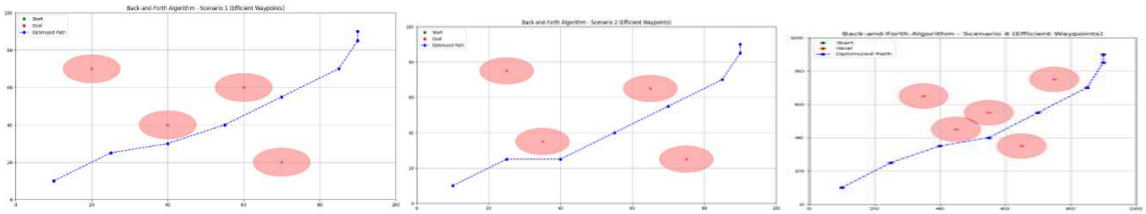


Figure 11: a.Scenario 1 *b. Scenario 2* *c. Scenario 3*

CASE 3:

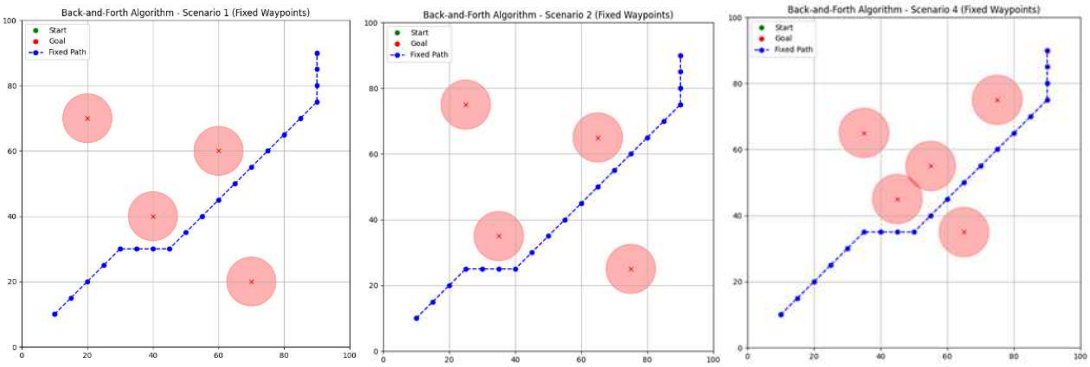


Figure 12: a.Scenario 1 *b. Scenario 2* *c. Scenario 3*

3.5 Probabilistic Uniform Roadmap (PRM) Algorithm.

The Probabilistic Roadmap Method (PRM) is a very popular approach to path planning in non-dynamic environments, especially successful when it comes to complex spaces where the direct analytical solution is not easily available. PRM operates based on the following steps: During a preprocessing step, a roadmap of the environment is typically built, in which many random samples are created throughout the free space, and viable connections are formed between the samples, using distance and obstacle-free paths.

3.5.1 Algorithm:

Initialization:

Define the start and goal positions.

Define the obstacle positions and threat radii.

Set number of random samples n, and connection radius r.

Random Sampling:

Generate a set of random points in the free space, ensuring they do not collide with any obstacles by checking the collision-free condition:

Repeat until n valid points are sampled. Add start and goal points to the set.

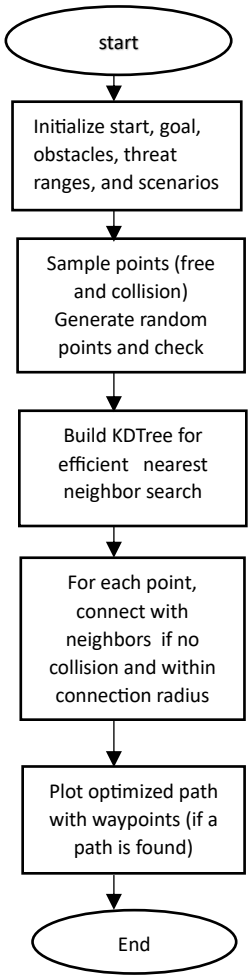
Graph Construction:

Build a graph G where each valid sample is a node.

For each pair of nodes check if they are within the connection radius

Add an edge between nodes if the path between them is collision-free (use interpolation to check for collisions along the path).

3.5.1.1 Flowchart:



CASE 1

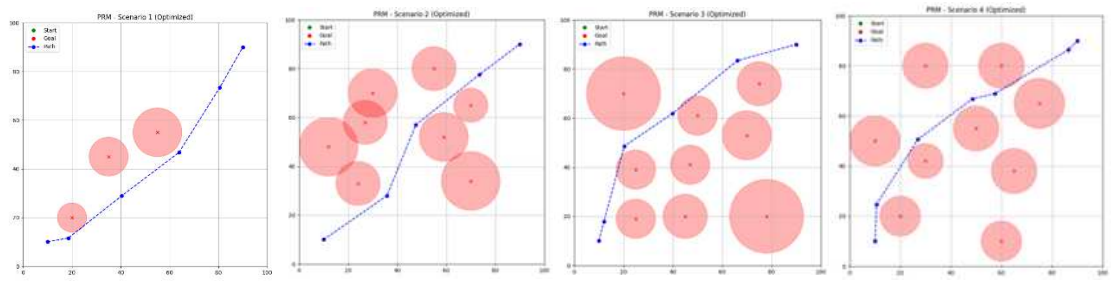


Figure 13: a.Scenario 1 b.Scenario 2 c. Scenario 3 d. Scenario 4

CASE 2:

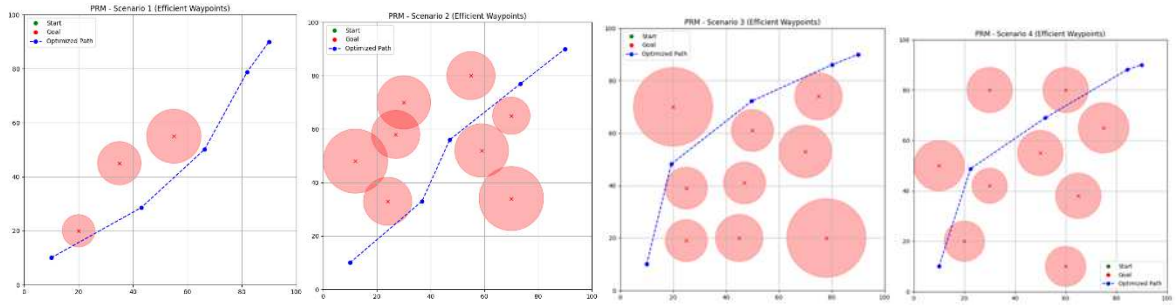


Figure 14: a.Scenario 1 b.Scenario 2 c.Scenario 3 d. Scenario 4

CASE 3:

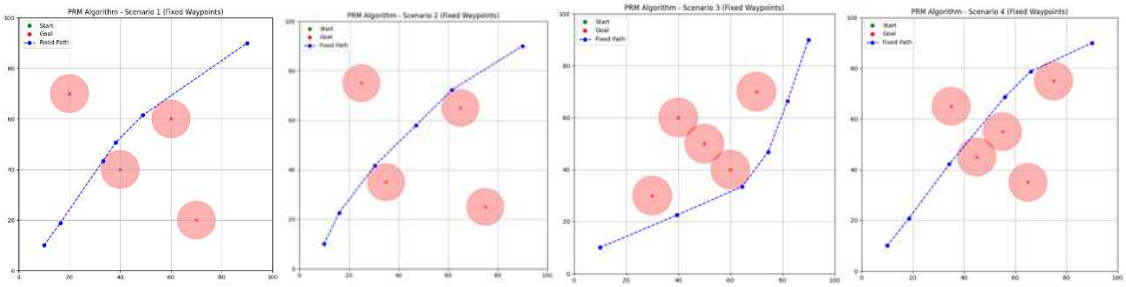


Figure 15:a. Scenario 1 b.Scenario 2 c.Scenario 3 d. Scenario 4

3.6 Dynamic Window Approach (DWA) Algorithm

The Dynamic Window Approach (DWA) is a real-time, local path planning method that focuses on the kinematic and dynamic constraints of mobile robots, making it highly suitable for swarm drone applications in cluttered environments. Unlike global planners, DWA generates a set of feasible motion trajectories by considering the drone's current velocity, acceleration limits, and obstacle proximity within a limited prediction window.

3.7 EXPERIMENTAL RESULTS:

In this section we will compare the performance of various path planning algorithms under pre-defined obstacle scenarios. The algorithms we considered are A*, Dijkstra's Algorithm, Rapidly Exploring Random Tree (RRT) Algorithm, Back and Forth Algorithm, Probabilistic

Roadmap (PRM) Algorithm and Dynamic Window Approach (DWA) Algorithm. The performance evaluation has been done on the basis of path length, time required to find the path and number of waypoints.

3.7.1 **PATH LENGTH COMPARISON:**

Path length is the primary parameter to determine the efficiency of a particular pathfinding algorithm. A shorter path length results in reduction of time to reach the target point and hence reduces computational cost. In UAVs, path length plays crucial role as a result we try to find efficient algorithm which can generate shorter path length.The unit of measurement of path length from starting point to target point is in meters (m).

The path length comparison is given in the below table.

Table 1: Total path length (in metres) obtained for each algorithm under four obstacle scenarios.

Algorithm	Cases	Scenario 1	Scenario 2	Scenario 3	Scenario 4
A *	1	123.99	116.92	125.71	119.85
	2	118.10	111.65	118.90	118.10
	3	118.10	111.65	118.90	118.49
DIJKSTRA	1	123.99	116.92	125.71	119.85
	2	118.10	111.65	119.13	117.05
	3	123.99	116.92	124.82	119.85
RRT	1	156.01	144.35	158.71	177.74
	2	129.48	119.96	124.40	131.40
	3	163.79	148.10	140.18	144.94
BaF	1	121.92	121.92	121.92	121.92
	2	118.29	120.66	120.65	118.29
	3	121.92	121.92	121.92	121.92
PRM	1	116.58	116.51	122.28	120.39
	2	116.08	115.25	122.09	120.05
	3	114.72	115.04	121.93	115.26

The result proves that A* algorithm produces shortest path continuously due to its heuristic approach, which gives priority to the ideal paths first. Dijkstra's Algorithm is reliable but it fails to optimize the path length since it explores all the possible routes. RRT Algorithm can handle complex and higher dimensional scenarios of obstacle but here it generates longer path due to random sampling behaviour. The Back and Forth Algorithm will work well in straightforward obstacle scenarios but it tends to zig-zag steps which lead to longer path length. The PRM Algorithm explores a large search space and tries to connect randomly generated nodes which mostly doesn't yield the direct route. The DWA Algorithm works well with dynamic environment and we use it primarily in real time navigation where obstacles move over time. It cannot generate proper results for 2D static obstacles.

The observation made with based on results are

1. A* is the best algorithm in terms of path length because it combines heuristic search and optimal pathfinding.
2. Dijkstra's Algorithm is less efficient as it exhaustively explores all paths without considering the direction of the goal.
3. RRT, DWA, and PRM are suitable for complex environments but tend to generate longer paths due to their exploration techniques.
4. Back-and-Forth Algorithm is the least efficient in terms of path length due to its simplistic and non-optimal path generation.

3.7.2 EXECUTION TIME COMPARISON:

Execution time is crucial parameter in real time application for UAVs. In UAVs path must be computed rapidly to ensure timely decisions. Faster algorithm leads to quicker responses which makes them more suitable for dynamic obstacle scenarios. The execution time (in seconds) defines the time taken by algorithm to find optimal path from starting point to the target point.

A* is faster than Dijkstra's Algorithm due to heuristic approach. RRT and PRM are prone to higher execution times. It is more higher when the obstacle density is high.

The execution time comparison table is given below

Table 2: Computation time (in seconds) required by each algorithm across different obstacle scenarios

Algorithm	Cases	Scenario 1	Scenario 2	Scenario 3	Scenario 4
A *	1	0.0307	0.0571	0.0864	0.0730
	2	0.0412	0.1035	0.1740	0.1217
	3	0.0436	0.1170	0.1290	0.1227
DIJKSTRA	1	0.0953	0.1765	0.1801	0.2066
	2	0.1099	0.2052	0.2270	0.2472
	3	0.0923	0.1897	0.1760	0.2246
RRT	1	0.3157	0.3370	0.2447	0.6527
	2	0.1762	0.2186	0.4503	0.3141
	3	0.0846	0.1665	0.3113	0.4344
BaF	1	0.0052	0.0073	0.0071	0.0076
	2	0.0049	0.0045	0.0044	0.0049
	3	0.0027	0.0067	0.0065	0.0052
PRM	1	0.7438	1.1291	1.1304	1.2238
	2	0.6896	1.0593	1.1124	1.1128
	3	8.3303	8.1205	8.7985	8.7645

The observation made is that A* is the fastest algorithm due to heuristic approach. It converges quickly in environment which has clear path especially low-density obstacle scenario.

Dijkstra's algorithm gives an optimal path but takes more time to find path in comparison with A*. RRT algorithm is prone to have higher execution time as it has random sampling behaviour. It results in high computational costs specifically in high dense obstacle scenarios. PRM algorithm performs good relatively in less dense obstacle environment than highly dense obstacle scenario. Back and Forth algorithm is simple and fast in computing the path as it combines the forward and Backward path to get the optimal path. But it is not much efficient in path planning in longer and optimal path.

The observation made based on results are

1. A* is the fastest algorithm in terms of execution time, especially in less complex environments, due to its efficient heuristic search.
2. Dijkstra's Algorithm takes longer because it exhaustively explores all possible paths.
3. RRT and DWA have longer execution times due to their random sampling and real-time adjustment, respectively.
4. PRM's execution time varies but can become quite long in densely obstacle-packed environments.
5. Back-and-Forth Algorithm has the shortest execution time but produces inefficient paths.

3.7.3 WAYPOINT COUNT COMPARISON:

The Algorithm can navigate through the obstacle scenarios but due to obstacle there are some decision-making points (waypoints) to navigate the UAV to the target. Lower waypoint count signifies that the path is optimal and there are very less corrections during navigation. It leads to smooth path and helps to minimize the complexity of decision making in UAVs.

The table below gives the waypoint count comparison.

Table 3: Number of decision points (waypoints) used by each algorithm to reach the goal in various scenarios.

Algorithm	Cases	Scenario 1	Scenario 2	Scenario 3	Scenario 4
A *	1	20	19	22	20
	2	4	4	4	5
	3	4	4	4	5
DIJKSTRA	1	20	19	22	20
	2	4	4	4	5
	3	20	19	20	20
RRT	1	33	30	33	37
	2	4	4	6	4
	3	29	29	29	29
BaF	1	20	20	20	20
	2	8	8	8	8
	3	20	20	20	20
PRM	1	6	5	6	7

	2	5	5	5	5
	3	6	6	6	6

The above table states that A* has fewest waypoints because it uses heuristic approach. Dijkstra's algorithm has higher number of waypoints as it explores all the routes. RRT produces higher waypoint due to the random sampling behaviour. PRM produces lesser waypoints than RRT but may still have more than A *. Back and Forth produces higher waypoints due to zig-zag nature.

In case 1 A* and Dijkstra's gives stable, moderately dense paths. RRT is highly random with unnecessary waypoints. BaF is consistent but rigid. PRM is shortcut-based but risky in complex environments.

In case 2 A*, Dijkstra, PRM achieve minimal and consistent waypoints. RRT is slightly less stable but mostly optimized. BaF needs more waypoints due to its structure.

In case 3 A* is best suited for minimal waypoints even under fixed constraint. Dijkstra struggles due to lack of heuristics. RRT still produces many unnecessary nodes. BaF is rigid and less adaptive. PRM handles waypoint fixing moderately well.

The following observations could be made:

1. A* produces the fewest waypoints due to its heuristic approach, which avoids unnecessary detours.
2. Dijkstra's Algorithm has more waypoints due to exhaustive exploration.
3. RRT and DWA have higher waypoint counts because of their random sampling and real-time adjustments.
4. PRM strikes a balance but still generates more waypoints than A*.
5. Back-and-Forth Algorithm has the highest waypoint count due to inefficient navigation patterns.

3.8 CONCLUSION

This research presented a detailed analysis of several path planning algorithms, namely A*, Dijkstra's algorithm, Rapidly-exploring Random Tree (RRT), Back-and-Forth (BaF), and Probabilistic Roadmap (PRM), across a range of obstacle-rich scenarios with varying degrees of complexity. The evaluation considered multiple cases focused on waypoint optimization, path efficiency, and responsiveness to obstacle distributions.

Among the tested methods, A* consistently delivered the most optimal results, producing shorter and more direct paths due to its heuristic-driven approach. In contrast, Dijkstra's algorithm, although reliable in finding feasible paths, tended to generate less efficient routes because of its exhaustive search strategy without heuristic support. RRT, while effective for navigating complex spaces, resulted in paths with a higher number of waypoints, reflecting its random nature. The Back-and-Forth method maintained uniform performance but lacked adaptability in dynamically changing environments. PRM offered moderate success but

showed sensitivity to obstacle density and required careful parameter tuning for satisfactory results.

Overall, the findings confirm that heuristic-based strategies, particularly A*, are better suited for efficient and consistent path planning in static environments. The results obtained through simulations matched theoretical expectations, demonstrating the validity and accuracy of the implementations. Future investigations could explore combining different algorithmic strengths to improve adaptability, efficiency, and real-time performance in more unpredictable or dynamic settings.

DECLARATION STATEMENTS:

Funding: This research did not receive any specific funding.
Data Availability Statement: Not applicable.

REFERENCES

[1] Y. Yue, X. Tang, "Predictive A* Path Planning Algorithm Based on Dynamic Obstacles," *Journal of Intelligent & Robotic Systems*, 2020.

[2] L. Han, M. Li, "Accelerated A* Algorithm with Hash Table Optimization for Pathfinding," *Sensors*, 2019.

[3] W. Liu, Y. Wang, "Fast Start-Up A* Algorithm for Real-Time UAV Navigation," *International Conference on Robotics and Automation*, 2021.

[4] D. Mandoi, F. Salvetti, "Comparative Analysis of Path Planning Algorithms: A*, Dijkstra, and RRT," *Applied Sciences*, 2020.

[5] P. Ma, Q. Li, "Heuristic Function Adaptation in A* Path Planning for Dynamic Environments," *IEEE Transactions on Intelligent Vehicles*, 2022.

[6] D. Karve, "Shortest Path Analysis for Turbulent Environments using Dijkstra's Algorithm," *Journal of Aerospace Information Systems*, 2018.

[7] E. Dhulkefl, "Extension of Dijkstra Algorithm to 3D Environments for UAV Navigation," *Aerospace Science and Technology*, 2020.

[8] R. Zhao, Y. Chen, "Matrix Alignment Techniques to Speed Up Dijkstra-Based Trajectory Planning," *IEEE Access*, 2020.

[9] A. K. Gupta, R. Saini, "Dijkstra-PSO Hybrid Algorithm for Optimized Path Planning," *Procedia Computer Science*, 2021.

[10] Y. Sun, "Multi-angle Scanning Path Planning for Efficient UAV Routing," *Sensors*, 2021.

[11] Q. Li, "Optimized Sampling Strategy for Probabilistic Roadmaps," *IEEE Robotics and Automation Letters*, 2019.

[12] F. R. Bana, "Monte Carlo-Enhanced Risk-Aware Probabilistic Roadmaps," *Autonomous Robots*, 2021.

[13] H. Zhao, Y. Xu, "Dimensionality Reduction for PRM-based UAV Navigation Indoors," *Robotics and Autonomous Systems*, 2022.

[14] G. Li, "Optimized Node Connection Strategies in Probabilistic Roadmap Method," *IEEE Transactions on Robotics*, 2020.

[15] S. Aviram, "Dynamic Boundary Adjustment in PRM Planning," *Robotics and Autonomous Systems*, 2021.

[16] F. Yang, "Ant Colony Optimization-Enhanced RRT for Rapid Path Planning," *Applied Intelligence*, 2021.

[17] J. Chen, "Two-Node Growth Strategy for Accelerated RRT Expansion," *IEEE Access*, 2022.

[18] M. Patel, S. Chhabra, "Redundant Sampling Reduction in RRT," *International Journal of Robotics Research*, 2020.

[19] V. Kumar, M. Singh, "Performance Analysis of RRT Variants in Dynamic Environments," *Autonomous Robots*, 2022.

[20] J. Fang, "Path Optimization in RRT via Node Elimination and Smoothing," *IEEE Transactions on Cybernetics*, 2021.

[21] L. Jintao, "Efficient Coverage Path Planning with Back-and-Forth Patterns," *Robotics and Autonomous Systems*, 2019.

[22] Young-Jo Lee, "Application of Back-and-Forth Coverage in Agricultural UAVs," *Computers and Electronics in Agriculture*, 2021.

[23] Yuxin Wang, "Adaptive Sweep Planning for Moving Obstacle Environments," *IEEE Robotics and Automation Letters*, 2022.

[24] M. Ortiz, "Integrated Obstacle Avoidance in Sweep Coverage Strategies," *Journal of Field Robotics*, 2020.

[25] D. Fox, W. Burgard, S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robotics and Automation Magazine*, 1997.

[26] L. Zhang, "Improved Velocity Sampling Techniques in DWA," *IEEE Transactions on Robotics*, 2020.

[27] X. Wu, "Predictive DWA for Moving Obstacle Avoidance," *IEEE Access*, 2021.

[28] J. Jang, "Cost Function Optimization in DWA for Smooth Path Planning," *Autonomous Robots*, 2022.

[29] S. Chen, "Dynamic Window Approach Adaptation for High-Speed UAV Navigation," *Sensors*, 2021.