Journal of Autonomous Intelligence (2024) Volume 7 Issue 2
doi: 10.32629/jai.v7i2.1284

ORIGINAL RESEARCH ARTICLE

Efficiency analysis of path-finding algorithms in a 2D grid environment

Ch Nirmal Prabhath[1], M. Kavitha[1], Kanak Kalita[2,3,*]

[1] *Department of Computer Science Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi 600062, India*

[2] *Department of Mechanical Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi 600062, India*

[3] *University Centre for Research & Development, Chandigarh University, Mohali 140413, India*

*** Corresponding author:** Kanak Kalita, drkanakkalita@veltech.edu.in, kanakkalita02@gmail.com

ABSTRACT

This paper offers a focused overview of pathfinding algorithms, particularly emphasizing Greedy Best First Search (G-BFS) and Rapidly-Exploring Random Trees (RRT). Their performance is evaluated within a 2D grid setting tailored for Unmanned Aerial Vehicles (UAVs). Divided into two main sections, the study first expounds on the theoretical underpinnings of these algorithms, followed by empirical validation. A series of systematic experiments, involving varied 2D grid dimensions and traversal patterns, facilitates a comparative analysis between G-BFS and RRT. Importantly, the real-world implementation of these algorithms in UAV navigation underscores their practicality, illuminating their respective execution times and resource utilization. While G-BFS thrives in straightforward scenarios, RRT, especially RRT*, displays superior capability in navigating more intricate and expansive terrains, albeit with marginally extended execution durations attributed to its explorative nature.

Keywords: RRT; G-BFS; path-finding problem; 2D grid; UAV

ARTICLE INFO

Received: 24 September 2023
Accepted: 14 November 2023
Available online: 21 December 2023

1. Introduction

Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, have burgeoned as multifunctional tools with a plethora of applications, spanning from surveillance and reconnaissance to package delivery and environmental monitoring. Such aerial platforms hold the promise of navigating intricate environments both autonomously and efficiently, rendering pathfinding an integral facet of their functionality. When visualizing UAVs maneuvering in a 2D grid, pathfinding algorithms become indispensable in ascertaining optimal routes that bypass obstructions while conforming to specific constraints[1]. Pathfinding in 2D grids for UAVs amalgamates principles from computer science, mathematics and engineering[2], aiming to architect methods for UAVs to chart their courses astutely, ensuring a seamless and efficient passage through a grid-based milieu. This milieu typically mirrors a 2D grid, where individual cells might be navigable or non-navigable, signifying open airspace or impediments, respectively[3].

Pathfinding algorithms find their relevance in an expansive array of domains, including real-world autonomous navigation mechanisms, simulated digital environments and the diverse terrains of gaming.

However, the challenges intrinsic to pathfinding, especially for UAVs, stand distinct. Unlike their terrestrial counterparts or vehicles directed by humans, UAVs have the capability to soar over obstructions. Nevertheless, their movement is tethered to constraints encompassing limited energy reserves, physical confines and regulations governing altitude and airspace[4]. The ever-evolving nature of environmental factors, such as shifting weather patterns and unanticipated barriers, further convolutes the pathfinding equation[5].

In response to these complexities, a myriad of pathfinding algorithms, fine-tuned specifically for UAVs navigating 2D grids, have been conceived. Their designs target the optimization of trajectories, keeping pivotal parameters like fuel economy, mission objectives and real-time barrier updates at the forefront. Noteworthy algorithms in this realm include A* (A-star), Dijkstra's algorithm, Greedy Best First Search (G-BFS) and the Rapidly-Exploring Random Tree (RRT) algorithm. An ideal pathfinding algorithm merges efficiency with cost-effectiveness and intuitiveness[6]. Both the G-BFS and RRT algorithms, encapsulated within the 2D grid milieu, emulate decisions a human might instinctively make when discerning a practical and frequently trodden path from start to endpoint[7].

The foundational objective of this paper is to navigate the nuanced intricacies influencing the efficacy of the G-BFS and RRT algorithms. By doing so, we aspire to amplify the comprehension of these algorithmic behaviours and proficiencies within a 2D grid framework, bolstering their applicability in myriad real-world UAV scenarios.

Our methodology fosters an exhaustive juxtaposition of the G-BFS and RRT algorithms, centered on their prowess in pinpointing the crux of pathways within a 2D grid dotted with challenges. By infusing diverse grid dimensions and tabulating means across multiple instances, we venture to furnish a steadfast appraisal of these algorithms' performance metrics. This evaluation enriches the discourse on their inherent advantages and potential bottlenecks. At the heart of our analysis lies an acute recognition of the symbiotic relationship between the grid's scale and the algorithms' node traversal strategies[8]. As these algorithms embark on their node-centric journeys, they leave an indelible mark on pivotal performance indicators, such as execution speed and resource commitment[9]. This intricate dance substantially dictates the algorithms' prowess in unveiling the optimal routes. To reinforce the authenticity of our insights, we commit to a rigorous battery of iterative tests, establishing both rigor and replicability. This empirical foray begets a vast data repository, forming the bedrock for our profound inferences on these algorithms' performance nuances. In its entirety, this study plunges deep into the realm of pathfinding, marrying the theoretical frameworks with the pragmatic challenges of maneuvering amidst hurdles.

2. Literature-pathfinding algorithms

Efficient navigation remains an intricate issue across domains, from real-world navigation systems to immersive computer-generated simulations[10]. At the core of these challenges, pathfinding algorithms stand out as pivotal tools. They not only underpin self-driving vehicles and robots but also enhance players' experiences in dynamic gaming environments[11]. Particularly for UAVs, the significance of dynamic environments elevates the importance of algorithms like G-BFS and RRT[12].

Dijkstra's algorithm, while reliable, struggles with large search scopes due to time constraints, prompting authors[13] to introduce a time-sensitive variant better suited for real-time applications like car navigation. This innovative approach attempts to approximate the optimal path swiftly but demands higher computational resources, a potential drawback for UAV navigation.

A* has established its prominence in pathfinding, especially in gaming contexts, due to its optimality. However, its performance can be hindered by heuristic choice and its memory usage in vast search areas might limit its application in resource-limited scenarios like UAVs. Research into path planning for mobile robots, working on reactive navigation and SLAM, has seen A* variants emphasizing computational efficiency and path optimality[14]. But the primary focus remains on algorithm performance rather than intricate aspects of robot navigation. A study by Heusner et al.[15] delves deep into GBFS search behaviors, introducing "high-water mark benches" to understand state expansion patterns in the search space. Similarly, Tripathy et al.[16] studied GBFS's behavior using this concept, revealing nuances in state expansions.

The growing significance of UAVs in navigation is highlighted by Fu et al.[5], who accentuate the differentiation between graph-based and sampling-based path planning algorithms. Their work emphasizes the speed and efficiency of random sampling-based methods in complex scenarios. This focus on efficiency in path planning extends to the RRT algorithm, which is explored in the context of wheeled mobile robots[17]. A novel approach combining artificial-guided points with heuristic search algorithms promises enhanced navigation in complex terrains.

A unique blend of reinforcement learning with path graphs is introduced by Liu et al.[12], targeting enhanced robot navigation. This approach aims to surpass traditional BFS and RRT algorithms in generating smoother paths. Meanwhile, a study by Navya and Ranjith[18] pits BFS against DFS in the context of medical robots in a hospital ward, concluding BFS's superiority in this specific application. Li et al.[19] proposes a new path planning method using visibility graphs (v-graphs) to overcome the challenges of occupancy grid-based planning. Despite its efficiency, especially in intricate environments, its primary application focuses on mobile robot navigation, with potential limitations in extremely convoluted terrains.

3. Methodology

The research methodology comprises a comparative evaluation of G-BFS and RRT pathfinding algorithms. The core focus is on determining the path between two random nodes in a 2D grid space, using execution time and node traversal as key performance indicators.

3.1. Experiment space

A 2D grid described in the study of Yuanhao et al.[20] serves as the experimental backdrop. The grid integrates a source, a destination and obstacles making up roughly 20% of the grid space. These obstacles simulate real-world navigation challenges where the pathfinding task involves circumventing barriers[21]. Experiments are performed on a computer with a 2.40 GHz Intel CPU. The algorithms are coded in Python 2.8. To gauge the efficiency across different complexities, the experiment spans multiple grid sizes: 64×64, 128×128, 256×256, 512×512 and 1024×1024. Each grid dimension is tested using an average of 10 samples, mitigating anomalies from random obstacle and node placements. The algorithms operate under a limitation of strictly horizontal or vertical movements. Moving between two neighbouring points in the grid comes at a standardized cost of 1, streamlining the efficiency analysis.

3.2. Greedy Best First Search (G-BFS) algorithm

G-BFS is deeply entrenched in the evolution of heuristic search algorithms. It's foundation lay in the invention of Dijkstra's algorithm by Edsger W. Dijkstra, aiming for the shortest path computation between nodes[16]. Dijkstra's algorithm, while ground-breaking, lacked heuristic guidance. This gap was filled by A*, an innovative algorithm introduced by Peter Hart, Nils Nilsson and Bertram Raphael. Merging Dijkstra's precision with heuristic finesse, A* offered an intelligent, optimal pathfinding method. From this backdrop, G-BFS surfaced as a heuristic-focused variant of A*, characterized by its "greedy" approach: consistently selecting the node that, by heuristic measures, seems closest to the goal[22,23]. This made G-BFS a go-to choice in situations demanding quick results, even if the resulting path wasn't always optimal. Today, it is invaluable in realms like robotics, gaming and GPS[24].

G-BFS uses a heuristic function to estimate costs from the current node to the goal. This estimation makes it "greedy"[8]. G-BFS may be efficient but doesn't always guarantee the shortest path. It can sometimes overlook shorter routes if they are not in alignment with the heuristic prediction. G-BFS is a staple in applications where a near-optimal solution is acceptable, but swift computation is essential.

G-BFS belongs to the informed search category. It assigns an evaluation function, $f(n)$, to each node. This function represents the total estimated cost from the current node to the destination, with nodes expanded based on the smallest function value. The heuristic or evaluation function typically used in G-BFS is the Manhattan distance:

$$f(n) = |y2 - y1| + |x2 - x1| \tag{1}$$

where $x1, y1$ are coordinates of the current node $x2, y2$ are the destination's coordinates. The algorithm evaluates the Manhattan distance for neighboring nodes and expands to whichever appears closest to the goal. Despite its efficiency, it is not always optimal and in certain contexts, the algorithm may get stuck in infinite loops. In terms of time complexity, it operates at $O(2n)$, marking significant progress in pathfinding algorithms.

For this study, G-BFS (Algorithm 1) is implemented in a 2D grid, treating each grid point as a node initially set to infinity. Obstacles are denoted by "1" and paths by "2". The algorithm initiates at the source, estimating heuristic values for each neighbouring node, selecting nodes based on the lowest value and bypassing obstacles during evaluation.

Algorithm 1 G-BFS

```
1:   Declare an $M \times N$ integer array with an initial value of $\infty$
2:   Initialize Source and Destination nodes with the value 2
3:   Mark obstacles in the array with the value 3
4:   Declare $cur\_vari$ and set it to Source
5:   while $cur\_vari \neq$ Destination:
6:   for each neighbouring node:
7:   Compute the distance from the node to the Destination
8:   Pick $new\_vari$ with the smallest distance
9:   Set $new\_vari$ to 2 (indicating a path)
10:  Update $cur\_vari$ to $new\_vari$
11:  end for
12:  end while
13:  visually represent the array.
```

3.3. Rapidly-Exploring Random Trees (RRT) algorithm

The history of Rapidly-Exploring Random Trees (RRT) is fundamentally linked to the evolution of robotics and autonomous systems. Steven M. LaValle's introduction of RRT in the 1990s marked a notable shift in navigation algorithms, emphasizing efficient exploration of high-dimensional spaces[25]. Unlike conventional deterministic models, RRT's randomized nature allowed for rapid sampling of random points in the search space and extending a tree towards them. This method provided a unique edge in traversing intricate environments, most prominently seen in robotics. Today, with adaptations like RRT*, the algorithm's variations are central to robotics, autonomous vehicles and other dynamic settings[26].

At its core, RRT is random. It grows a tree by regularly sampling points from the search space and inching the tree towards them[27]. RRT excels in obstacle-rich settings and complex architectures. It navigates challenging terrains without demanding significant memory. Given its versatility, RRT finds use in robotics, autonomous vehicles and other unpredictable environments.

While often dubbed as a "brute-force" approach, RRT's essence lies in its ability to tackle pathfinding issues in challenging environments. The algorithm banks on an iterative tree-building mechanism, commencing from an arbitrary point and systematically growing the tree by adding new random samples. RRT's uniqueness rests in its preference for uncharted regions, facilitating rapid exploration into newer territories[28].

The method consistently picks a new random point, identifies the nearest node in the current tree and then connects them. This random exploration enables RRT to handle intricate environments and identify non-obvious optimal paths[29]. Both leaf nodes and parent nodes are stochastically chosen, offering RRT its agility in traversing terrains laden with obstacles.

Once the RRT tree takes shape, generating paths between specific start and end points becomes feasible. By integrating these points with the tree, the algorithm maps a node sequence, detailing the path from the onset to the conclusion. This inherent adaptability shines especially in obstacle scenarios, where RRT instinctively plots routes around these barriers.

A distinctive RRT advantage is its recall ability. By retaining the tree structure, the algorithm can be repurposed for subsequent pathfinding missions. This reuse principle amplifies RRT's efficiency and speed in familiar terrains. RRT harnesses randomness and incremental growth to proficiently dissect complex spaces, unveiling feasible routes in demanding scenarios. Its adaptability, combined with the probabilistic roadmap technique, underpins its formidable reputation in diverse navigation problems[30]. The algorithmic impelementation of RRT in this work is shown in Algorithm 2.

Algorithm 2 RRT

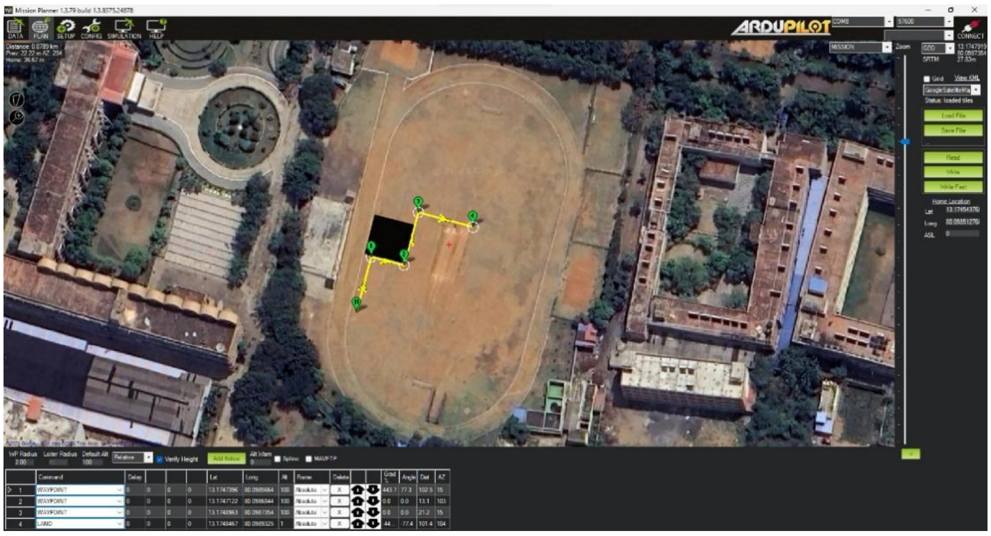
```
1:  Declare an M × N integer array initialized to 0
2:  Mark Source and Destination as 2
3:  Initialize obstacles with 1
4:  Initialize $cur\_vari$ with Source
5:  Construct a tree structure, $tree\_vari$
6:  Set $tree\_vari$ parent node as Source
7:  While $cur\_vari \neq$ Destination:
8:  Randomly pick a node, assign to $new\_vari$
9:  If $new\_vari$ isn't an obstacle:
10: Check if the line between $new\_vari$ and $cur\_vari$ doesn't intersect any obstacle:
11: Incorporate $new\_vari$ into $tree\_vari$, with $cur\_vari$ as its parent
12: Update $cur\_vari$ to $new\_vari$
13: End if
14: End if
15: Trace the path between parent and the latest child
16: Update the array with 2
17: End while
18: Render the array visually
```

3.4. Experiemntal implementation

In the fascinating realm of autonomous aerial vehicles, the implementation of advanced algorithms for path planning becomes a pivotal aspect. It not only ensures the efficiency of these vehicles but also their safety, especially when navigating through obstacle-rich environments. Our experiment with a quadcopter equipped with a Pixhawk 4 flight controller offers a comprehensive analysis in this domain, focusing on two prominent algorithms: RRT and G-BFS.

The quadcopter, powered by the Pixhawk 4, served as the experimental platform. Both RRT and G-BFS algorithms were embedded into its onboard computer, designed to churn out a sequence of waypoints. These waypoints, essential for guiding the drone's trajectory, were communicated to the mission planner software, offering a visual representation (Refer to **Figure 1** for G-BFS and **Figure 2** for RRT). These figures showcased an aerial perspective of the institute's playground, indicating the path formulated by each algorithm. The testing grounds were carefully chosen – an open space within the academic establishment, its perimeters precisely demarcated using GPS coordinates. This controlled yet realistic setting offered the right balance for an unbiased assessment of each algorithm.

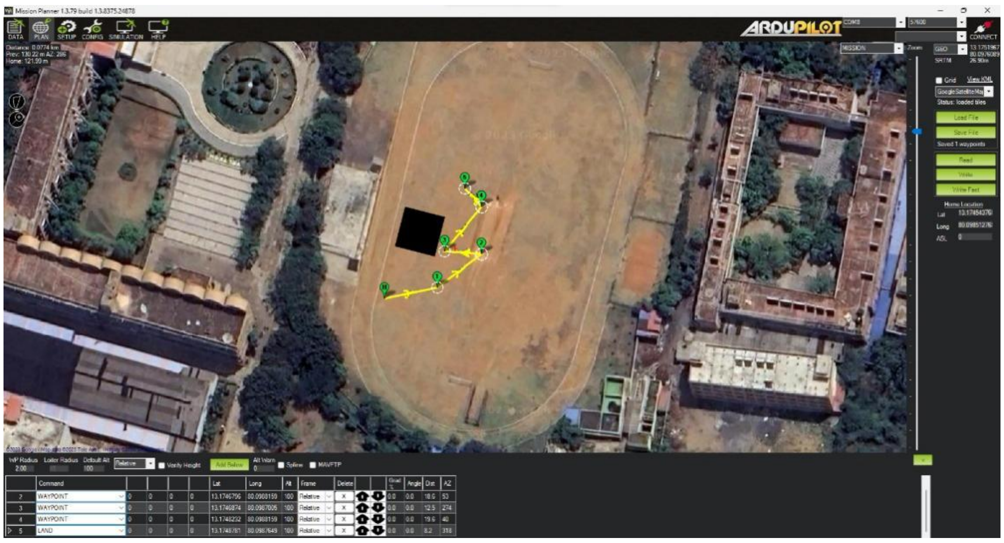
To ensure a comprehensive evaluation, each algorithm underwent ten unique tests. This repetitive approach aimed to factor in all possible anomalies and ensure a thorough understanding of the algorithms' performance. Realism was at the forefront of this experiment. To emulate genuine challenges a drone might face, imaginary static obstacles were deliberately incorporated into the drone's trajectory. These obstacles served a dual purpose – they not only tested the algorithms' robustness but also their proficiency in real-world obstacle avoidance.



WP Radius 2.00 | Loiter Radius | Default Alt 100 | Relative | ✓ Verify Height | Add Below | Alt Warn 0 | Spline | MAVFTP

	Command	Delay				Lat	Long	Alt	Frame	Delete			Grad %	Angle	Dist	AZ
1	WAYPOINT	0	0	0	0	13.1747396	80.0985664	100	Absolute	X			443.7	77.3	102.5	15
2	WAYPOINT	0	0	0	0	13.1747122	80.0986844	100	Absolute	X			0.0	0.0	13.1	103
3	WAYPOINT	0	0	0	0	13.1748963	80.0987354	100	Absolute	X			0.0	0.0	21.2	15
4	LAND	0	0	0	0	13.1748467	80.0989325	1	Absolute	X			-44..	-77.4	101.4	104

Figure 1. Induing G-BFS algorithm in Mission Planner software for obstacle avoidance.



WP Radius 2.00 | Loiter Radius | Default Alt 100 | Relative | Verify Height | Add Below | Alt Warn 0 | Spline | MAVFTP

	Command					Lat	Long	Alt	Frame	Delete			Grad %	Angle	Dist	AZ
2	WAYPOINT	0	0	0	0	13.1746796	80.0988159	100	Relative	X			0.0	0.0	18.6	53
3	WAYPOINT	0	0	0	0	13.1746874	80.0987005	100	Relative	X			0.0	0.0	12.5	274
4	WAYPOINT	0	0	0	0	13.1748232	80.0988159	100	Relative	X			0.0	0.0	19.6	40
5	LAND	0	0	0	0	13.1746781	80.0987649	100	Relative	X			0.0	0.0	8.2	318

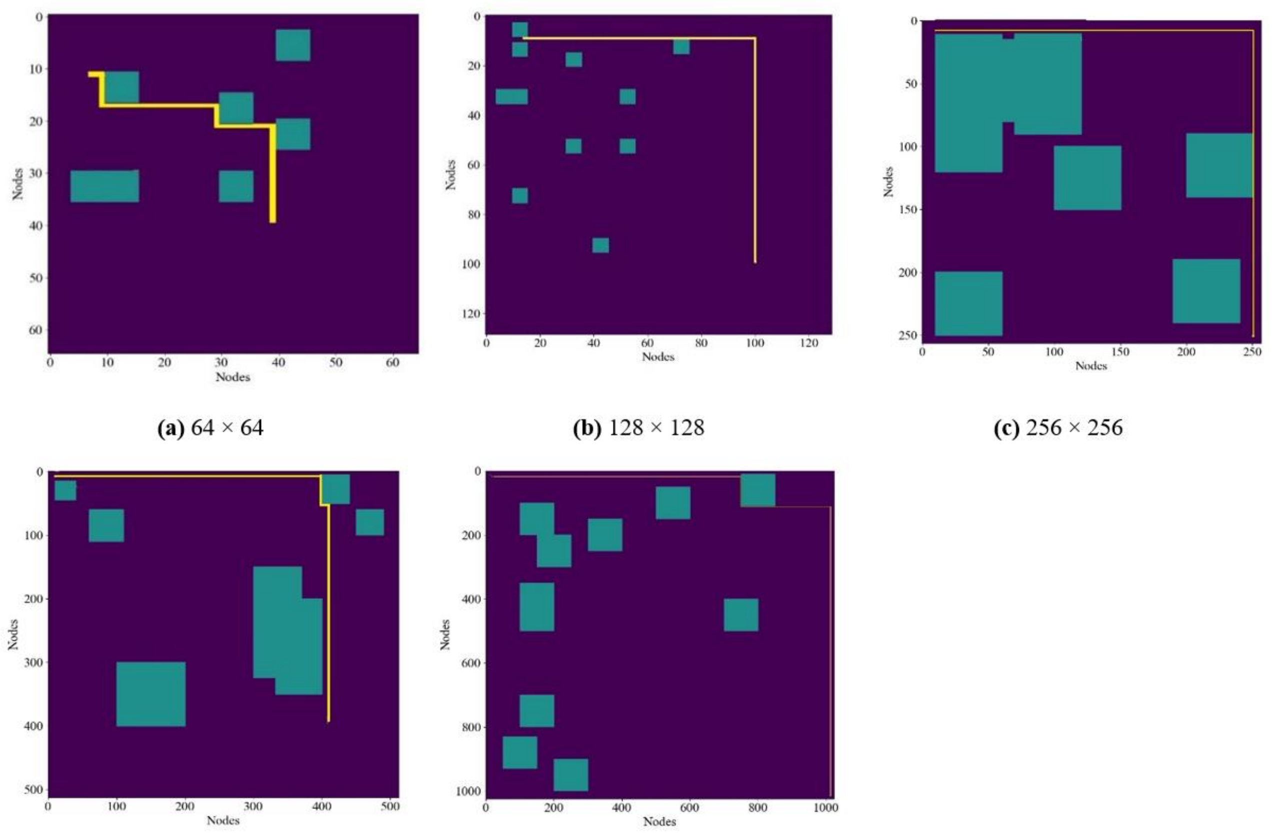
Figure 2. Induing RRT algorithm in Mission Planner software for obstacle avoidance.

4. Results and discussion

Visual representation is a potent means of presenting complex data in an easy-to-digest manner. In the domain of path-finding algorithms, this becomes especially pertinent given the inherently spatial nature of their task. The use of Heat-Map graphs in our study offered a brilliant way to encapsulate the nuances of the G-BFS and the Rapidly-Exploring Random Trees (RRT) algorithms' performance.

As depicted in **Figures 3** and **4**, the yellow dot holds an essential position, marking the starting point for the algorithm's expedition. Acting as the launching pad, it signifies the origin from where the computation of the optimal path begins. The transition from the starting point to the desired destination is beautifully illustrated either by the light-green line in **Figure 3** or the sequence of dots in **Figure 4**. This demarcation symbolizes the calculated trajectory by G-BFS and RRT, providing an immediate visualization of their operational paths. The real challenge for any path-finding algorithm is not merely connecting two points but doing so in an environment filled with obstacles. The blue boxes in the visuals offer a clear picture of these hurdles. By placing these obstacles, the complexity of the task becomes apparent and the proficiency of the algorithms in navigating around them gets accentuated.

Perhaps the most potent representation in the visualization is the dark blue region. This area, free from any markings, effectively portrays the discovered path by the algorithms. It underscores the capability of both G-BFS and RRT to not just avoid obstacles but to do so in a manner that is efficient and optimal. A unique facet of our visual representation was the incremental scaling of the 2D grid. The images in **Figure 3**, derived from the G-BFS, display this scalability. By showcasing results from grids of sizes 64 × 64 to an extensive 1024 × 1024, we demonstrate the adaptability and scalability of the algorithm. Such scaling gives insights into the algorithms' performance across varying complexities, ensuring that our findings aren't just confined to a specific scenario but have broader applicability.

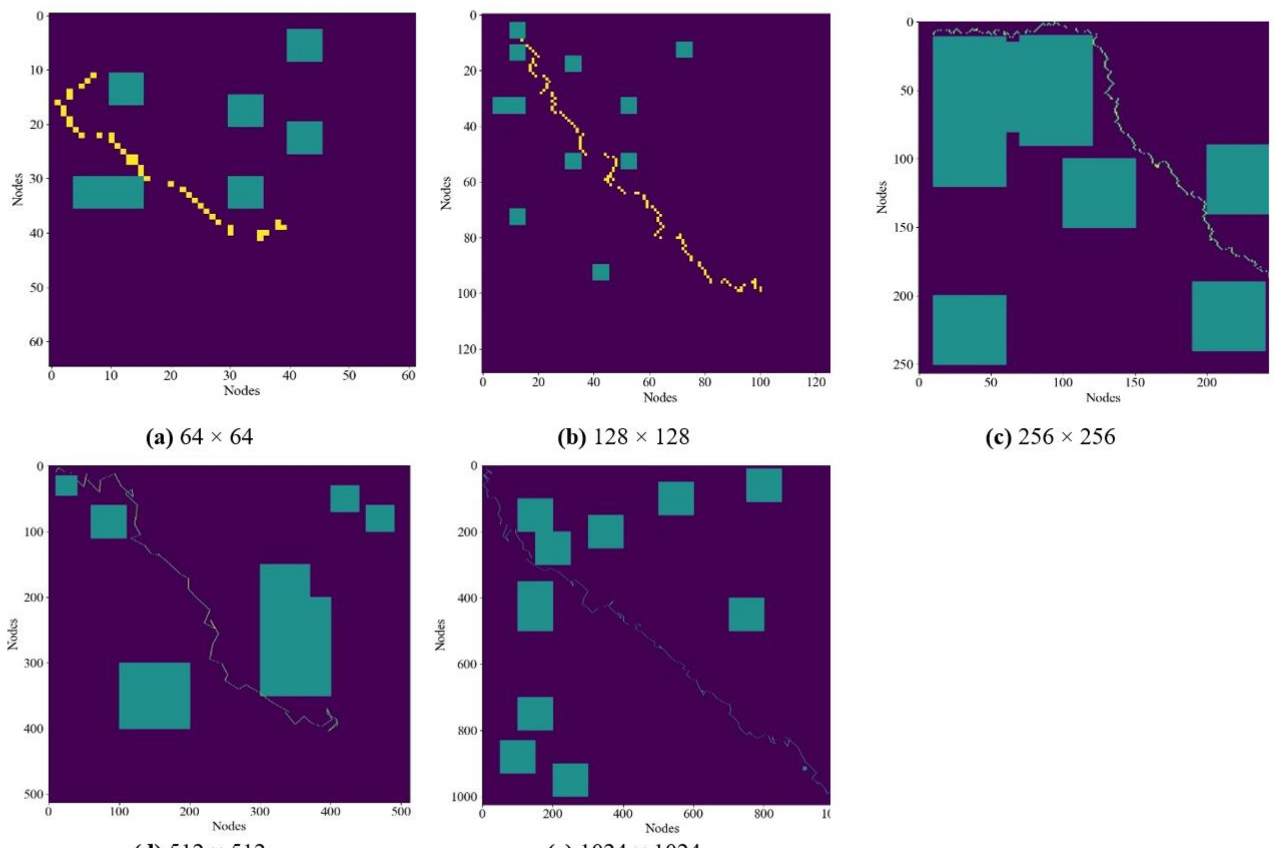


(a) 64 × 64 **(b)** 128 × 128 **(c)** 256 × 256

(d) 512 × 512 **(e)** 1024 × 1024

Figure 3. Greedy Best First Search algorithm **(a)** 64 × 64 **(b)** 128 × 128 **(c)** 256 × 256 **(d)** 512 × 512 **(e)** 1024 × 1024 2D grid layout.

The Heat-Map representations in **Figure 4** capture the intricate workings of the Rapidly-Exploring Random Trees (RRT) algorithm. Each screenshot represents the algorithm's performance over different grid scales, from 64 × 64 to 1024 × 1024. The evolution of the algorithm's behaviour with increasing grid complexity offers valuable insights into its scalability and adaptability.



(a) 64 × 64 **(b)** 128 × 128 **(c)** 256 × 256 **(d)** 512 × 512 **(e)** 1024 × 1024

Figure 4. Rapidly-exploring Random Tree algorithm **(a)** 64 × 64 **(b)** 128 × 128 **(c)** 256 × 256 **(d)** 512 × 512 **(e)** 1024 × 1024 2D grid layout.

4.1. Execution time

4.1.1. Simulation

The G-BFS is an informed search algorithm, but its behaviour can sometimes be likened to brute force in certain scenarios. From **Table 1**, it is evident that the execution time for G-BFS tends to grow, not strictly linearly, but more prominently with increasing grid size. For instance, while the execution time for a 512 × 512 grid layout is approximately 0.04686 seconds, it more than doubles when the grid size is scaled up to 1024 × 1024, clocking in at roughly 0.11947 seconds. Despite the exponential growth in computational time, its speed on larger grids suggests potential suitability for real-world applications where immediate pathfinding solutions are required.

Table 1. Execution time(s) in various 2D grid layout.

Algorithm	**64 × 64**	**128 × 128**	**256 × 256**	**512 × 512**	**1024 × 1024**
G-BFS	0.01562	0.0313	0.03952	0.0486	0.1194
RRT	0.39042	2.7504	14.7392	0.1405	0.5255

In contrast, the RRT algorithm, known for its randomized exploration behaviour, displayed inferior performance in comparison to the G-BFS algorithm, at least in the selected 2D grid scenarios. Its computational demands are higher and its execution time lacks the predictability observed with G-BFS. The inconsistency in

RRT's time performance across different grid sizes underscores the unpredictability brought about by its randomized nature. The data underscores the complexity inherent in path-finding algorithms and their sensitivities to various environmental factors. It is not just about the size of the grid but also about how the algorithms interact with the complexities within those grids.

From a real-world application standpoint, understanding these nuances is critical. If the goal is real-world pathfinding in large environments, G-BFS seems to have an edge, especially in grid scenarios. However, the increased computational demand of RRT, despite its slower speeds, might offer other advantages in more dynamic or unpredictable environments, where exploration can be more valuable than immediate optimization.

While G-BFS displays superior performance in terms of speed across all grid sizes, the relationship between grid size and execution time isn't strictly linear. The complexities involved in these algorithms necessitate careful consideration and optimization for real-world applications, balancing speed, accuracy and computational resources.

4.1.2. Experiment

When algorithms transition from the simulation phase to real-world application, their true capabilities are put to the test. In the context of the G-BFS algorithm, its real-world deployment within a quadcopter's navigation system provided a tangible demonstration of its merits. In the controlled environment of the institute's grounds, with the visual aid from **Figures 5** and **6**, it is evident that G-BFS was able to consistently deliver on its promise. The institute's grounds, armed with accurate GPS coordinates and an open-air setting, acted as a true representation of the challenges that autonomous aerial vehicles face in everyday scenarios. This environment was not just a sterile testing ground—it was a dynamic space where the algorithm had to contend with real-world variables.

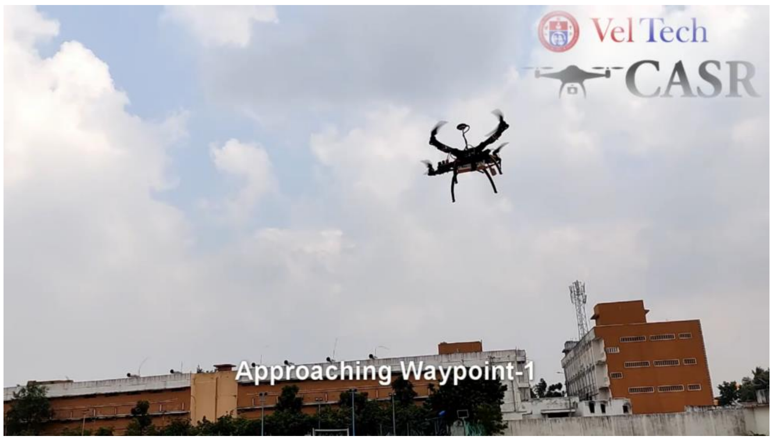


Figure 5. Samples of UAV flying between the designated source and destination.

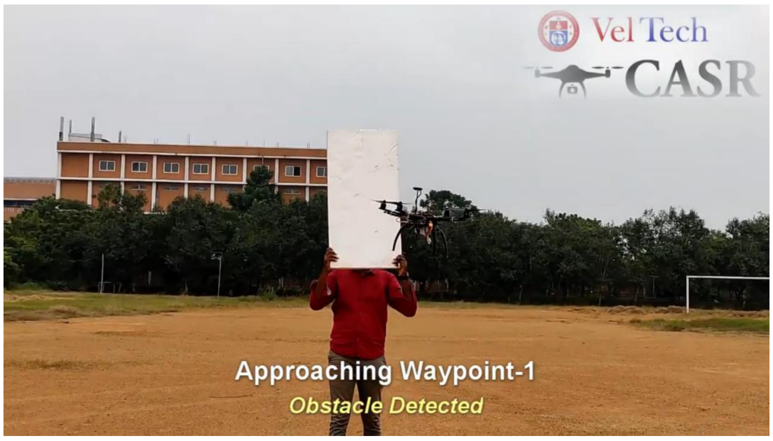


Figure 6. Manual Inducing of obstacles.

Perhaps one of the most challenging aspects of aerial navigation is the unpredictable nature of obstacles. While some can be static and known, others might appear suddenly or be less visible. In this experiment, the introduction of static imaginary obstacles was a deliberate move to simulate these challenges.

The G-BFS algorithm's response to these obstacles was commendable. Not only did it successfully detect them, but it also seamlessly recalculated the path to circumvent them. This obstacle avoidance capability is paramount for any real-world application of drone technology, whether it is for delivery services, surveillance, or recreational use.

The transition from a simulated environment to a real-world experiment brought forward the strengths of the G-BFS algorithm. Its consistent performance, coupled with precise navigation and reliable obstacle avoidance, is indicative of its potential in practical applications. Whether it is navigating busy urban spaces or exploring remote terrains, the G-BFS algorithm's real-world experiment underscores its readiness and efficiency in guiding autonomous aerial vehicles safely and effectively.

4.2. Traversed nodes

4.2.1. Simulation

In autonomous path planning, the number of nodes traversed by an algorithm is often indicative of its search strategy and efficiency. G-BFS, being a more exhaustive search method, naturally examines more nodes as it looks for the best possible path. RRT, on the other hand, focuses on quickly expanding into unexplored territories and thus often traverses fewer nodes (**Table 2**).

From the data, it is clear that the G-BFS algorithm tends to visit significantly more nodes compared to RRT, especially as grid sizes increase. This implies that while G-BFS might be more thorough in its search, RRT achieves results faster and with less computational effort, at least in terms of nodes visited.

It is worth noting that a higher node traversal does not inherently mean a better solution. It is about how the algorithm uses the information from those nodes. G-BFS might visit many nodes but might also find the most optimal path. RRT's efficiency in terms of nodes traversed could sometimes come at the cost of optimality.

For applications where quick responses are paramount, the efficiency of RRT in terms of nodes explored might be favoured. In contrast, for tasks where the best possible path is crucial and time is less of an issue, G-BFS's thorough exploration might be preferred. The trade-off between speed (RRT) and accuracy (G-BFS) needs to be carefully considered, depending on the application's needs.

Table 2. Consolidated algorithm traverse nodes.

Algorithm	**Grid size**				
	64 × 64	**128 × 128**	**256 × 256**	**512 × 512**	**1024 × 1024**
G-BFS	1120	2400	5900	10,100	20,300
RRT	42	69	153	1238	1863

4.2.2. Experiment

Implementing the RRT algorithm in a real-world setting like a quadcopter's navigation system is a definitive test of its capabilities. The results from the institute's grounds suggest that RRT not only meets the demands of such a complex system but excels in it.

The outdoor experiment underscored RRT's efficiency in real-world navigation. Despite the challenges posed by static imaginary obstacles, the algorithm showcased an impressive ability to swiftly compute paths, a testament to its quick exploration strategy observed in the simulation.

These experimental results reiterate the strengths of the RRT algorithm in scenarios where real-world responses are crucial. Its ability to provide efficient paths, especially when faced with unpredictable challenges, makes it a strong candidate for applications in aerial vehicle navigation. Developers and researchers might consider this algorithm when designing systems where speed and efficiency are at a premium, even if it means a slight compromise on the optimality of the path.

5. Conclusion

The intensive evaluation and experimentation of the G-BFS and RRT algorithms for path-finding in a 2D grid environment reveals the following:

- G-BFS is particularly effective in smaller grid scenarios due to its heuristic-driven approach, yielding faster execution times. However, the algorithm's performance deteriorates as grid size and complexity increase.
- RRT, with its stochastic nature, excels in large and intricate grid environments. While its execution times may be prolonged, especially in more complex search areas, it is notable for its efficient node traversal, even in such environments.
- The exhaustive exploration nature of G-BFS might be ideal for scenarios where high accuracy is demanded, especially in manageable grid sizes.
- Conversely, RRT's adaptability makes it a more suitable choice for environments that are expansive or convoluted, even if some accuracy is compromised.
- Considering the unique strengths of both algorithms, there is a promising avenue in developing hybrid algorithms that capitalize on G-BFS's precision and RRT's adaptability.
- The dynamic nature of path-finding algorithms suggests ample opportunities for future research, particularly in enhancing efficiency and broadening their application scope.

In essence, the nuanced behaviours and performances of G-BFS and RRT in 2D grid environments provide valuable insights for practitioners and decision-makers in UAV navigation. The significance of striking a balance between accuracy and efficiency remains central and the choice of algorithm should align with the specific requirements of the problem at hand. As UAV technology continues to evolve and integrate into diverse applications, so too will the necessity to refine and adapt path-finding algorithms for optimal performance. Future endeavours could include exploring strategies to tackle the challenges presented by larger, more intricate grid scenarios, as well as investigating the integration of other algorithms or methodologies to further enhance the efficiency and accuracy of UAV path-finding.

Author contributions

Conceptualization, CNP and MK; methodology, CNP, MK and KK; software, CNP, MK and KK; validation, CNP; formal analysis, CNP; investigation, CNP; resources, CNP; data curation, CNP; writing—original draft preparation, CNP; writing—review and editing, MK and KK; visualization, CNP; supervision, MK; project administration, MK; funding acquisition, CNP and KK. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1. Aggarwal S, Kumar N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. Computer Communications. 2020, 149: 270-299. doi: 10.1016/j.comcom.2019.10.014
2. Han J. An efficient approach to 3D path planning. Information Sciences. 2019, 478: 318-330. doi: 10.1016/j.ins.2018.11.045

3. Wakabayashi T, Yukimasa Suzuki, Suzuki S. Dynamic obstacle avoidance for Multi-rotor UAV using chance-constraints based on obstacle velocity. Robotics and Autonomous Systems. 2023, 160: 104320. doi: 10.1016/j.robot.2022.104320
4. Patle BK, Babu L G, Pandey A, et al. A review: On path planning strategies for navigation of mobile robot. Defence Technology. 2019, 15(4): 582-606. doi: 10.1016/j.dt.2019.04.011
5. Fu YT, Hsu CM, Lee MC, et al. A Path Planning Algorithm based on Leading Rapidly-exploring Random Trees. In: Proceedings of the 2019 International Automatic Control Conference (CACS); 13-16 November 2019; Keelung, Taiwan. pp. 1-6.
6. Karur K, Sharma N, Dharmatti C, et al. A Survey of Path Planning Algorithms for Mobile Robots. Vehicles. 2021, 3(3): 448-468. doi: 10.3390/vehicles3030027
7. Zhang L, Manocha D. An efficient retraction-based RRT planner. In: Proceedings of the 2008 IEEE International Conference on Robotics and Automation; 19-23 May 2008; Pasadena, CA, USA. pp. 3743-3750.
8. Singh M, Dhillon JS. A Simple Opposition-based Greedy Heuristic Search for Dynamic Economic Thermal Power Dispatch. Electric Power Components and Systems. 2016, 44(6): 589-605. doi: 10.1080/15325008.2015.1122113
9. Ahmed SS, Rao BPC, Jayakumar T. Greedy breadth-first-search based chain classification for nondestructive sizing of subsurface defects. Applied Soft Computing. 2016, 40: 260-273. doi: 10.1016/j.asoc.2015.11.032
10. Tian J, Chao T, Yang M, et al. A path planning algorithm based on improved RRT* for UAVs. In: Proceedings of the 2022 IEEE International Conference on Unmanned Systems (ICUS); 28-30 October 2022; Guangzhou, China. pp. 1-6.
11. Wang Y, Xi M, Weng Y. Intelligent path planning algorithm of Autonomous Underwater Vehicle based on vision under ocean current. Expert Systems. 2023. doi: 10.1111/exsy.13399
12. Liu B, Xiao X, Stone P. A Lifelong Learning Approach to Mobile Robot Navigation. IEEE Robotics and Automation Letters. 2021, 6(2): 1090-1096. doi: 10.1109/lra.2021.3056373
13. Noto M, Sato H. A method for the shortest path search by extended Dijkstra algorithm. In: Proceedings of the 2000 IEEE International Conference on Systems, Man and Cybernetics. “Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions” (Cat No00CH37166); 8-11 October 2000; Nashville, TN, USA. pp. 2316-2320.
14. Duchoň F, Babinec A, Kajan M, et al. Path Planning with Modified a Star Algorithm for a Mobile Robot. Procedia Engineering. 2014, 96: 59-69. doi: 10.1016/j.proeng.2014.12.098
15. Heusner M, Keller T, Helmert M. Best-Case and Worst-Case Behavior of Greedy Best-First Search. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence; 13-19 July 2018; Stockholm, Sweden. pp. 1463-1470.
16. Tripathy HK, Mishra S, Thakkar HK, et al. CARE: A Collision-Aware Mobile Robot Navigation in Grid Environment using Improved Breadth First Search. Computers & Electrical Engineering. 2021, 94: 107327. doi: 10.1016/j.compeleceng.2021.107327
17. Gao P, Liu Z, Wu Z, et al. A Global Path Planning Algorithm for Robots Using Reinforcement Learning. In: Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO); 6-8 December 2019; Dali, China. pp. 1693-1698.
18. Navya P, Ranjith R. Analysis of Path Planning Algorithms For Service Robots in Hospital Environment. In: Proceedings of the 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT); 6-8 July 2021; Kharagpur, India. pp. 1-6.
19. Li Q, Xie F, Zhao J, et al. FPS: Fast Path Planner Algorithm Based on Sparse Visibility Graph and Bidirectional Breadth-First Search. Remote Sensing. 2022, 14(15): 3720. doi: 10.3390/rs14153720
20. Yuanhao H, Shi H, Hao W, et al. Application of 3-D Path Planning and Obstacle Avoidance Algorithms on Obstacle-Overcoming Robots. In: Proceedings of the 2023 IEEE 5th Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS); 2-4 June 2023; Tainan, Taiwan. pp. 207-212.
21. Paces P, Udatny V. Comparison of Flight-Planning Algorithms in View of Certification Requirements. In: Proceedings of the 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC); 8-12 September 2019; San Diego, CA, USA. pp. 1-8.
22. Gnanaraj AAM, Abbas Ali F, Arumugam AC, et al. Hierarchical IoT network management and cloud computing to make healthcare green. AIP Conference Proceedings. 2023, 2581(1): 070001. doi: 10.1063/5.0126165
23. Kumar SV, Mary GAA, Mahdal M. Integrated Edge Deployable Fault Diagnostic Algorithm for the Internet of Things (IoT): A Methane Sensing Application. Sensors. 2023, 23(14): 6266. doi: 10.3390/s23146266
24. Vásconez JP, Basoalto F, Briceño IC, et al. Comparison of path planning methods for robot navigation in simulated agricultural environments. Procedia Computer Science. 2023, 220: 898-903. doi: 10.1016/j.procs.2023.03.122
25. Cao S, Fan P, Yan T, et al. Inland Waterway Ship Path Planning Based on Improved RRT Algorithm. Journal of Marine Science and Engineering. 2022, 10(10): 1460. doi: 10.3390/jmse10101460
26. Karaman S, Walter MR, Perez A, et al. Anytime Motion Planning using the RRT*. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation; 9-13 May 2011; Shanghai, China. pp. 1478-1483.
27. Melchior NA, Simmons R. Particle RRT for Path Planning with Uncertainty. In: Proceedings 2007 IEEE International Conference on Robotics and Automation; 10-14 April 2007; Rome, Italy. pp. 1617-1624.

28. Kang JG, Lim DW, Choi YS, et al. Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning. Sensors. 2021, 21(2): 333. doi: 10.3390/s21020333
29. Ma H, Meng F, Ye C, et al. Bi-Risk-RRT Based Efficient Motion Planning for Autonomous Ground Vehicles. IEEE Transactions on Intelligent Vehicles. 2022, 7(3): 722-733. doi: 10.1109/tiv.2022.3152740
30. Mao S, Yang P, Gao D, et al. A Motion Planning Method for Unmanned Surface Vehicle Based on Improved RRT Algorithm. Journal of Marine Science and Engineering. 2023, 11(4): 687. doi: 10.3390/jmse11040687