MANCHESTER 1824
The University of Manchester

Enhanced RRT* Path Planning for Drone Autonomous Navigation

A dissertation submitted to The University of Manchester for the degree of
Master of Science in Robotics
in the Faculty of Science and Engineering

Year of submission
2024

Student ID
10614473

School of Engineering

Contents

Word count: 5835

Abstract

The ability to generate dynamically feasible trajectories and avoid collision with both static and dynamic obstacles is crucial for the autonomous navigation of Unmanned Aerial Vehicles (UAV). The aim of this dissertation is to develop an enhanced version of the RRT* path planning algorithm that effectively navigates 3D dynamic environments while accommodating drone kinematic constraints. The proposed method introduces additional constraints on the random node sampling and steering processes of the 3D RRT* algorithm, including maximum turning angles and ground clearance. It also incorporates efficient dynamic re-planning to handle moving obstacles, and re-uses previously constructed trees to manage moving targets. Simulation results demonstrated that a feasible and collision-free path could be found in a static obstructed environment, although computation time increased with the number of constraints and obstacles. While the adaptation of the algorithm to dynamic environments was successful with high computational efficiency, the lack of real-world drone testing revealed limitations in robustness with unpredictable target movements and rapidly moving obstacles. Finally, due to the inherent randomness of the algorithm, the choice of random seed significantly influenced the quality of the generated paths.

Declaration of originality

I hereby confirm that this dissertation is my own original work unless referenced clearly to the contrary, and that no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual property statement

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/_files/Library-regulations.pdf`).

1 Introduction

1.1 Background and motivation

The use of unmanned aerial vehicles (UAVs), particularly quadcopters, has significantly expanded in recent years, primarily to automate high-precision processes and handle hazardous tasks. While drones have revolutionised numerous industries, the current drone technology remains limited and faces considerable challenges [1]. Most drones today are either teleoperated by a human operator or follow a pre-programmed path, making their operation complex and restricting their potential applications. Consequently, developing autonomous systems for drones would be highly advantageous in such use cases [2].

Path planning algorithms are crucial for the navigation of autonomous systems, as they are closely linked to high-level mission planning. These algorithms are typically designed to generate a path from the position of the system to a specified target position in the environment through a series of waypoints (Eq. 1).

$$path = \{(x_{\text{start}}, y_{\text{start}}, z_{\text{start}}), \ldots, (x_n, y_n, z_n), \ldots, (x_{\text{goal}}, y_{\text{goal}}, z_{\text{goal}})\} \tag{1}$$

For years, path planning was predominantly approached as a two-dimensional (2D) problem, primarily utilised by ground vehicles to navigate relatively simple environments. As more complex three-dimensional (3D) environments emerged, the development of 3D path planning methods became essential [3]. Although many algorithms can produce an optimal path from the start position to the goal, they often fail to account for the motion constraints of the system and the dynamic nature of real-world environments, where obstacles may move, new obstacles can emerge, or targets may change positions during operation [3].

1.2 Aims and objectives

This dissertation aims to develop and evaluate an enhanced version of the RRT* path planning algorithm capable of effectively navigating 3D dynamic environments while accounting for drone kinematic constraints, ensuring reliable autonomous drone operation. Three research objectives were formulated:

- Implement the 3D RRT* base algorithm with obstacle avoidance and heuristic search strategy within a static environment.
- Develop an enhanced RRT* algorithm which accounts for drone motion and size constraints, such as maximum turning angles and ground clearance, within a static environment.
- Adapt the enhanced RRT* algorithm for operation in dynamic environments, while improving processing efficiency and planning times.

2 Literature review

This section reviews the current state of the art in path planning algorithms for autonomous systems, with an emphasis on the improved variations of the Rapidly Random Tree (RRT) algorithm. Furthermore, this dissertation will be contextualised within the current existing literature, highlighting how it builds upon and extends these developments.

2.1 State of The Art in Path Planning for Autonomous Systems

Today, path planning algorithms are generally classified into two main categories: **Graph-based** and **Sample-based**. **Graph-based** algorithms represent the environment as a graph or grid of nodes. Each node in the grid is then evaluated by the algorithm to determine whether it coincides with an obstacle. The cost associated with each node is then calculated, and the optimal path is identified by navigating through the nodes with the lowest costs. This characteristic is why such algorithms are often referred to as heuristic-based [4]. In contrast, **Sampling-based** methods involve the random sampling of nodes within the environment and assessing if each node complies with a set of constraints. Valid nodes are then connected and build a path [5]. Finally, other popular methods were developed to overcome the limitations of individual algorithms such as Potential Field Method, Biologically Inspired algorithm, and Hybrid/Multi-Fusion algorithms [6].

The **A* algorithm** is an example of graph-based path planning methods and is widely considered as one of the most popular in this category. The working principle of A* builds on Dijkstra's algorithm which methodologically investigates all possible paths from a start position to a target position in a weighted graph. By comparing the cumulative cost of each possible path from the start to the goal, the path with the lowest cost is chosen, ensuring an optimal path. To improve the efficiency of the search and reduce computation time, the A* algorithm utilises the following heuristic function,

$$f(n) = g(n) + h(n) \tag{2}$$

where g(n) represents the cost from the start node to the current node n, and h(n) is the estimated cost from node n to the goal, typically calculated using the Euclidean or Manhattan distance. This additional heuristic ensures a goal-oriented search compared to Dijkstra's where the algorithm does not converge until all possible paths are evaluated [7].

An example of the A* algorithm being applied to UAV 3D path planning can be shown in [8]. The heuristic function f(n) has been redesigned such that it takes into consideration the motion constraints of the drone, as well as possible threats such as missiles and weather. The effectiveness of the proposed method was highlighted where the UAV was able to navigate in low altitudes, avoiding radar detection in air-battle scenarios. Yet, the computational effort increased in more complex environments, and the method was not suitable for dynamic environments.

Another study tested a dynamic A* variant, D* Lite, for 3D optimal path planning for quadcopters [9]. The D* Lite method employs a heuristic-search which enables fast re-planning by updating only new nodes, while still guaranteeing the shortest path. The algorithm demonstrated effective avoidance of static and unknown obstacles, however, it was not tested in a dynamic environment.

While graph-based methods like A* have proven effective in finding the shortest path in static 2D environments, they require high computational resources in complex 3D environments and are not suitable for dynamic real-time trajectory planning [6].

The **Rapidly Random Tree** (RRT) algorithm is widely regarded as one of the most advanced techniques in the sample-based methods [7]. It involves constructing a unidirectional tree from randomly sampled nodes in the environment, extending from the start position to the goal. Generally, RRT trees expand uniformly in collision-free space if there are no additional constraints on the random node generation process. The path is then generated by connecting the different nodes in the tree which do not collide with obstacles, continuing this process until one branch reaches the goal.

In fact, the RRT algorithm was tested with a ground mobile manipulator for delivering goods in an obstructed 2D environment [10]. The robot navigated the environment successfully while avoiding static obstacles and picking up objects. However, it was mentioned that computational complexity increased with real-time path planning, and that the choice of random seed affected the path length.

In another study, the collision avoidance capabilities of the RRT algorithm were evaluated in a 3D underwater environment with an autonomous underwater vehicle [11]. The short path length, fast processing time, and accurate obstacle avoidance were highlighted, although local-optima trapping, convergence, and dynamic environments were not tested.

Finally, a hybrid method has been proposed in [12], which consists fusing the D* Lite algorithm (graph-based) with the RRT algorithm (sample-based). The RRT algorithm starts by constructing a 3D random tree while considering the motion constraints of the UAV, and saves the collision checking results in a road map. Then, when a threat arises, the D* Lite uses the road map to re-plan a collision free path, which solves the 3D dynamic path planning problem and reduces processing time.

Although the RRT algorithm does not guarantee an optimal path due to the randomness in the algorithm, it is a computationally efficient solution which is applicable to 3D complex and dynamic environments [13]. It can also manage kinodynamic, non-holonomic, and holonomic constraints [14].

2.2 RRT Enhancements

To address the lack of optimality in the RRT algorithm, a more advanced version was created, RRT*. The latter improves upon RRT by introducing a “rewiring” step which considers the minimisation of the path length when selecting nodes from the tree. This refinement allows RRT* to achieve asymptotic optimality [15], nevertheless, the planning time increased significantly in higher-dimensional environments. In the aim of accelerating the convergence rate, the RRT*-Smart was proposed in [16]. Similarly to RRT*, the extended version samples the environment and generates an initial path. Yet, the random node generation is done through intelligent sampling and the initial path is then optimised. While the algorithm converges in fewer iterations, it has not been tested in 3D environments. Other variants of RRT* have been also developed to further improve feasibility, optimality, computational time, and smoothness of the generated path.

Kinodynamic RRT* (MK-RRT*) was developed for multi-robot 3D trajectory planning and was tested with the Ryze Tello EDU Drone [17]. The RRT* algorithm is coupled with a Model Predictive Controller which predicts the next position of the drone, considers motion constraints, and performs collision checks in each iteration. Although asymptotic optimality was achieved with real-time planning, computation time increased with the complexity of obstacles and number of drones.

Another method was proposed in [18] to achieve a feasible, optimal and collision-free path for UAVs by adding dynamic constraints on the random node generation process in RRT* such as maximum range, turning angle, heuristic search, and dynamic step. Simulation results showed a decrease in the number of nodes in the path and an improved running time, but it was not tested in a three-dimensional environment.

A hybrid method combining RRT and artificial potential field (APF) algorithm accelerated convergence speed, optimised, and smoothed the generated path in a simulated 3D Dynamic Environment. Real-time re-planning was faster as the path was only re-planned from the current position of the UAV to the goal when an obstacle appeared [19].

Methods such as RRT*-AR [20] and bi-RRT* [21] are used in real-time UAV applications to manage moving targets in dynamic environments. Both methods generate two or more trees simultaneously in different directions until one of them reaches the goal. Still, they are inefficient in environments with complex obstacle placement. To increase efficiency, [20] reuses partial trees from previous iterations to decrease the number of new generated nodes.

Finally, the RRT*-Connect algorithm implements a pruning-reconnecting mechanism and adaptive sampling such that dynamic obstacles are avoided efficiently [22][23]. It consists of only re-planning the part of the path which intersects with the obstacle, decreasing re-planning time considerably.

In summary, while graph-based techniques ensure an optimal path, they struggle with complex kinematic constraints and they exhibit poor real-time performance due to their inefficient search strategy. In contrast, although sampling-based methods generate paths based on randomly sampled environment information, they do not require pre-processing, and can be easily adapted to 3D environments due to their simplicity [6]. This dissertation builds upon the developments done on the RRT* algorithm to better handle dynamic environments, incorporating drone constraints.

Table 1. Path Planning Categories Comparison

Category	Algorithms	Optimality	Computation	Motion Constraints	Dynamic Planning
Graph-based	Djikstra's, A*	High	High	Limited	Limited
Sample-based	RRT, PRM	Moderate	Moderate	Good	Good
Hybrid Methods	Algorithm Fusion	Varies	Varies	Good	Good

3 Methodology

This section details the development of an improved RRT* path planning algorithm for autonomous drone navigation. The approach focuses on three primary aspects: the implementation of the standard 3D RRT* algorithm, the development of a 3D RRT* variant, and the adaptation of the algorithm to dynamic environments.

Custom implementations of the algorithms were created to ensure flexibility and control over the development process using a high–level, interpreted programming language, Python (Version 3.11.7). The only external libraries utilised were NumPy, for numerical operations, and Matplotlib, for visualisation of the results. The complete codebase and documentation for this work are available on this Github repository.

3.1 3D RRT* Algorithm

To establish the foundational concepts of the Rapidly-exploring Random Tree Star (RRT*), the basic form of the algorithm was first introduced in a two-dimensional and obstacle-free configuration space, then expanded to more complex environments.

3.1.1 Base RRT* Algorithm

In robotics, path planning is framed as a search problem within an environment, represented by a configuration space (C-space), which is divided into free-space C_{free} and obstacle space C_{obs}. The robot is allowed to take any configuration q in the C-space with $q = (x, y) \in \mathbb{R}^2$ and $q \in C$. At this stage, the C-space was assumed to be completely obstacle-free with $C = C_{\text{free}}$.

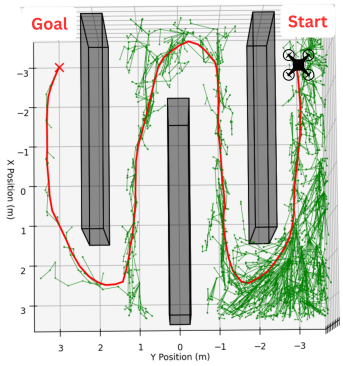

Fig. 1. RRT* Path Planning Tree

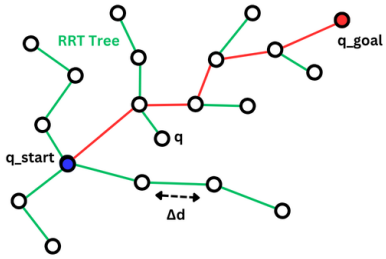

Fig. 2. RRT* Terminology

First, some notation.

- **Node (q):** A point in the C-space, represented by spatial coordinates $q = (x, y)$ (Fig. 2).
- **Tree (T):** A data structure composed of interconnected nodes expanding iteratively (Fig. 1).
- **Initial Configuration (q_{start}):** The starting position of the robot within the C-space.
- **Goal Configuration (q_{goal}):** The target position in the C-space where the robot aims to go (Fig. 1).

- **Maximum Extend Length Δd:** The maximum allowed distance that the tree can extend to in one step towards a random node, controlling the granularity of the exploration. A small Δd provides a smoother path and more precise control, but requires more computational effort. While a larger Δd requires less processing time, it could result in overshooting or missing narrow gaps.
- **Search Radius r:** Nodes within this radius are considered for potential shorter paths to the new node.
- **Maximum Iteration Number N:** The total number of iterations or samples the algorithm will generate, limiting the duration of the search process.

The algorithm begins by initialising the tree T with q_{start} as the root node. The tree is then expanded by one random node towards the goal in each iteration by performing the following steps:

1. **Sampling:** A random node q_{rand} with $q_{\text{rand}} \in C$ is generated using the python random number generator with a fixed $seed$. This seed ensures the reproducibility of random samples across different runs of the algorithm.
2. **Nearest Node:** The nearest node q_{nearest} in the tree T to the random sample q_{rand} is identified by calculating the Euclidean distance between them:

$$d_{\text{Eucl}} = \sqrt{(x_{\text{rand}} - x_{\text{nearest}})^2 + (y_{\text{rand}} - y_{\text{nearest}})^2} \tag{3}$$

3. **Steering:** The nearest node q_{nearest} is connected to the random node q_{rand} in the direction of q_{rand} with distance d_{Eucl} or the maximum extend distance Δd if $d_{\text{Eucl}} > \Delta d$. This new node q_{new} is then added to the tree T. This concept is illustrated in Fig. 3.
4. **Rewiring:** The Euclidean distance between q_{new} and all neighbouring nodes within the search radius r of q_{new} is calculated to assess if connecting q_{new} with any of those neighbouring nodes will result in a shorter path, with the aim of optimising the path length (Fig. 4). If a shorter path is found, the tree is updated by rewiring q_{nearest} to q_{new}.
5. **Iteration:** The same process is repeated until a path has been found from q_{start} to q_{goal}. If the maximum number of iterations N is reached and q_{goal} is not yet connected to q_{start}, the algorithm has failed to find a valid path.

Fig. 3. Steering Mechanism

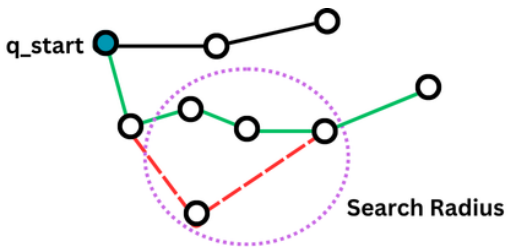

Fig. 4. Rewiring Mechanism

3.1.2 3D RRT* Algorithm with Obstacle Avoidance

Now, the algorithm was expanded further with obstacle avoidance capabilities in a three-dimensional configuration space. The configuration space $\mathcal{C}$ was represented as $\mathbb{R}^3$, with configurations defined by $q = (x, y, z)$, and $\mathcal{C} = \mathcal{C}_{\text{free}} + \mathcal{C}_{\text{obs}}$.

To account for the third dimension (z-axis), the start, goal, and sampled nodes were points in 3D space with coordinates (x, y, z). Regarding **steering**, the Euclidean distance calculation was also adapted to include the z-axis. In the **rewiring** process, while the search radius was represented as a circle, encompassing all nodes around q_{new} within that circular region, the search radius was expanded into a sphere in 3D.

To incorporate obstacle detection and avoidance, an additional step, **collision-detection** was added after the **steering** process and in the **rewiring** process. Obstacles in the 3D configuration space were represented as cuboids parameterised by $[x, y, z, d_x, d_y, d_z]$, where (x, y, z) are the coordinates of the geometric centre of the cuboid, and (d_x, d_y, d_z) are the lengths of the cuboid in each direction. All obstacles were then stored in an array and passed to the collision detection function alongside q_{new} and q_{nearest}.

The **collision-detection** function was responsible for determining in each iteration if the segment connecting q_{new} and q_{nearest} intersects with any obstacle by iterating through all the obstacles in the environment. The minimum and maximum corners of each obstacle were calculated by subtracting and adding half of the obstacle's dimensions from its centre coordinates respectively. Moreover, a safety margin of 0.15 m was added to each side of the obstacle to consider the width of the drone (0.1 m). Finally, if a collision has been detected, the new sampled node was omitted from the tree to save memory, and a new node was sampled.

The expansion to three-dimensional environments increased the complexity of the problem, making the process more computationally expensive, which required some parameter tuning. The maximum extend length Δd and search radius r were increased as the size of the configuration space has expanded. Finally, the maximum number of iterations N was also increased to allow more time for the algorithm to converge to a near-optimal path.

3.2 Improved 3D RRT* Algorithm

While the 3D RRT* algorithm with obstacle avoidance provides a foundational approach for drone path planning, it does not fully account for the kinematic constraints of the drone, which can result in unrealistic paths [17]. To generate more feasible trajectories, the algorithm was further enhanced by incorporating two additional constraints alongside obstacle avoidance, and smoothing. Additionally, to reduce computational efforts, the heuristic search strategy was implemented.

3.2.1 Turning Angle Constraints

As discussed in the literature review, the base 3D RRT* algorithm may produce paths with sharp turns, which are neither realistically navigable by the drone nor safe [18]. Implementing turning angle constraints limits the maximum allowable change in direction between consecutive path segments, resulting in smoother trajectories, and safer roll, pitch, and yaw angles for the drone. The

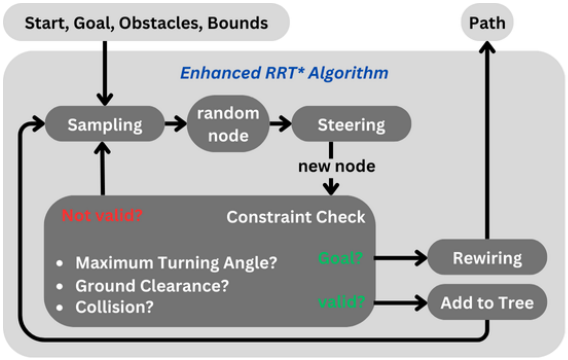

Fig. 5. Improved RRT* Flow Chart

max-turning-angle function was introduced after the **steering** process and in the **rewiring** process alongside obstacle detection to address the practicality of the generated path.

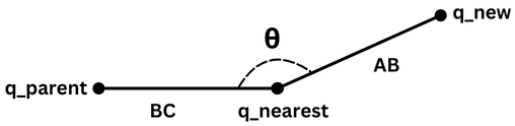

Fig. 6. Turning Angle Constraints

The purpose of the **max-turning-angle** function was to evaluate, during each iteration, whether adding the newly sampled node q_{new} to the path would lead to a sharp turn. To determine this, the angle between two segments in the path, formed by three points q_{new}, q_{nearest}, and q_{parent}, was calculated (Eq. 4), where the last node added to the path in the previous iteration is represented by q_{parent}. In Eq. 4, AB is the segment formed by q_{new} and q_{nearest}, and BC is the segment formed by q_{nearest} and q_{parent} (Fig. 6).

$$\theta = \cos^{-1}\left(\frac{AB \cdot BC}{\|AB\| \cdot \|BC\|}\right) \tag{4}$$

Three distinct angle ranges were selected to ensure the safe navigation of the drone: $[-25°, 25°]$, $[\pm 80°, \pm 100°]$, $[\pm 170°, \pm 180°]$

These angles were based on quadcopter dynamics, which dictate that the roll, pitch, and yaw angles of the drone control its vertical and horizontal movements [24]. Specifically, the drone rolls to turn right or left, pitches to move forward or backward, and adjusts its yaw angle to change heading. Notably, ascending or descending does not require pitching or rolling. The Tello drone was chosen as a case study example due to its affordability and ease of use, and testing attempts were made with this drone to determine angle constraints. Additionally, the Tello has been previously tested with the Kynodynamic RRT* method discussed in the literature review [17].

The first range, $[-25°, 25°]$, was selected based on the Ryze Tello Drone's user manual, which specifies that the maximum pitch and roll angles are $\pm 25°$ [25]. The second range, $[\pm 80°, \pm 100°]$, accounted for right-angle turns, which are manageable for drones since they ascend/descend without abrupt directional changes. The third range, $[\pm 170°, \pm 180°]$, corresponded to the drone travelling in a straight line, which does not necessitate additional manoeuvres. Angles outside these ranges could result in sharp turns or sudden altitude changes, potentially destabilising the drone.

3.2.2 Ground Clearance

Due to the inherent randomness of the base RRT* algorithm, there is a possibility that the algorithm might generate a path which intersects with the ground. To ensure that the generated path maintained a 3D trajectory that is feasible and safe for the drone, the **ground-clearance** function was introduced after the **steering** process and in the **rewiring** process alongside **obstacle-detection** and **max-turning-angle**.

During each iteration, the ground-clearance function evaluated whether the newly sampled node q_{new} intersects with the ground, i.e., if $q_{\text{new}}(z) = 0$. Nevertheless, this approach introduced some limitations in certain scenarios. If the starting position of the drone was on the ground ($q_{\text{start}}(z) = 0$), the algorithm would fail. Similarly, if the drone was airborne, yet the goal position was on the ground ($q_{\text{goal}}(z) = 0$), the drone would be unable to land. Finally, the minimum ground clearance must equal the height of the drone, which was just below 0.1 m.

To handle these scenarios, **ground-clearance** was only initialised after two iterations, giving additional time for the drone to take off if it was on the ground. In the landing scenario, ground-clearance was turned off when the drone was within two times the maximum extend length ($2\Delta d$). Finally, for safety reasons, the ground clearance was set to two times the height of the drone, 0.2 m.

An alternative approach could involve modelling the ground as an obstacle, but this would increase the complexity of the problem and lead to longer computation times.

3.2.3 Heuristic Search Strategy

While the additional constraints enabled the RRT* algorithm to generate a feasible path for drone navigation, they significantly increased computation effort. To mitigate this, a heuristic search strategy was implemented in the first step of the algorithm, **sampling**.

This method involved setting a threshold probability, **goal-sample-rate**, which dictates the likelihood of directly sampling the goal point during the random node generation process. During each iteration, a random number between 0 and 1 is generated and compared to the threshold probability. If the random number exceeds the threshold, the algorithm samples a random node within the C-space as usual. Otherwise, the algorithm selects the goal point as the next node. This **goal-bias** strategy speeds up the search process and reduces unnecessary exploration, particularly in the later stages of the search when the algorithm is approaching the goal.

3.3 Dynamic 3D RRT* Path Planning

An improved RRT* algorithm was developed for 3D static environments which accommodates kinematic constraints of drones. In this section, the enhanced algorithm was adapted to handle dynamic obstacles and targets.

3.3.1 Dynamic Obstacles

Dynamic path planning for mobile obstacles involves running the improved RRT* algorithm in real time. A straightforward approach to real-time path planning is to generate an entirely new tree and

corresponding path at each time step (Δt). However, this method is highly inefficient, demanding significant computational resources and potentially leading to slower system performance. The aim of this proposed method was to improve the efficiency of real-time path planning and accelerate the process. This method operated similarly to the approach used in static environments, with two main modifications.

At ($t = 0$), the algorithm was initialised with the starting position of the drone, the target location, and the current obstacles detected in the environment. An initial asymptotically optimal path was generated from the start to the goal, avoiding these obstacles. The drone then began to follow this path towards the static goal.

At each time step, only a **path-validation** function was executed to determine if any new obstacles have emerged that obstruct the current path. If an obstruction was detected, the **path-validation** function identified and returned the remaining valid portions of the path which were not obstructed, to a **path-reconstruction** function. The latter then localised the invalid segments and re-planned them to avoid the new obstacles using the same improved RRT* algorithm. The newly re-planned segments were subsequently merged with the valid portions of the original path.

Regarding the re-planning process, it was important to consider that the maximum extend length (Δd), search radius (r), and maximum iteration number (N) must be tuned to suit the smaller paths and achieve faster path planning. Finally, the selection of time step (Δt) was critical; if too large, the system becomes unresponsive, while if too small, it increases computational load.

3.3.2 Dynamic Target

In scenarios with a dynamic target, the **reuse-Tree** function was employed to maintain and adapt the existing tree structure. Instead of disregarding the entire tree when the target moves, this function reconfigured the previous tree based on the current position of the drone, and the new goal location.

If the drone has drifted from the current path, the drone was first “latched” to the existing tree, becoming the new root of the tree, and resetting the cost associated with all the existing branches. Then, without sampling new nodes, the algorithm evaluated the new cost of the nearest nodes relative to the new goal and attempted to construct a path with the existing branches. If the goal has moved significantly, the algorithm might have required to sample more nodes, which would be used to expand the tree towards the goal in the usual way.

Finally, to increase the robustness of the algorithm, a **predict-goal** function was added, which observed the goal position for at least two time steps, and predicted the next position of the goal. The predicted goal location was then returned to the **reuse-Tree** function.

4 Results and discussion

This section presents three simulation experiments focused on UAV applications, examining the feasibility and efficiency of the generated RRT* paths in both static and dynamic 3D environments with complex obstacle configurations. To ensure consistency, all tests were conducted on the same computer with an Intel® Core™ i7-9750H CPU, 2.60 GHz, NVIDIA GeForce GTX 1650 GPU, and

16 GB RAM memory, running Python (Version 3.11.7) on Windows 10.

4.1 Path Planning in Static Environments

The first experiment evaluated how each additional constraint (collision, turning, ground clearance) affected the feasibility of the path and efficiency of the improved algorithm.

To maintain coherence, the same environment map with dimensions 5m by 5m by 2m was used with identical obstacle configuration throughout this experiment. The drone consistently started at position (-4, 4, 0) with the target located at (4, -4, 0). Additionally, the same RRT* parameters and random seed were applied for all tests.

Table 2. Parameters for the Improved RRT* Algorithm

Maximum Extend Length	Search Radius	Goal Sample Rate	Random Seed
0.25 m	0.9 m	0.05	106

The following two figures (Fig. 7 and Fig. 8) show the performance of the 3D RRT* algorithm with obstacle avoidance, compared to the base RRT* algorithm. To begin with, the base RRT* path behaved as expected. It did not react to obstacles and reached the goal within 400 iterations, in a path measuring 14.76 meters with 61 nodes. By adding the collision avoidance constraint, the algorithm successfully navigated around the obstacles and reached the goal, which resulted in the generation of a longer path measuring 17 meters with 70 nodes and the doubling of processing time with 1,000 iterations. It was also observed that the collision-free path came too close to the obstacles, and that the first 8 nodes of the path, spanning to just above 0.7 meters, were positioned on the ground (z=0), as shown in Fig. 8. This phenomenon could have been caused by the inherent randomness of the algorithm and choice of the random seed. Therefore, the generated path was unsuitable for drone navigation, as drones cannot traverse on the ground nor approach obstacles closely.

Fig. 7. RRT* with Obstacle Avoidance (Top view)

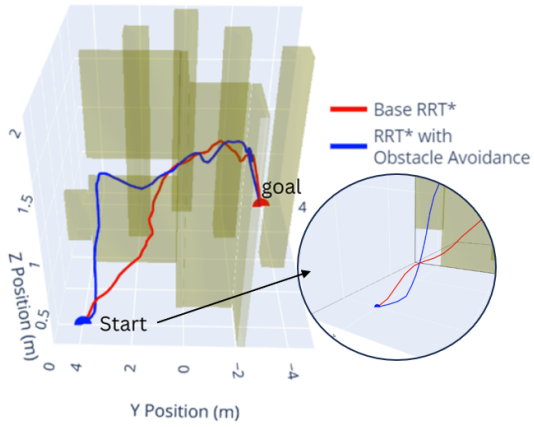

Fig. 8. RRT* with Obstacle Avoidance (Side view)

The effectiveness of the improved algorithm, considering the kinematic constraints of the drone, is illustrated in the following two figures (Fig. 9 and Fig. 10). It was observed that the generated path

with turning constraints exhibited greater smoothness and smaller turning angles, with a maximum turning angle of 25 degrees. Additionally, the path length significantly decreased, measuring 14.90 meters with 61 nodes, making it almost equal to the base path. Finally, the final improved 3D RRT* generated a path that remained off the ground, maintained a minimum collision margin of 0.15 meters (did not pass through the small gap shown in Fig. 10), and effectively limited turning angles. Nevertheless, the number of iterations increased remarkably, reaching 20,000 iterations, with a path length of 19.19 meters and 73 nodes in the path. Although the number of nodes per path only increased by 10 nodes and the path length was extended by just 4 metres to avoid obstacles, node utilisation dropped sharply from 28% to 8%, which could have been caused by inefficient sampling.

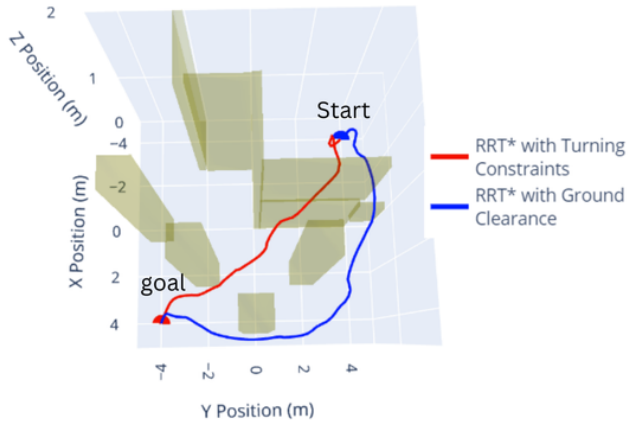

Fig. 9. Enhanced RRT* (Top View)

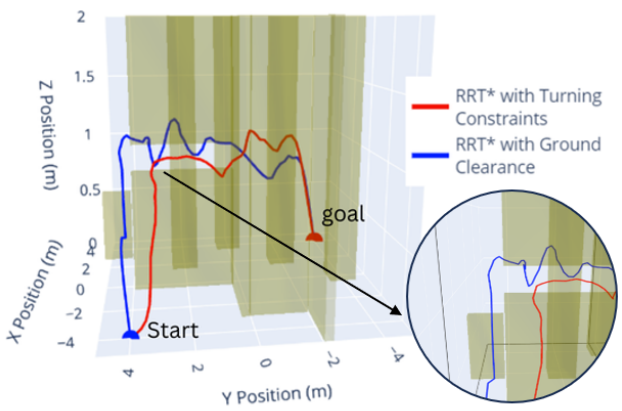

Fig. 10. Enhanced RRT* (Side View)

4.2 Path Planning in Dynamic Environments

The second experiment evaluated the performance of the improved algorithm in navigating around moving and newly emerging obstacles. It was tested in the same environment map as the first experiment, but featured a different obstacle configuration.

The experiment simulated a scenario where a drone is already airborne and wishes to land while there are dynamic obstacles. The drone started at location (-4, -4, 1.8) and landing site was at (4, 4, 0). For clarity, only 3 seconds of the simulation were presented to demonstrate the efficacity of the algorithm, although the simulation spanned 15 seconds. During the three-second intervals shown in figures 11, 12, and 13, the drone was assumed to be hovering at the start position, while detecting any moving or emerging obstacles that may obstruct its path to the landing site. The obstacles were positioned intentionally such that they collide with the initial path.

At ($t = 0$), a static obstacle (obstacle 1) was already present in the environment, detected by the algorithm and avoided (Fig. 11). At ($t = 1$), a new obstacle (obstacle 2) emerged, obstructing the initial path generated at ($t = 0$) (Fig. 12). The algorithm detected the collision immediately, re-planned only the affected segment, and merged it with the original path. At ($t = 2$), obstacle 2 moved again, obstructing the newly merged path (Fig. 13). The algorithm once more detected the collision and re-planned the obstructed segment. This approach accelerated the path planning considerably, as the re-planned section consisted of only 20 nodes, requiring just 3000 iterations, compared to 8,000 iterations needed to re-plan the entire path from start to goal which involved 60 nodes.

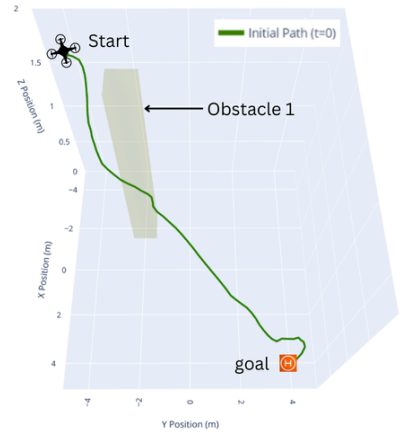

Fig. 11. Dynamic RRT* with Dynamic Obstacles (t=0)

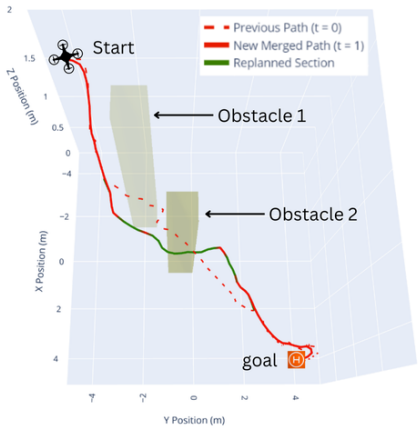

Fig. 12. Dynamic RRT* with Dynamic Obstacles (t=1)

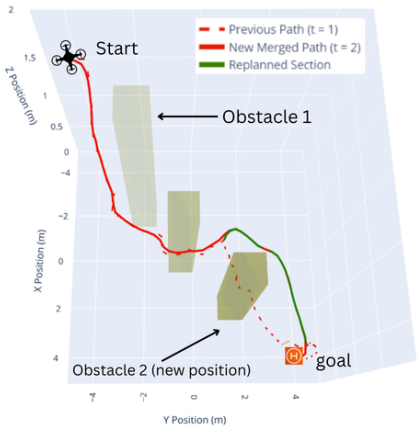

Fig. 13. Dynamic RRT* with Dynamic Obstacles (t=2)

Finally, the third experiment assessed the performance of the improved 3D RRT* algorithm in managing moving targets. It was tested in a different obstacle-free environment map with dimensions 55 m by 5 m by 2 m. The experiment simulated a scenario where a drone, already airborne, aims to land on a moving platform within a long, narrow corridor over a duration of 7 seconds. The drone began at coordinates (-18, 0, 1), while the moving landing platform started at (-15, 0, 0) with a constant velocity of 6 m/s (Fig. 14).

At ($t = 0$), the algorithm took the initial positions of the drone and platform, and generated an initial path. Then, the drone is assumed to have moved 1/3 of the initial path before detecting that the platform has moved, as simulating actual drone movements required the implementation of a drone system with robust controllers and positioning system. At ($t = 1$), the algorithm detected that the landing platform has moved, and re-planned a new path to the new target location using the previously generated tree. Then, the algorithm detected that the platform was moving in straight line at a constant velocity. Therefore, it predicted the next position of the platform at ($t = 2$), and planned a new path to the predicted goal location, making the drone always one step ahead of the platform. By utilising previous trees each iteration, the number of iterations per time-step was reduced to 250. However, the quality of the path declined and showed an important overshoot when the prediction step was activated due to the fast dynamics of the system.

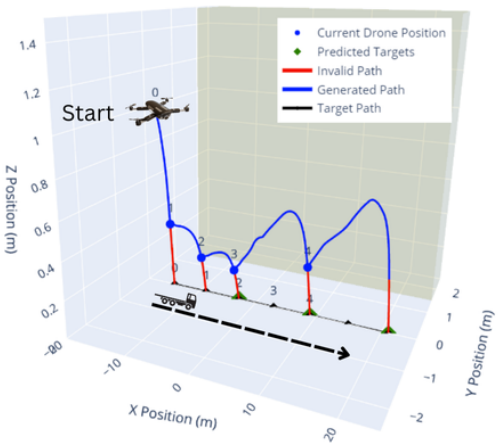

Fig. 14. Dynamic RRT* with Moving Target

4.3 Discussion

The improved 3D RRT* algorithm, including obstacle avoidance, turning constraints, and ground and obstacle clearance, generated a feasible path for drone navigation in both static and dynamic 3D environments.

In static obstructed environments, while the inclusion of obstacle avoidance and turning constraints only raised the iteration count from 400 to 4,000, the substantial increase to 20,000 iterations and decrease in efficiency was primarily due to the ground clearance requirement and the need to maintain a safety margin around obstacles, as shown in figure 15.

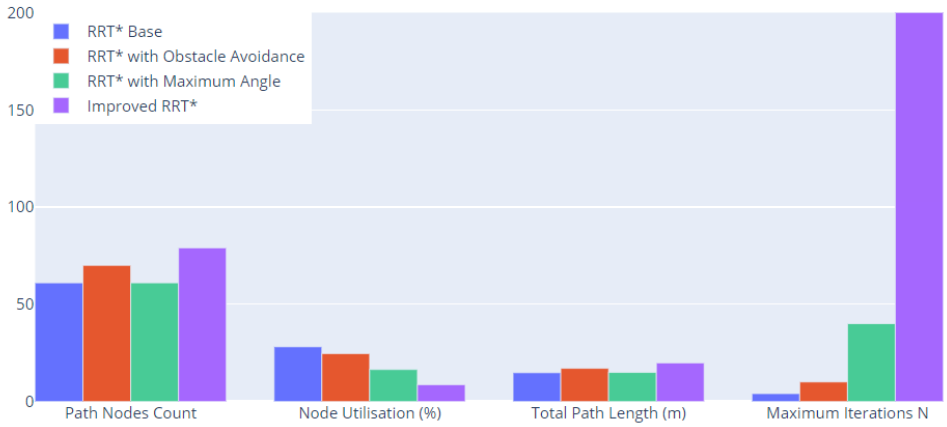

Fig. 15. 3D RRT* Comparison

The ground clearance constraint was particularly demanding because the drone operated at a relatively low altitude of 1.5 metres, increasing the likelihood of sampling nodes on the ground. This constraint also necessitated generating a completely new node each time a randomly sampled node failed to meet the criteria, which led to a higher number of generated nodes, thereby reducing node utilisation and efficiency. To improve the efficiency of constraint checking when a sampled node did not meet the constraints, a new node could have been steered in the same direction as the sampled node, but with a smaller angle or at a higher altitude, similar to the maximum extend length concept [26]. Accordingly, the addition of constraints on the base algorithm, as well as inefficient steering, led to the increase in computational effort, making the process less efficient.

The choice of random seed also affected the system's performance notably. Different random seeds influenced the distribution of sampled nodes, which in turn impacted the algorithm's ability to find a feasible path. In some cases, a certain seed could lead to a higher concentration of samples in less optimal regions, leading to the failure of the algorithm, while others provide near-optimal paths [27]. Therefore, the presented results were specific to the chosen seed, where other seeds have failed the experiments.

While the results for dynamic obstacles demonstrated a significant increase in processing efficiency, several limitations were identified. As the number of emerging obstacles obstructing the path increased, the number of segments requiring simultaneous re-planning also rose, leading to longer path planning times. Moreover, in highly dynamic environments, rapidly moving obstacles could overwhelm the algorithm, leading to inefficiencies caused by frequent interruptions. This

approach also depends heavily on the quality of the initial path; if the initial path is sub-optimal or fails, the cumulative effects of re-planning could introduce errors and reduce overall path efficiency. Lastly, the proposed experiment assumed the drone was hovering, which misrepresents reality where the algorithm must account for the drone's movement and actual position relative to obstacles. Testing was attempted with the Tello drone, however, the lack of an accurate positioning system made it impractical to effectively evaluate the algorithm. Brief results from these tests are provided in appendix A.

Regarding moving targets, the algorithm was tested in a simple environment with no obstacles, and the landing platform was assumed to have a constant velocity in a straight line. If the moving platform's movements were unpredictable, the prediction step would fail, leading to landing failure as the drone would be reacting to the current position of the platform, making it one step behind in each iteration. Furthermore, the algorithm could employ a dynamic extension length that adjusts based on the distance to the target, addressing the overshooting issue caused by the target's sudden distancing [28]. For instance, if the target is located at a greater distance, the extension length would increase, whereas if the target is closer, the extension length would decrease. Hence, the improved RRT* algorithm lacked robustness in more complex dynamic environments, and required real-world drone testing to validate the effectiveness and adaptability in dynamic settings.

5 Conclusions and future work

In conclusion, this dissertation focused on adapting the RRT* algorithm for drone autonomous navigation by incorporating kinematic constraints, obstacle avoidance, and dynamic re-planning. While these enhancements were effective in static environments enabling the generation of feasible paths for drones, they also led to an increase in computational effort. Although the algorithm demonstrated exceptional efficiency in dynamic environments, it struggled with rapidly moving obstacles and more complex obstacle configurations. Moreover, the algorithm's performance with moving targets was limited by the assumption of a simple drone motion model and the lack of real-world drone testing.

Future work should address these limitations by focusing on several key areas. Firstly, more advanced and efficient methods for kinematic constraints and dynamically adjusting the extension length of the tree could enhance computational efficiency. Additionally, strategies to manage varying random seeds should be developed to ensure greater robustness. Finally, conducting real-world testing with actual drones in complex dynamic environments is crucial to validate the algorithm's effectiveness and adaptability.

References

[1] J. Peksa and D. Mamchur, “A review on the state of the art in copter drones and flight control systems,” *Sensors*, vol. 24, no. 11, p. 3349, 2024. DOI: 10.3390/s24113349 (cited on p. 6).

[2] A. Moura *et al.*, “Autonomous uav landing approach for marine operations,” 2023. DOI: 10.1109/oceanslimerick52467.2023.10244606 (cited on p. 6).

[3] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, “Real-time motion planning with applications to autonomous urban driving,” *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009 (cited on p. 6).

[4] Y. Ma, H. Zhang, L. Qi, and J. He, “Three-dimensional uav path planning based on improved a* algorithm,” *Electro-Optics and Control*, vol. 26, no. 10, pp. 22–25, 2019 (cited on p. 7).

[5] A. Short, Z. Pan, Z. Larkin, and S. van Duin, “Recent progress on sampling based dynamic motion planning algorithms,” in *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, Alberta, Canada, 2016, pp. 1305–1311 (cited on p. 7).

[6] Y. Zheng, F. Sun, X. Lu, and Y. Song, “Overview of research on 3d path planning methods for rotor uav,” 2021. DOI: 10.1109/ecie52353.2021.00081 (cited on pp. 7–9).

[7] C. Zammit and E.-J. van Kampen, “Comparison between a* and rrt algorithms for 3d uav path planning,” *Unmanned Systems*, pp. 1–18, 2021. DOI: 10.1142/s2301385022500078 (cited on pp. 7, 8).

[8] R. Tianzhu, Z. Rui, X. Jie, and D. Zhuoning, “Three-dimensional path planning of uav based on an improved a* algorithm,” in *IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, Nanjing, 2016, pp. 140–145 (cited on p. 7).

[9] H. Kim, J. Jeong, N. Kim, and B. Kang, “A study on 3d optimal path planning for quadcopter uav based on d* lite,” 2019. DOI: 10.1109/icuas.2019.8797815 (cited on p. 7).

[10] Y. Dong, Y. Zhang, and J. Ai, “Experimental test of unmanned ground vehicle delivering goods using rrt path planning algorithm,” *Unmanned Systems*, vol. 05, no. 01, pp. 45–57, 2017. DOI: 10.1142/s2301385017500042 (cited on p. 8).

[11] A. Arifi, J. Lepagnot, S. Bouallègue, and L. Jourdan, “3d path planning of autonomous underwater vehicles using a rapidly-exploring random trees algorithm,” 2023. DOI: 10.1109/icaige58321.2023.10346490 (cited on p. 8).

[12] Y. Liu, W. Zhang, J. Shi, and G. Li, “A path planning method based on improved rrt,” 2014. DOI: 10.1109/cgncc.2014.7007284 (cited on p. 8).

[13] D. Devaurs, T. Simeon, and J. Cortés, “Optimal path planning in complex cost spaces with sampling-based algorithms,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 415–424, 2015 (cited on p. 8).

[14] C. Mthabela, D. Withey, and C. Kuchwa-Dube, “Rrt based path planning for mobile robots on a 3d surface mesh,” 2021. DOI: 10.1109/saupec/robmech/prasa52254.2021.9377014 (cited on p. 8).

[15] S. Karaman and E. Frazzoli, “Optimal kinodynamic motion planning using incremental sampling-based methods,” 2010. DOI: 10.1109/cdc.2010.5717430 (cited on p. 8).

[16] J. Nasir *et al.*, “Rrt*-smart: A rapid convergence implementation of rrt*,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 7, p. 299, 2013. DOI: 10.5772/56718 (cited on p. 8).

[17] B. Cain, M. Kalaitzakis, and N. Vitzilaios, “Mk-rrt*: Multi-robot kinodynamic rrt trajectory planning,” 2021. DOI: 10.1109/icuas51884.2021.9476688 (cited on pp. 9, 12, 13).

[18] X. Ren, L. Tan, S. Jiaqi, and X. Lian, “Multi-target uav path planning based on improved rrt algorithm,” *Journal of Physics: Conference Series*, vol. 1786, no. 1, p. 012 038, 2021. DOI: 10.1088/1742-6596/1786/1/012038 (cited on pp. 9, 12).

[19] W. Hai, Z. J. Sun, D. Li, and Q. Jin, “An improved rrt based 3-d path planning algorithm for uav,” 2019. DOI: 10.1109/ccdc.2019.8832661 (cited on p. 9).

[20] S. Choudhury, S. Scherer, and S. Singh, “Rrt*-ar: Sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter,” in *Zenodo (CERN European Organization for Nuclear Research)*, 2013. DOI: 10.1109/icra.2013.6631133 (cited on p. 9).

[21] S. Liu, H. Liu, K. Dong, X. Zhu, and B. Liang, “A path optimization algorithm for motion planning with the moving target,” 2018. DOI: 10.1109/icma.2018.8484607 (cited on p. 9).

[22] Y. Chen and L. Wang, “Adaptively dynamic rrt*-connect: Path planning for uavs against dynamic obstacles,” 2022. DOI: 10.1109/cacre54574.2022.9834188 (cited on p. 9).

[23] D. Connell and H. M. La, “Dynamic path planning and replanning for mobile robots using rrt,” 2017. DOI: 10.1109/smc.2017.8122814 (cited on p. 9).

[24] C. Rus, E. Lupulescu, M. Leba, and M. Risteiu, “Advanced mathematical modeling and control strategies for autonomous drone systems,” 2024. DOI: 10.1109/icccc62069.2024.10569613 (cited on p. 13).

[25] R. Robotics. “V1.4 tello user manual.” Accessed: August 28, 2024. (2018), [Online]. Available: https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20User%20Manual%20v1.4.pdf (cited on p. 13).

[26] L. Yu, Z. Wei, Z. Wang, Y. Hu, and H. Wang, “Path optimization of auv based on smooth-rrt algorithm,” in *Proceedings of the 2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, Aug. 2017. DOI: 10.1109/ICMA.2017.8016038 (cited on p. 19).

[27] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. DOI: 10.1177/0278364911406761 (cited on p. 19).

[28] J. Xiong and X. Duan, “Path planning for uav based on improved dynamic step rrt algorithm,” in *Journal of Physics: Conference Series*, vol. 1983, Jul. 2021, p. 012 034. DOI: 10.1088/1742-6596/1983/1/012034 (cited on p. 20).

Appendices

A Further Results

Algorithm 1 Improved 3D RRT* Algorithm Flow

Input: Start point q_{start}, Goal point q_{goal}, Map M, Max iterations N, Step size Δ, Neighbourhood radius r
Output: Optimal path from q_{start} to q_{goal} (if found)
Initialize tree T with root q_{start}
for $i = 1$ to N **do**
    Sample a random point q_{rand} in 3D space
    Find the nearest node q_{nearest} in tree T to q_{rand}
    Compute the new point q_{new} by moving from q_{nearest} towards q_{rand} by Δ
    if q_{new} is in free space **then**
        Find the set of nearby nodes S within radius r from q_{new}
        Select the best node q_{best} from S that minimises the cost
        Rewire the tree:
        for each node q in S **do**
            if path from q_{best} to q is cheaper than the existing path **then**
                Reconnect q to q_{best}
            end if
        end for
        Add q_{new} to tree T
        Connect q_{new} to q_{best}
        if distance between q_{new} and q_{goal} is less than Δ **then**
            Connect q_{new} to q_{goal}
            Return the optimal path from q_{start} to q_{goal} via tree T
        end if
    end if
end for
Return failure to find a path

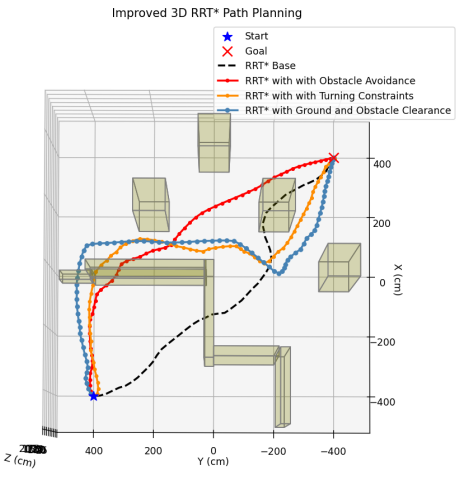

Fig. 16. Improved 3D RRT* Comparison

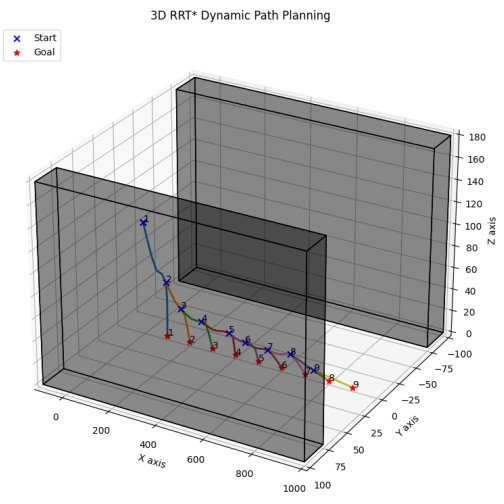

Fig. 17. Improved 3D RRT* with Moving Target

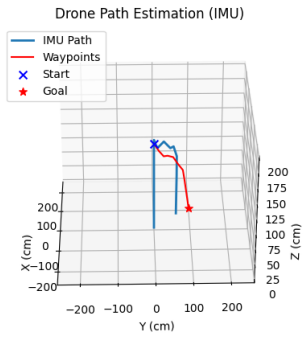

Fig. 18. Improved 3D RRT* testing with Tello Drone - Feedback positioning system based on IMU

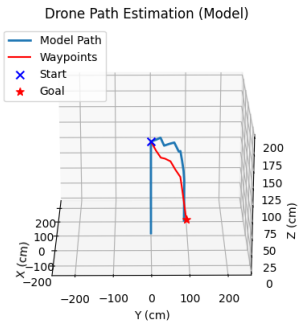

Fig. 19. Improved 3D RRT* testing with Tello Drone - Feedback positioning system based on Drone Model

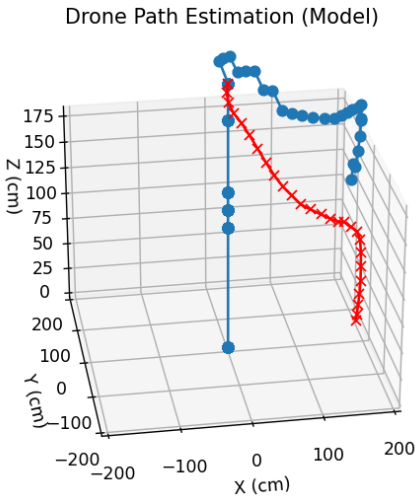

Fig. 20. Improved 3D RRT* testing with Tello Drone - Feedback positioning system based on Drone Model

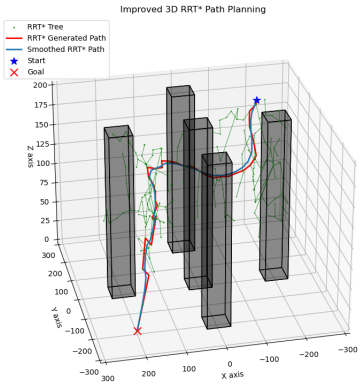

Fig. 21. Improved 3D RRT* testing in a maze

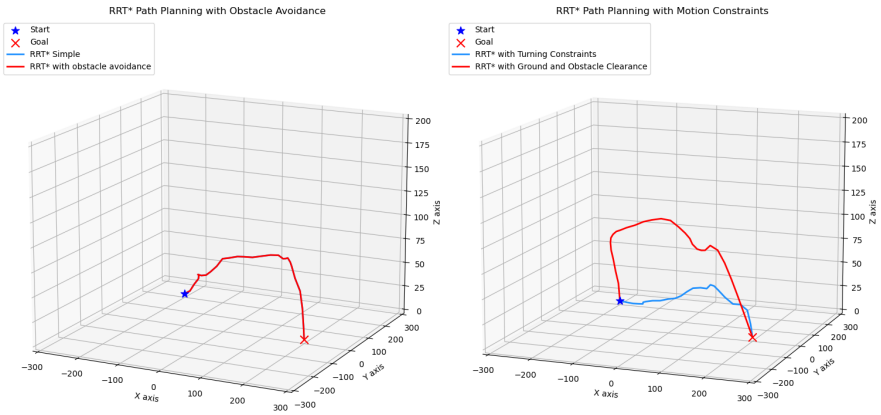

Fig. 22. Improved 3D RRT* with no obstacles - showing that improved RRT* does not traverse ground even when there are no obstacles

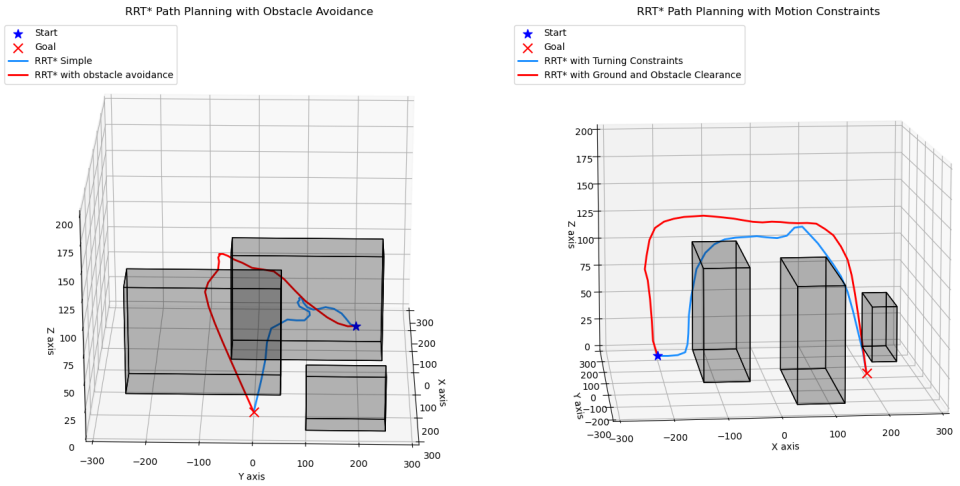

Fig. 23. Improved 3D RRT* with obstacles - map simulating indoor environment

B Project outline

The use of unmanned aerial vehicles (UAVs), particularly quadcopters, has grown significantly over the past years, aimed at handling monotonous tasks, protecting humans from hazardous activities, and optimising high precision processes. Nowadays, numerous applications rely on drone technology and can be found in nearly all the industries. Among these use cases are cinematography, industrial inspection, search and rescue missions, agriculture, and product delivery [1].

While drones have revolutionised the mentioned industries, the current drone technology remains limited and faces challenges. For instance, the current battery technology constrains the flight time and range of drones. Consequently, in applications such as offshore missions or operations in remote areas, drones must be transported by a ground, water, or aerial vehicle, take-off from it, perform its mission, and then land back on the moving vehicle to recharge. Most drones today are also still teleoperated by a human operator, or follow a pre-programmed path, which increases the difficulty of landing on a moving vehicle. Therefore, developing autonomous landing systems for drones would be highly advantageous in such scenarios [2].

For robust autonomous landing on a mobile station, the drone must have an accurate positioning system. Modern drones rely on various sensors, with GPS being the most common, to capture essential data including heading, orientation, speed, and altitude. However, GPS signals are unreliable in scenarios such as indoor missions, urban areas, regions with electromagnetic interference, dense forests, and wartime. As a result, GPS-free positioning systems are required [3][4].

Aims and Objectives

This dissertation aims to develop and analyse the performance of an autonomous landing system for drones on mobile base stations using a UWB positioning system, integrated with infrared sensors and IMU data. Four research objectives were formulated:

- Compare and identify the most suitable navigation algorithms for autonomous dynamic land-

ing in simulation

- Test the identified navigation algorithm on the DJI Tello drone for autonomous landing on a fixed platform, and analyse accuracy, precision, and mission time
- Test the identified navigation algorithm on the DJI Tello drone for autonomous landing on a moving platform, and analyse accuracy, precision, and mission time
- Evaluate the accuracy of the position measurement provided by the UWB and IR positioning system, and incorporate IMU measurements to increase accuracy

Project Plan:

- Week 1 (01/07 – 07/07): Order Components, and create drone simulation
- Week 2 (08/07 – 14/07): Test and identify best landing algorithm in simulation
- Week 3 (15/07 – 21/07): Setup DJI Tello drone, test simple manoeuvres, and integrate UWB and IR positioning system
- Week 4-5 (22/07 – 04/08): Test autonomous landing on static platform
- Week 6-7 (05/08 – 18/08): Test autonomous landing on mobile platform and Integrate IMU measurements with UWB and IR positioning system
- Week 8-9 (19/08 – 01/09): Dissertation writing

References

[1] J. Peksa and Dmytro Mamchur, “A Review on the State of the Art in Copter Drones and Flight Control Systems,” Sensors, vol. 24, no. 11, pp. 3349–3349, May 2024, doi: https://doi.org/10.3390/s24113349.

[2] A. Moura et al., “Autonomous UAV Landing Approach for Marine Operations,” Jun. 2023, doi: https://doi.org/10.1109/oceanslimerick52467.2023.10244606.

[3] C. Kim, EungChang Mason Lee, J. Choi, J. Jeon, S. Kim, and H. Myung, “ROLAND: Robust Landing of UAV on Moving Platform using Object Detection and UWB based Extended Kalman Filter,” 2021 21st International Conference on Control, Automation and Systems (ICCAS), Oct. 2021, doi: https://doi.org/10.23919/iccas52745.2021.9649920.

[4] T.-Y. Lo et al., “GPS-Free Wireless Precise Positioning System for Automatic Flying and Landing Application of Shipborne Unmanned Aerial Vehicle,” Sensors, vol. 24, no. 2, pp. 550–550, Jan. 2024, doi: https://doi.org/10.3390/s24020550.

C Risk assessment

MANCHESTER 1824

The University of Manchester

General Risk Assessment Form

Date: (1)	Assessed by: (2)	Checked / Validated* by: (3)	Location: (4)	Assessment ref no (5)	Review date: (6)
01/07/2024	Mahmoud Shanan		Engineering Building		

Task / premises: (7) Dynamic Landing of an Autonomous Quadcopter

The work includes three main activities:

- Programming the firmware of the drone
- Software simulation of the mission
- Drone flight testing

Activity (8)	Hazard (9)	Who might be harmed and how (10)	Existing measures to control risk (11)	Risk rating (12)	Result (13)
Use of display screen equipment (DSE)	Poor posture, repetitive movements, long periods looking at DSE	Student working on PC Repetitive strain injuries, neck and back pain, eye strain and/or fatigue	Provision of an adjustable chair, adjustable screen height, suitable and sufficient lighting is maintained in each area. Taking breaks to avoid eye strain and fatigue.	Low	A
Computer Use	Electric shock, burns, fires, electrocution	Student working on PC Electric shock, electrocution, and/or burns	Avoid interfering with plugs, cables, or any device, especially when any equipment is connected to the power supply. Report defective items to supervisor in the first instance. Avoid eating or drinking while working to minimise the risk of spillage onto electrical equipment.	Low	A

Result : T = trivial, A = adequately controlled, N = not adequately controlled, action required, U = unknown risk

Activity (8)	Hazard (9)	Who might be harmed and how (10)	Existing measures to control risk (11)	Risk rating (12)	Result (13)
Working from home	Lone working, electrical hardware failure	Home working student Anxiety, stress, electric shock, burns and fire	Remaining in contact with flatmates, friends, or family. Avoid working late at night and define working hours. Visual checks to make sure electrical equipment and cables are free from defects. Avoid daisy chaining.	Low	A
Indoor Drone flight testing	Failure of propellers, uncontrolled behaviour of drone	Drone operator and/or anyone present within the drone testing area Drone collision with person's body, sharp plastic parts scattering causing eye damage	The drone is equipped with a plastic mechanical frame, including propeller guards. Pre-flight inspection of the drone. Testing drone flight in the net area of the Engineering A_GA.033 lab.	Low	A
Indoor Drone flight testing	Electrical failure of batteries, electronics, sensors, motors	Drone operator and/or anyone present within the drone testing area Crash, electric shock, burns and fire	Visual and electrical checks before testing to make sure equipment and wires are free from defects. Testing drone flight in the net area of the Engineering A_GA.033 lab.	Low	A
Indoor Drone flight testing	Overheating	Drone operator and/or anyone present within the drone testing area Skin burns or fire	Limiting the testing time per day. Taking breaks between tests to let motors cool down.	Low	A

Result : T = trivial, A = adequately controlled, N = not adequately controlled, action required, U = unknown risk

Action plan (14)				
Ref No	**Further action required**	**Action by whom**	**Action by when**	**Done**

Result : T = trivial, A = adequately controlled, N = not adequately controlled, action required, U = unknown risk

Notes to accompany General Risk Assessment Form

This form is the one recommended by Health & Safety Services, and used on the University's risk assessment training courses. It is strongly suggested that you use it for all new assessments, and when existing assessments are being substantially revised. However, its use is not compulsory. Providing the assessor addresses the same issues; alternative layouts may be used.

(1) **Date** : Insert date that assessment form is completed. The assessment must be valid on that day, and subsequent days, unless circumstances change and amendments are necessary.

(2) **Assessed by** : Insert the name and signature of the assessor. For assessments other than very simple ones, the assessor should have attended the University course on risk assessments (link to STDU)

(3) **Checked / Validated* by** : delete one.

Checked by : Insert the name and signature of someone in a position to check that the assessment has been carried out by a competent person who can identify hazards and assess risk, and that the control measures are reasonable and in place. The checker will normally be a line manager, supervisor, principal investigator, etc. Checking will be appropriate for most risk assessments.

Validated by : Use this for higher risk scenarios, eg where complex calculations have to be validated by another "independent" person who is competent to do so, or where the control measure is a strict permit-to-work procedure requiring thorough preparation of a workplace. The validator should also have attended the University's risk assessment course or equivalent, and will probably be a chartered engineer or professional with expertise in the task being considered. Examples of where validation is required include designs for pressure vessels, load-bearing equipment, lifting equipment carrying personnel or items over populated areas, and similar situations.

(4) **Location** : insert details of the exact location, ie building, floor, room or laboratory etc

(5) **Assessment ref no** : use this to insert any local tracking references used by the school or administrative directorate

(6) **Review date** : insert details of when the assessment will be reviewed as a matter of routine. This might be in 1 year's time, at the end of a short programme of work, or longer period if risks are known to be stable. Note that any assessment must be reviewed if there are any significant changes – to the work activity, the vicinity, the people exposed to the risk, etc

(7) **Task / premises** : insert a brief summary of the task, eg typical office activities such as filing, DSE work, lifting and moving small objects, use of misc electrical equipment. Or, research project [title] involving the use of typical laboratory hardware, including fume cupboards, hot plates, ovens, analysis equipment, flammable solvents, etc.

(8) **Activity** : use the column to describe each separate activity covered by the assessment. The number of rows is unlimited, although how many are used for one assessment will depend on how the task / premises is sub-divided. For laboratory work, activities in one particular lab or for one particular project might include; use of gas cylinders, use of fume cupboard, use of computer or other electrical equipment, use of lab ovens, hot plates or heaters, use of substances hazardous to health, etc

(9) **Hazard** : for each activity, list the hazards. Remember to look at hazards that are not immediately obvious. For example, use of a lathe will require identification of the machine hazards, but also identification of hazards associated with the use of cutting oils (dermatitis), poor lighting, slipping on oil leaks, etc. The same activity might well have several hazards associated with it. Assessment of simple chemical risks (eg use of cleaning chemicals in accordance with the instructions on the bottle) may be recorded here. More complex COSHH

assessments eg for laboratory processes, should be recorded on the specific COSHH forms (link).

(10) **Who might be harmed and how** : insert everyone who might be affected by the activity and specify groups particularly at risk. Remember those who are not immediately involved in the work, including cleaners, young persons on work experience, maintenance contractors, Estates personnel carrying out routine maintenance and other work. Remember also that the risks for different groups will vary. Eg someone who needs to repair a laser may need to expose the beam path more than users of the laser would do. Vulnerable groups could include children on organised visits, someone who is pregnant, or employees and students with known disabilities or health conditions (this is not a definitive list).

For each group, describe how harm might come about, eg an obstruction or wet patch on an exit route is a hazard that might cause a trip and fall; use of electrical equipment might give rise to a risk of electric shock; use of a ultraviolet light source could burn eyes or skin.

(11) **Existing measures to control the risk** : list all measures that already mitigate the risk. Many of these will have been implemented for other reasons, but should nevertheless be recognised as means of controlling risk. For example, restricting access to laboratories or machine rooms for security reasons also controls the risk of unauthorised and unskilled access to dangerous equipment. A standard operating procedure or local rules (eg for work with ionising radiation, lasers or biological hazards) will often address risks. Some specific hazards may require detailed assessments in accordance with specific legislation (eg COSHH, DSEAR, manual handling, DSE work). Where this is the case, and a detailed assessment has already been done in another format, the master risk assessment can simply cross-reference to other documentation. For example, the activity might be use of a carcinogen, the hazard might be exposure to hazardous substances, the existing control measures might all be listed in a COSHH assessment. Controls might also include use of qualified and/or experienced staff who are competent to carry out certain tasks; an action plan might include training requirements for other people who will be carrying out those tasks.

(12) **Risk Rating** : the simplest form of risk assessment is to rate the remaining risk as high, medium or low, depending on how likely the activity is to cause harm and how serious that harm might be.

The risk is **LOW** - if it is most unlikely that harm would arise under the controlled conditions listed, and even if exposure occurred, the injury would be relatively slight.
The risk is **MEDIUM** - if it is more likely that harm might actually occur and the outcome could be more serious (eg some time off work, or a minor physical injury.
The risk is **HIGH** - if injury is likely to arise (eg there have been previous incidents, the situation looks like an accident waiting to happen) and that injury might be serious (broken bones, trip to the hospital, loss of consciousness), or even a fatality.

Schools or administrative directorates may choose to use other rating systems. Typical amongst these are matrices (of 3x3, 4x4, 5x5 or even more complex) which require the assessor to select a numerical rating for both "likelihood that harm will arise" and "severity of that harm". These may give a spurious sense of accuracy and reliability – none are based on quantitative methods. There are methods of estimating risk quantitatively, and these may be appropriate for complex design of load bearing structures and the like. Advice on methods of risk assessment is available from HSS. Whatever system of assessment is adopted, it is **essential** that the assessor has received suitable training and is familiar with the meaning of the terms (or numbers) used.

(13) **Result** : this stage of assessment is often overlooked, but is probably the most important. Assigning a number or rating to a risk does not mean that the risk is necessarily adequately controlled. The options for this column are:

T = trivial risk. Use for very low risk activities to show that you have correctly identified a hazard, but that in the particular circumstances, the risk is insignificant.

A = adequately controlled, no further action necessary. If your control measures lead you to conclude that the risk is low, and that all legislative requirements have been met (and University policies complied with), then insert A in this column.

N = not adequately controlled, actions required. Sometimes, particularly when setting up new procedures or adapting existing processes, the risk assessment might identify that the risk is high or medium when it is capable of being reduced by methods that are reasonably practicable. In these cases, an action plan is required. The plan should list the actions necessary, who they are to be carried out by, a date for completing the actions, and a signature box for the assessor to sign off that the action(s) has been satisfactorily completed. Some action plans will be complex documents; others may be one or two actions that can be completed with a short timescale.

U = unable to decide. Further information required. Use this designation if the assessor is unable to complete any of the boxes, for any reason. Sometimes, additional information can be obtained readily (eg from equipment or chemicals suppliers, specialist University advisors) but sometimes detailed and prolonged enquiries might be required. Eg is someone is moving a research programme from a research establishment overseas where health and safety legislation is very different from that in the UK.

For T and A results, the assessment is complete.
For N or U results, more work is required before the assessment can be signed off.

(14) **Action Plan**. Include details of any actions necessary in order to meet the requirements of the information in Section 11 'Existing measures to control the risk'. Identify someone who will be responsible for ensuring the action is taken and the date by which this should be completed. Put the date when the action has been completed in the final column.