RAPID RANDOM TREE AND DUBINS PATH ALGORITHM FOR DRONE OBSTACLE AVOIDANCE

Nur Aliah Saparin, Mastura Ab Wahid*, Norazila Othman

Faculty of Mechanical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia

*Corresponding email: masturawahid@utm.my

Article history
Received
15th August 2024
Revised
16th December 2024
Accepted
10th February 2025
Published
20th June 2025

ABSTRACT

Unmanned aerial vehicles (UAVs) have been commonly used for domestic and military surveillance. Path planning algorithms are one of the many tools that can be used to control UAVs to make the flight autonomous and decrease the involvement of human pilots. The purpose of this study is to develop and implement path planning algorithms to enhance the navigation capabilities of drones in complex environments. Rapidly-exploring Random Tree (RRT) is one of the algorithms that uses sampling based methods to find possible flight paths in highly-dimension space. This research presents the simulation of the algorithm testing in various maps with randomly placed obstacles in MATLAB software to see whether the algorithm managed to manoeuvre around the obstacle successfully with minimum collisions. To further refine the generated path, Dubins path smoothing techniques are implemented in the simulation, to ensure a smoother and more feasible trajectories for drone navigation. The path generated is observed through the result shown in the simulation in both 2D view and 3D view for 3 different random cases. To summarize, the application of Dubins path smoothing significantly improves the efficiency of the path-planning algorithm and reduces the time to complete the obstacle navigation by 10-27% depending on the complexity of the map. The smoothed paths are more direct, involve fewer abrupt turns, and consistently reduce the time to reach the goal across different maps.

Keywords: *RRT algorithm, Dubin paths, Obstacle Avoidance, MATLAB.*

1.0 INTRODUCTION

Unmanned Aerial Vehicles (UAVs), commonly known as drones, operate without onboard human pilots, controlled remotely or autonomously by computers. UAVs are widely used for repetitive, complex, or dangerous tasks, finding applications in both military and civilian sectors such as payload delivery, traffic monitoring, surveillance, aerial surveys, search and rescue, agriculture, and corporate marketing. The effectiveness of UAVs in these roles depends on feasible and optimal trajectory planning, as they must dynamically interact with other objects during flight [1-9]. Current UAVs are designed to increase autonomy and flight stability to handle diverse tasks. UAVs are classified into single-rotor, multi-rotor, fixed-wing, and hybrid types, each with distinct advantages and limitations.

In scenarios where the spaces will be shared with other vehicles, machinery, humans, or objects in movement, the UAV to be equipped with new tools or systems to avoid those risks. The goal is to design and implement a reliable algorithm that can detect and maneuver around obstacles while minimizing the risk of collisions. These risks explain why researchers create best possible path-planning method and test it in simulations similar to real-life environments. Through extensive simulations, the researchers will analyze the performance of different algorithms to see whether they can successfully maneuver the drone through various obstacles. Ultimately, these findings aim to advance UAV technology while enhancing drones' safety and reliability in practical applications.

Path planning for UAVs involves considerations such as stealth to minimize detection, physical feasibility regarding path distance and leg length, mission performance requirements like maximum turning angles and flying height, and real-time implementation for efficiency in changing environments. There are different path-planning methods that can be explored to aid obstacle avoidance as shown in Table 1. These methods are used for UAVs that need to replan paths to adapt to unforeseen circumstances [7-9].

Table 1: Comparison of algorithm

Algorithm	Advantages	Disadvantages
Dijkstra	Find the shortest path	High computational cost
	Simple and easy to implement	Explore unnecessarily in large areas
	Works with weighted graph	
A*	Efficient (faster than Dijkstra)	Performance depends on the Heuristic quality
	Find the shortest path	Consume lots of memory
	Flexible	
D*	Designed for dynamic environments	More complex to implement
	Efficient replanning	Initial computation can be expensive
	Adapt to changes in real-time	
RRT	Suitable for high-dimensional spaces	Does not guarantee the shortest path
	Probabilistic completeness	Randomness
	Efficient (in exploring large spaces)	Sensitive to the tuning parameters

These characteristics make each algorithm suitable for different problems and environments depending on the applications' specific requirements and constraints. After careful consideration, the RRT algorithm will be used for the simulation to work in high dimensional spaces and will not require high computational cost.

Path smoothing techniques are important in various applications including robotics, autonomous vehicles and computer graphics, to create feasible and efficient vehicle path. The main goal of path smoothing techniques is to ensure that the paths are smooth and to reduce abrupt changes in the direction or speed which could lead to inefficient or unsafe maneuvers. Various techniques have been developed for path smoothing, and Dubins paths are particularly notable for their simplicity and effectiveness in environments with constraints on the curvature of the path. Even though there are downsides to the Dubins paths, such as a lack of flexibility for complex motion in comparison to other smoothing techniques like Bezier curves, B-splines and others, considering the current constraint of the problem is only on obstacle avoidance with constant speed, the Dubins path is choosen. This research uses different scenarios to test the performance of the RRT with Dubins path as compared to RRT algorithm [10-14]. Study in [11] perform Dubins path oriented RRT* algorithm which focuses on the flight and the minimum radius of rotation. It was found that the algorithm improves the path length by 14.87% and computing time by 82.36%. In this study, the Dubins path is applied to RRT and the map is generated randomly to see the effectiveness of the implementation of the Dubins path.

2.0 METHODOLOGY

The methodology used in of the project consisted of several elements. First, construct and build the environment and obstacles for the simulation. Next, the RRT algorithm will be implemented in the MATLAB software. Incorporate Dubins path smoothing to ensure the path is smoother and feasible for drone maneuvering. The simulation is fully tested in MATLAB software.

2.1 Modelling and Building the Simulation Environment

A 3D occupancy map is created using the MATLAB's 'occupancyMap3D' object. The map has the dimensions of 200*200 units and ten randomly placed obstacles is generated. Each obstacle is defined randomly, generating its width, length and height within the specified ranges. The position will also be randomly placed but still within the map boundaries. Each obstacle will be checked for intersections with existing obstacles by verifying if any points within the new obstacle are already occupied on the map. The code run until non-intersection obstacle was found and finally, the ground plane at *z* = 0 was filled to create a base layer, and the map was displayed using the 'show' function. This simulation will be tested various map displays, as shown in Figure 1.

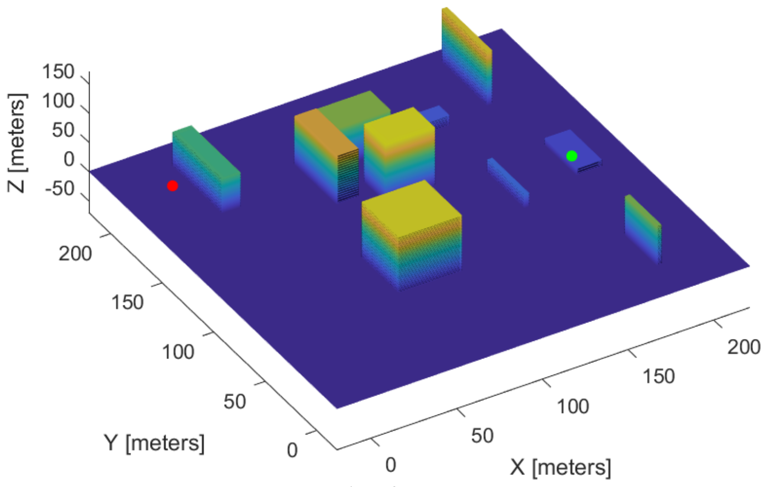


Figure 1: Example of 3D occupancy map

2.2 RRT Algorithm

Rapidly Exploring Random Trees (RRT) is an algorithm used in robotics and motion planning to explore and navigate high-dimensional configuration spaces efficiently. The algorithm iteratively builds a tree by randomly sampling points in the configuration space and connecting them to the nearest existing node, ensuring collision-free paths. Starting from an initial configuration, a random configuration is sampled, and a new node is created by extending from the nearest node towards the sampled point within a distance threshold. If the path is collision-free, the new node is added to the tree. This process continues until a specified number of iterations or a termination condition is met, resulting in a connected roadmap of the configuration space. The algorithm can be adapted to terminate when the closest distance to the goal point is reached, using a while loop instead of a for loop. Detailed information on the algorithm of the RRT and Dubins method can be found in [11-13]. Table 2 shows the algorithm used for the testing.

Table 2: *RRT algorithm* [15]

```
Generate_RRT($x_{init}, K, \Delta t$)
1  $\tau.init(x_{init})$;
2  for $k$=1 to $K$ do
3  $x_{rand} \leftarrow RANDOM\_STATE()$;
4  $x_{near} \leftarrow NEAREST\_NEIGHBOR(x_{rand}, \tau)$;
5  $u \leftarrow$ SELECT_INPUT$(x_{rand}, x_{near})$;
6  $x\_new \leftarrow NEW\_STATE(x_{near}, u, \Delta t)$;
7  $\tau.add\_vertex(x_{new})$;
8  $\tau.add\_vertex\ (x_{near}, x_{new}, u)$;
9  $Return\ \tau$
```

The x_{init} indicates an initial position of a robot in the Cartesian coordinate. *K* indicates the number of vertices of a tree, and the algorithm iterates *K* times before termination. This loop termination condition can be substituted by checking the closest distance from the tree to the goal point. To implement this, use ‘while loop’ instead of ‘for loop’. Additionally, it can make the loop iterate at least *K* times before termination. *Δt* indicates the time interval and *τ* represents a tree structure containing nodes sampled from configuration space and *u* indicates the control input. C, C_{tree} and C_{obs} indicates configuration space, free configuration space and obstructed configuration space, respectively.Where C is given in the 2D-map.

2.3 Simulation

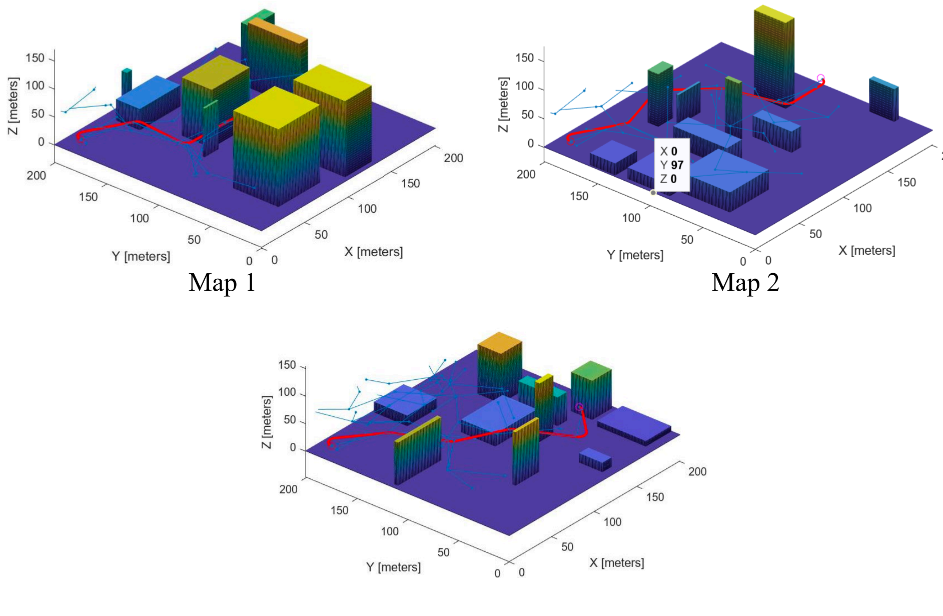
The simulation demonstrates UAV navigation through a 3D environment with obstacles. A pre-defined 3D occupancy map is used, with unknown spaces considered navigable. The UAV's starting and goal poses are defined, and an ‘ExamplehelperUAVStateSpace’ object constrains the UAV's roll angle, airspeed, flight path angle, and workspace bounds. A state validator ensures state validity within the map.

The RRT path planner, configured with a 50-unit maximum connection distance, 400 iterations, and a goal reach threshold of 5 units, attempts to find a feasible path from the start to the goal. Successful paths are stored in 'pthObj' and 'solnInfo'. The code visualizes the search tree, interpolates the planned path for smoothness, and simulates the UAV's trajectory (red) against the planned path (green).

To enhance realism, the initially planned path is smoothed using a Dubins path smoothing function, reducing sharp turns and providing a continuous trajectory. This is done once the RRT tree and vertex is found. The RRT will connect the vertex via a line, the Dubins path will be called to smooth the path. The final plot compares the smoothed reference path (green) with the simulated flight path (red), clearly representing the UAV's navigation.

3.0 RESULTS AND DISCUSSION

The simulation results are visualized using a 3D occupancy map where the UAV navigates through a complex environment populated with randomly placed obstacles. The UAV's path planning and trajectory are demonstrated using the Rapidly Exploring Random Trees (RRT) algorithm, followed by a Dubins path smoothing function for a more realistic flight path. The simulation is run multiple times with different maps and randomly placed obstacles. The simulated results are displayed in the 2D and 3D simulation map as shown in Figure 2. The figures illustrate both the planned (green) and the simulated (red) flight paths, allowing for an evaluation of the UAV's performance in navigating the environment.



Map 3
Figure 2: 3D view of RRT algorithm for 3 random cases

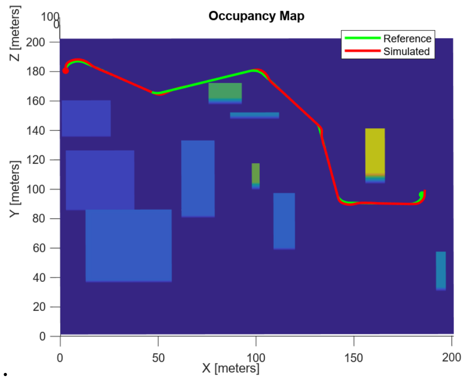


Figure 3: 2D view of the generated path

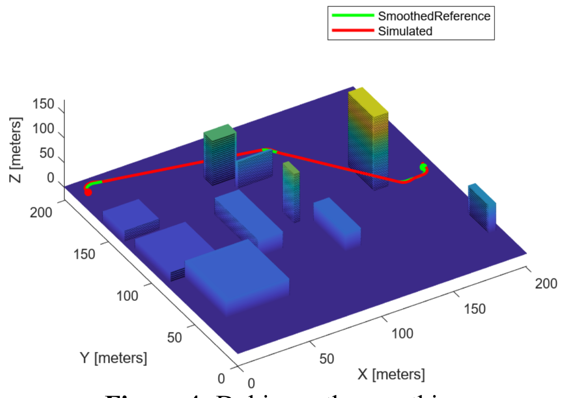


Figure 4: Dubins path smoothing

Figure 2 shows a 3D environment with various obstacles depicted as blocks of different heights. The red line represents the path planned by the UAV starting from the start pose and moving towards the goal pose while avoiding obstacles. The planned path appears to navigate clearly around the obstacles maintaining a safe distance from them. The blue line represents the RRT's exploration, forming a search tree that obtains the planned path. The RRT algorithm is designed to efficiently explore high-dimensional spaces by randomly sampling points in the map and connecting them to the nearest existing node in the tree, gradually expanding the tree until the goal is reached or the maximum number of iterations is achieved.

In this scenario, the search tree expands through the 3D occupancy map, navigating around the obstacles represented by the 3D blocks. The resulting path avoids collisions with these obstacles, showing the RRT's effectiveness in finding a path within a complex environment. However, the path can be rough due to the nature of RRT's random sampling and connection process.

Figure 3 provides a 2D top view of the planned path. This view helps better understanding the path's trajectory and its relation to the obstacles. In this view, the path weaves through the obstacles, following the shortest feasible route from start pose to the goal pose.

Meanwhile, Figure 4 shows the path after applying Dubins smoothing. Dubins paths are used for smoothing trajectories in environments where the drone has constraints on its turning radius, providing a more realistic and feasible path. The sharp turns and rough section from the original RRT path are replaced with smoother curves. This results in a more practical and efficient route for the drone to follow. It also reduced the likelihood of abrupt maneuvers that could be difficult for the drone to execute.

Table 3: Comparison of time taken to reach the goal before and after Dubins smoothing

Map	Time to reach the goal before Dubins smoothing (s)	Time to reach goal after Dubins smoothing (s)	Flight Time Improvement %
1	54.544	40.003	27
2	51.462	46.454	10
3	44.687	39.092	13

4.0 CONCLUSION

The application of the RRT algorithm for the simulation shows its effectiveness in navigating complex 3D environments by efficiently exploring space and generating feasible paths, as shown by the search tree and planned path. The 2D top view provides a clearer view of the path's trajectory relative to the obstacles and highlights areas for potential improvement. The application of Dubins smoothing transforms the initially rough path into a smoother and more practical route for the drone, ensuring a feasible path for real-world navigation. This process provides a foundation for understanding the RRT algorithm's function in initial path planning and the importance of post-processing for optimal and realistic path execution. Three random mappings were generated to test the RRT and Dubins Path algorithm. It was found that the addition of Dubins path on the path reduces the time to navigate from 10-27% depending on the complexity of the map.

CONFLICT OF INTEREST

The author declares that there is no conflict of interest regarding the publication of this paper.

ACKNOWLEDGEMENT

The author would like to thank and acknowledge Universiti Teknologi Malaysia (UTM) and UTM Aerolab for the facilities given for the research.

REFERENCES

[1] Mahmoud, M. S., Oyedeji, M. O., & Xia, Y. (2021). Path planning in autonomous aerial vehicles. In *Advanced distributed consensus for multiagent systems* (pp. 331–362). Elsevier. https://doi.org/10.1016/b978-0-12-821186-1.00018-0

[2] Radmanesh, M., Kumar, M., Guentert, P. H., & Sarim, M. (2018). Overview of path-planning and obstacle avoidance algorithms for UAVs: A comparative study. *Unmanned Systems, 6*(2), 95–118. https://doi.org/10.1142/S2301385018400022

[3] Dhulkefl, E. J., & Durdu, A. (2019). Path planning algorithms for unmanned aerial vehicles. *International Journal of Trend in Scientific Research and Development, 3*(4), 359–362. https://doi.org/10.31142/ijtsrd23696

[4] Gugan, G., & Haque, A. (2023). Path planning for autonomous drones: Challenges and future directions. *Drones, 7*(3). https://doi.org/10.3390/drones7030169

[5] Karur, K., Sharma, N., Dharmatti, C., & Siegel, J. E. (2021). A survey of path planning algorithms for mobile robots. *Vehicles, 3*(3), 448–468. https://doi.org/10.3390/vehicles3030027

[6] Tony, L. A., Ghose, D., & Chakravarthy, A. (2018). Precision UAV collision avoidance using computationally efficient avoidance maps. *2018 AIAA Guidance, Navigation, and Control Conference*. https://doi.org/10.2514/6.2018-0875

[7] Zhang, R., *et al.* (2024). Intelligent path planning by an improved RRT algorithm with dual grid map. *Alexandria Engineering Journal, 88*, 91–104. https://doi.org/10.1016/j.aej.2023.12.044

[8] Jeauneau, V., Jouanneau, L., & Kotenkoff, A. (2018). Path planner methods for UAVs in real environment. *IFAC-PapersOnLine, 51*(22), 292–297. https://doi.org/10.1016/j.ifacol.2018.11.557

[9] Vasudevan, V. (2021). *Obstacle detection and avoidance algorithm in drones: Airsim simulation with HIL* [Doctoral dissertation, California State University, Northridge].

[10] Xu, Z., *et al.* (2019). A study on path planning algorithms of UAV collision avoidance. *Journal of Northwestern Polytechnical University, 37*(1), 100–106. https://doi.org/10.1051/jnwpu/20193710100

[11] Yang, Y., Leeghim, H., & Kim, D. (2022). Dubins path-oriented rapidly exploring random tree* for three-dimensional path planning of unmanned aerial vehicles. *Electronics, 11*(15), 2338. https://doi.org/10.3390/electronics11152338

[12] Ravankar, A., *et al.* (2018). Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors, 18*(9). https://doi.org/10.3390/s18093170

[13] Shkel, A. M., & Lumelsky, V. (2001). Classification of the Dubins set. *Robotics and Autonomous Systems, 34*(4), 179–202. https://doi.org/10.1016/s0921-8890(00)00127-5

[14] Oh, Y., Cho, K., & Oh, S. (2018). Robust multi-objective path planning for flying robots under wind disturbance. *2018 15th International Conference on Ubiquitous Robots (UR)* (pp. 670–675). https://doi.org/10.1109/URAI.2018.8441822

[15] LaValle, S. (1998). *Rapidly-exploring random trees: A new tool for path planning* [Research Report 9811].