IMPERIAL

Multi-UAV Path Planning for Urban Air Mobility

Author
H.S. Saddour
CID: 02488721

Supervised by
Dr G.S. Scarciotti

A Thesis submitted in fulfillment of requirements for the degree of
Master of Science in Control and Optimisation

Department of Electrical and Electronic Engineering
Imperial College London
2024

Abstract

3D path planning for Unmanned Aerial Vehicles (UAVs) involves finding an optimal, collision-free trajectory within a 3D cluttered environment. While significant progress has been made in addressing the challenges of UAV path planning, much of the existing research focuses on simplified scenarios rather than real-world urban environments. Additionally, most studies do not consider scenarios involving multiple UAVs, and those that do often rely on real-time, reactive avoidance rather than pre-planned, offline algorithms. To this end, this thesis presents a framework for 3D path planning in urban environments for multiple UAVs. This work is distinguished by introducing a new RRT*-based collision-free path planning algorithm for multiple UAVs implemented on real-world urban maps.

The implementation environments used in this work are set up by creating 3D occupancy maps and 3D UAV scenarios derived from urban maps. An enhanced Rapidly-exploring RandomTree Star (RRT*) algorithm was developed to overcome the limitations of the basic RRT* algorithm, specifically its poor directionality in path planning and the unsuitability of its paths for UAV flight. First, the sampling area was constrained, improving the path search efficiency. Then, a novel method was introduced to prune the generated paths by removing redundant waypoints. The shortest possible path was then identified by running the planning algorithm multiple times with different random seeds. Finally, a path smoothing algorithm using cubic splines was applied to optimise UAV flight paths. Simulation results demonstrate that the improved RRT* algorithm reduces path planning time and produces more optimised paths for UAVs. The improved single UAV path planning algorithm was extended to generate paths for multiple UAVs. An original editing algorithm was proposed for multi-UAV path planning, designed to maintain a minimum safety distance between UAVs throughout their trajectories. This is achieved by placing pseudo-obstacles and locally replanning path segments where potential collisions are detected. The algorithm's performance was validated in two distinct real-world urban environments.

Keywords: Unmanned Aerial Vehicle (UAV); multi-UAV; Path Planning; Collision avoidance; RRT*.

Declaration of Originality

I hereby declare that the work presented in this thesis is my own unless otherwise stated. To the best of my knowledge the work is original and ideas developed in collaboration with others have been appropriately referenced.

Copyright Declaration

Acknowledgments

I would like to express my gratitude to my supervisor, Dr Giordano Scarciotti, as well as Dr Martina Sciola and Dr Roberto Valenti from MathWorks for their invaluable guidance throughout this project.

I am deeply thankful to my mother, siblings and friends for their never-ending encouragement.

Lastly, I honour the memory of my father, whose influence has inspired me throughout this journey. I hope this work reflects his lasting impact.

Contents

Abstract i

Declaration of Originality iii

Copyright Declaration v

Acknowledgments vii

List of Acronyms xi

List of Figures xiii

1 Introduction 1

1.1 Multi-UAV Path Planning Problem 1

1.2 Problem formulation and objectives 2

1.3 Outline of the Thesis 3

2 Theoretical Background 5

2.1 UAV 3D Path Planning Algorithms 5

2.2 Rapidly-exploring Random Tree Algorithm 7

2.3 Rapidly-exploring RandomTree Star Algorithm (RRT*) 8

3 Urban Mapping and Scenario Preparation 11

3.1 Urban Test Areas: New York and London 11

3.2 UAV Scenarios 12

3.3 3D Occupancy Map 15

4 The Improved RRT* Algorithm 17

4.1 Restricted Search Area 17

4.2 Path Pruning 19

4.3 The Shortest Path 22

4.4 Path Smoothing 25

4.5 Summary 25

5 Multi-UAV Path Planning - Algorithm **29**

5.1 Generating Initial paths 29

5.2 Detecting Danger Points 31

5.3 Placing Pseudo-Obstacles 31

5.4 Local regeneration of safe path segments 32

5.5 User Guide 34

6 Multi-UAV Path Planning- Simulation **37**

6.1 Scenario 1: New York 38

6.2 Scenario 2: Imperial College London Campus 42

Conclusions **47**

6.3 Future Work 47

A Supplementary Materials **49**

A.1 Project Code 49

Bibliography **51**

List of Acronyms

RRT Rapidly-exploring RandomTree

RRT* Rapidly-exploring RandomTree Star

UAV Unmanned Aerial Vehicle

List of Figures

1.1 Taxonomy of general multi-UAV path planning problem [4]. 2

2.1 Taxonomy of 3D path planning algorithm taxonomy [6]. 6
2.2 Two successive planning iterations of the RRT algorithm 7
2.3 Optimised selection of parent nodes in the RRT* algorithm [15]. 9
2.4 Rewiring nodes in the random tree in the RRT* algorithm [15]. 9

3.1 Scenario 1: Satellite imagery [17] and UAV scenario 13
3.2 Scenario 2: Satellite imagery [18] and UAV scenario 14
3.3 Coverage Path Generated for LiDAR Simulation – Scenario 1 15
3.4 Generated 3D Occupancy Map – Scenario 1 16
3.5 Generated 3D Occupancy Map – Scenario 2 16

4.1 RRT* Output Comparison: Full vs. Reduced Sampling Range – Example 1 18
4.2 RRT* Output Comparison: Full vs. Reduced Sampling Range – Example 2 19
4.3 Path pruning - example 1 21
4.4 Path pruning - example 2 21
4.5 Comparison of results between proposed pruning algorithm and MATLAB smoothing function 22
4.6 RRT* output from 5 iterations showing all possible paths and the shortest path selected for two start-goal pairs 23
4.7 Minimum path distance versus number of iterations for different start-goal pairs . . 24
4.8 Output paths after applying the *hobbysplines* path smoothing function 25
4.9 The different improvements in the RRT* algorithm for single UAV path planning . 27

5.1 Initial paths for 3 UAVs in scenario 2 using the improved single UAV path planning algorithm 30
5.2 Danger points and pseudo-obstacles positions from the path editing algorithm for UAV1 31
5.3 Placement of pseudo-obstacles for path editing of UAV1 32
5.4 Sequential steps of the path editing algorithm 33
5.5 Initial (a) vs. enhanced (b) versions of the edited path 33

5.6 UAV1 collision-free path after applying the path editing algorithm 34

5.7 Selection of the number of UAVs and their start and goal positions using the designed tool 35

6.1 Selected start-goal pairs for the simulation in Scenario 1 39

6.2 Initial path planning output for the ten UAVs in Scenario 1 40

6.3 Final path planning output for the ten UAVs in Scenario 1 41

6.4 Selected start-goal pairs for the simulation in Scenario 2 43

6.5 Initial path planning output for the eight UAVs in Scenario 2 44

6.6 Placement of pseudo-obstacles for path editing of the eight UAVs in Scenario 2 . . 45

6.7 Final path planning output for the eight UAVs in Scenario 2 46

1 Introduction

Contents

1.1 Multi-UAV Path Planning Problem 1
1.2 Problem formulation and objectives 2
1.3 Outline of the Thesis 3

1.1 Multi-UAV Path Planning Problem

Unmanned aerial vehicles are widely used as platforms to work in various environments. They have a wide range of applications, such as search and rescue, parcel delivery, hidden area exploration and many others [1]. The main advantages of applying these systems are their flexibility, simultaneous actions, time efficiency and low cost [2]
Motion planning represents a fundamental step in the operation of autonomous Unmanned Aerial Vehicles (UAVs). By extending the definition of multi-robot path planning problem [3] to UAVs, as the UAV is considered a flying robot, we can introduce the general multi-UAV path planning problem as follows: given a set of N UAVs, each with an initial starting configuration and the desired goal configuration, determine the path each UAV should take to reach its goal while avoiding obstacles in the environment [4].

As path planning is a nondeterministic polynomial-time (NP-hard) problem[5], no standard solutions exist[6]. Navigating urban environments is particularly challenging because these areas are dense and intricately structured [5]. In 3D urban landscapes, UAV path planning systems

1

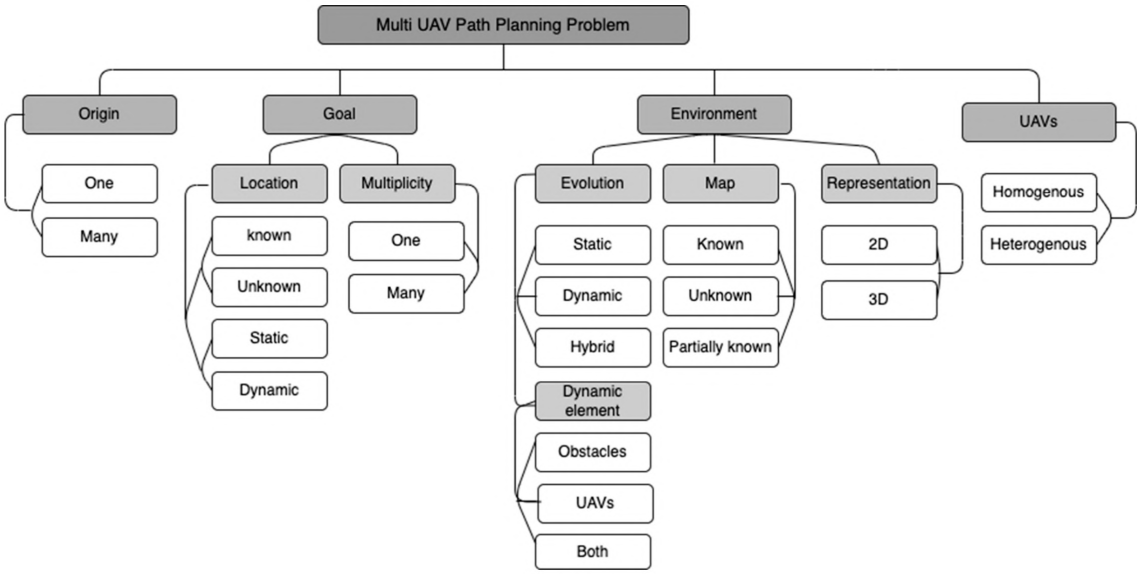


Figure 1.1: Taxonomy of general multi-UAV path planning problem [4].

must account for various constraints and requirements, including stationary and moving obstacles, dynamic targets, weather conditions, no-go zones, communication limitations, power restrictions, and other pertinent factors[7].

The multi-UAV path planning problem is usually described by some main parameters, including starting and end points, the environment, the type of the UAVs and the planning mode [4]. Multi-UAV path planning problem classification is illustrated in Figure 1.1 from [4].

1.2 Problem formulation and objectives

Based on the classification of the Multi-UAV Path Planning Problem presented in the previous section, the path planning problem tackled in the context of the work in this project is described as follows: we consider a mission of a global off-line path planning for a set of multiple UAVs travelling from different static starting points to different static goal locations, through a 3D fully known static environment consisting of static obstacles, represented by the buildings, and dynamic obstacles represented by the UAVs in the set. The goal is to generate paths for multiple UAVs in urban environments, prioritising path optimisation and collision avoidance.

1.3 Outline of the Thesis

This thesis is organised as follows: Chapter 2 introduces the preliminary material on the main UAV 3D path planning algorithms used in this project. Chapter 3 presents the selection and set-up process of the different scenarios and testing maps used to implement algorithms developed in this project's work. Chapter 4 introduces the proposed improved single UAV path planning algorithm and details the different enhancements from the basic RRT* algorithm and their purposes. In Chapter 5, the above algorithm is extended for multiple UAVs path planning, and a path editing algorithm is proposed to generate collision-free paths for multiple UAVs. Chapter 6 shows the results of applying the designed multi-UAV path planning algorithm in two different scenarios. Chapter 7, includes conclusions on the current work and suggestions for future work.

1

2 Theoretical Background

Contents

2.1 UAV 3D Path Planning Algorithms 5
2.2 Rapidly-exploring Random Tree Algorithm 7
2.3 Rapidly-exploring RandomTree Star Algorithm (RRT*) 8

2.1 UAV 3D Path Planning Algorithms

Various 3D path planning algorithms have been developed for UAVs to find flyable paths in different environments and scenarios. According to [6], the main UAV 3D path planning algorithms can be classified into five categories, as shown in Figure 2.1. Each one of these categories contains various methods that conform to specific characteristics.

Node-based and sampling-based algorithms are the most commonly used for aerial path planning due to their simplicity and adaptability to different environments and types of UAVs [7].

Sampling-based algorithms, such as the Rapidly-Exploring Random Tree (RRT) algorithm, begin by sampling the environment as a finite, non-structured set of points. They then establish connections between these points using techniques like the nearest node procedure or depth-first search. Finally, these methods initiate a search to find an optimal path [6]. The advantage of applying this kind of method is that it is easy to implement and efficient and ultimately guarantees a solution [7]

Node-based (or Graph-based) searching methods, like the A* algorithm, define the state space

2

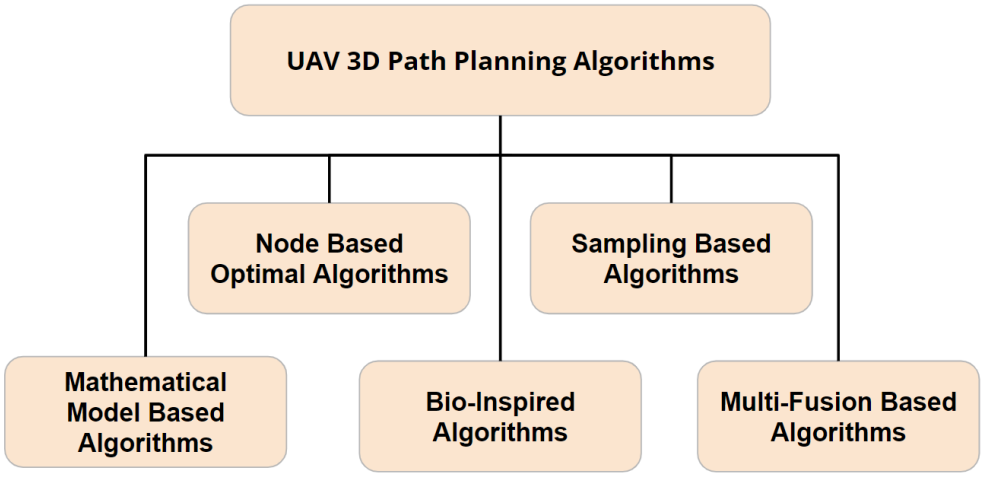


Figure 2.1: Taxonomy of 3D path planning algorithm taxonomy [6].

into an occupancy grid where obstacles are presented as inaccessible grid points. The algorithm searches through the available grid points to find the optimal path from the start to the goal positions if such a path is possible [8]. Under the condition that the grid resolution of the environment is sufficient, this kind of method can guarantee a solution [7]. Node-based methods can be combined with other methods to achieve optimal path planning, as in [9], where the A* algorithm is combined with the Voronoi map construction method to generate minimum-length Paths for UAVs.

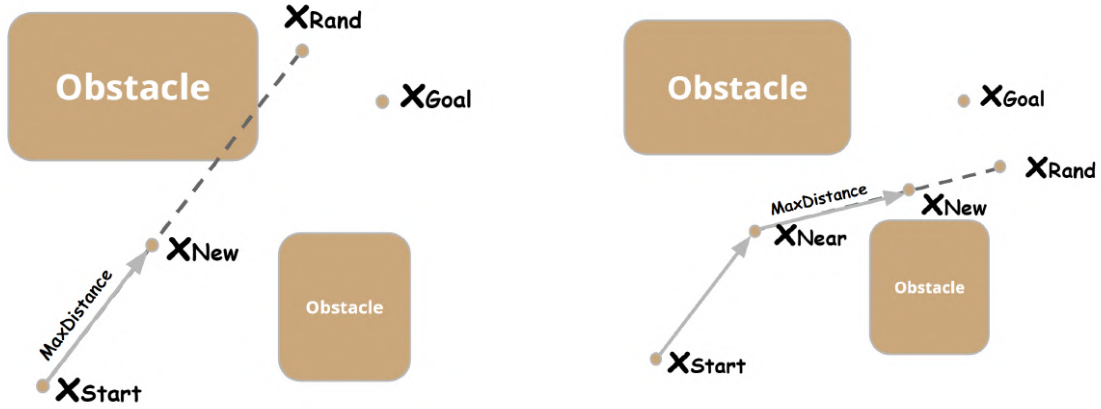
Both methods have been widely used to solve the problem of flight path planning. However, the need for Node-based algorithms to grid the environment in advance becomes very computationally expensive when the size and dimension of the state space increases [10]. On the other hand, sampling-based algorithms can quickly explore the whole state space through a continuous expansion of nodes without the need for geometric partitioning of the search area. This makes them more efficient for solving path-planning problems in high-dimensional environments[11] and complex, large-scale spaces, such as urban environments [10].

As a result, we selected the RRT, a sampling-based algorithm, to tackle the problem in this project. In the following sections, we will cover the standard RRT algorithm and one of its variations.

2.2 Rapidly-exploring Random Tree Algorithm

The Rapidly-exploring RandomTree (RRT) method was first developed in 1998 by Professor Lavalle [12] as a potential solution for path-finding problems. The basic RRT searches for the path from the starting node to the goal nodes by growing branches of a random tree in a high-dimensional environment until the tree contains a node in the goal area. The RRT algorithm begins by selecting the starting point as the base node of the expansion tree. It then randomly selects a state x_{rand} in the state space and identifies the point x_{near} on the expansion tree closest to the random sampling point. If these two points can be connected without collision with obstacles, the new sampling point is added to the path tree. The schematic diagram of the RRT algorithm is shown in Figure 2.2.

The pseudo-code of the RRT in **Algorithm** 1 suggested by [13] introduces some essential param-



(a) First iteration

(b) Second iteration

Figure 2.2: Two successive planning iterations of the RRT algorithm

eters of the algorithm; the names of the parameters were changed to align with the terms used in the rest of this work. **SE3** is the sampling space, and **MaxIterations** is the maximum number of iterations to find a path. The **MaxConnectionDistance** is the maximum size of the branch in the exploring random tree. Since x_{rand} can be anywhere in the state space, the node x_{new} will be generated by expanding the tree T by **MaxConnectionDistance** from the closest node to x_{rand}. Finally, **GoalThreshold** is the minimum distance from the tree nodes to consider that the goal is reached.

Algorithm 1 Rapidly exploring Random Tree (RRT)

Initialization: $\mathcal{T}$.root $= x_{\text{start}}$
Input: SE3, x_{start}, x_{goal}
for $i = 1$ to **MaxIterations do**
  $x_{\text{rand}} \leftarrow$ Sample(SE3)
  $x_{\text{near}} \leftarrow$ Nearest($\mathcal{T}, x_{\text{rand}}$)
  $x_{\text{new}} \leftarrow$ Extend($x_{\text{rand}}, x_{\text{near}},$ **MaxConnectionDistance**)
  if Collision Free($x_{\text{new}}, x_{\text{near}}$) **then**
    $\mathcal{T}$.addNode(x_{new})
  end if
  if $\|x_{\text{new}} - x_{\text{goal}}\| \leq$ **GoalThreshold then**
    break
  end if
end for
Output: A path Γ from x_{init} to x_{goal}
=0

2.3 Rapidly-exploring RandomTree Star Algorithm (RRT*)

The basic RRT algorithm suffers from poor guidance and low search efficiency, which leads to a significant increase in path cost. Therefore, literature [14] proposed the Rapidly-exploring RandomTree Star algorithm, an optimal extension of the standard RRT algorithm. This variant of RRT addresses the high path cost issue by enhancing the selection of potential parent nodes and rewiring the nodes to get the shortest path.

When adding a new node to the graph, the parent node is selected to ensure that the new point is reached through a minimum-cost path from the starting point, rather than selecting the closest node to the new point, as in the basic RRT algorithm.

An illustration of how potential parent nodes can be improved is given by [15] in Figure 2.3. In this example, the original path A-B-C-E with the closest parent node was exchanged for the path A-B-E because it is a shorter path from the initial node to the new location.

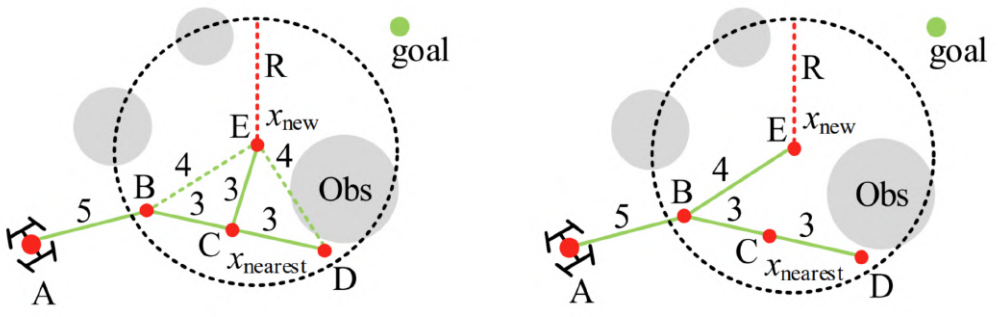


Figure 2.3: Optimised selection of parent nodes in the RRT* algorithm [15].

The rewiring process removes edges that are not part of the shortest path. Using the new node as the parent, the algorithm rewires within a radius R to find the lowest-cost path from the start node to the vertices. In the example of Figure 2.4, the path A-B-C-E is replaced by A-B-D-E after adding the new node for more optimised output.

The pseudo-code of RRT* algorithm is shown in Algorithm 2.

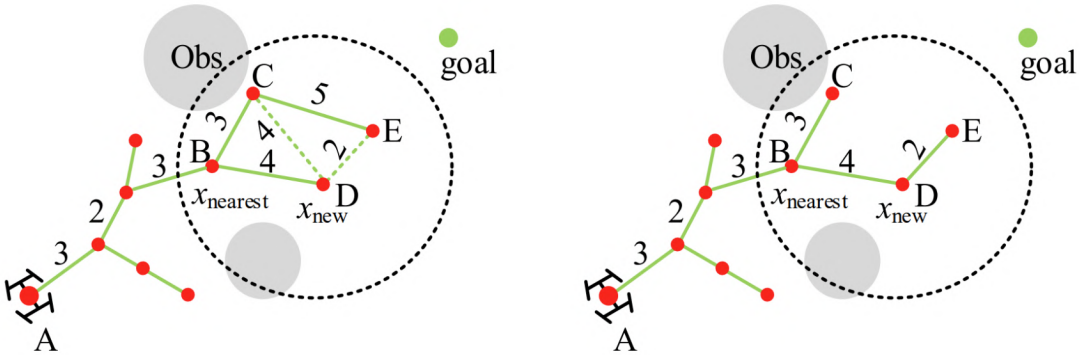


Figure 2.4: Rewiring nodes in the random tree in the RRT* algorithm [15].

Algorithm 2 RRT* algorithm

T.init(x_{start})
Input: SE3, x_{start}, x_{goal}
for $i = 1$ **to MaxIterations do**
  $x_{\text{rand}} \leftarrow \text{Sample}(SE3)$
  $x_{\text{near}} \leftarrow \text{Nearest}(T, x_{\text{rand}})$
  $x_{\text{new}} \leftarrow \text{Extend}(x_{\text{rand}}, x_{\text{near}}, \textbf{MaxConnectionDistance})$
  if Collision Free $(x_{\text{nearest}}, x_{\text{new}})$ **then**
    T.addNode(x_{new})
    $x_{\text{new}} \leftarrow \text{Circular}(T.x_{\text{near}}, R)$
    $x_{\text{pot.parent}} \leftarrow \text{Chooseparent}(T.x_{\text{new}}, x_{\text{near}}, R)$
    $T \leftarrow \text{Rewire}(T, x_{\text{new}}, x_{\text{near}}, R)$
    $T \leftarrow \text{Connect}(T, x_{\text{new}})$
  end if
  if $\|x_{\text{new}} - x_{\text{goal}}\| \leq$ **GoalThreshold then**
    break
  end if
end for
Output: A path Γ from x_{init} to x_{goal} =0

One more enhancement added to the RRT* is goal biasing. This determines how often the goal state is selected during the random state selection process in the search space. Biasing the generation of x_{rand} so that 10% of the time it equals x_{goal} rather than a random location can reduce the computational time for path planning [16]

Although the RRT* algorithm generally yields better paths than the standard RRT algorithm, it still faces several challenges. One major issue is its high level of randomness, which makes it hard to guide the path effectively across the entire feasible region and leads to longer pathfinding times. Additionally, while RRT* aims to create a nearly optimal path, the output is often not the shortest possible. Moreover, the sharp changes in path curvature produced by RRT* can make it difficult for UAVs to follow the paths smoothly. Chapter 4 discusses proposed improvements for a more optimised path planning to address these limitations.

3

Urban Mapping and Scenario Preparation

Contents

3.1 Urban Test Areas: New York and London 11
3.2 UAV Scenarios 12
3.3 3D Occupancy Map 15

The work of this project was implemented on actual maps of urban areas to address the path-planning problem for UAVs in urban environments.

3.1 Urban Test Areas: New York and London

Two urban areas were chosen for the work of this thesis. The main testing area was selected to be a part of New York City as it is considered a challenging environment for aerial navigation due to its skyscrapers. The selected area is a square with a side length of 600 metres and a centre of coordinates 40°45'38"N 73°58'16"W. The geographical coordinates range of this area is given in Table 3.1, and satellite imagery [17]of the area is shown in Figures 3.1a and 3.1c. This testing map will later be referred to as **Scenario 1**.

The second testing area was selected in London, covering the Imperial College London campus in South Kensington. It is also a square with a side length of 600 metres and a centre of coordinates 51°29'55"N 0°10'33"W. The geographical coordinates range of the second testing area is given in

Coordinate Type	Range	Direction
Latitude	40.7528° to 40.7672°	North-South
Longitude	-73.9859° to -73.9601°	West-East

Table 3.1: Geographical Coordinates Range for the selected area in New York City

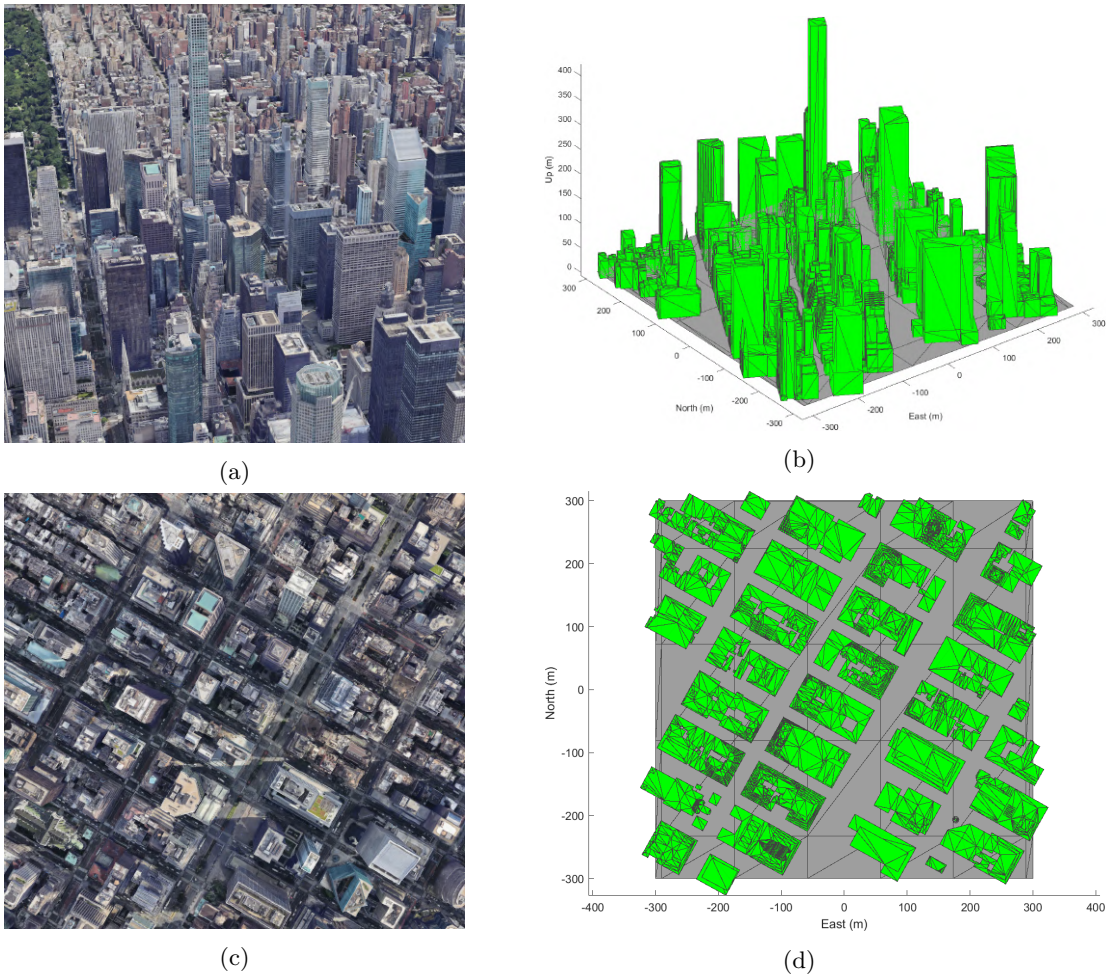
Table 3.2, and satellite imagery [18] of the area is shown in Figures 3.2a and 3.2c. This testing map will later be referred to as **Scenario 2.**

Coordinate Type	Range	Direction
Latitude	51.4953° to 51.5019°	North-South
Longitude	-0.1826° to -0.1627°	West-East

Table 3.2: Geographical Coordinates for Imperial Campus, South Kensington

3.2 UAV Scenarios

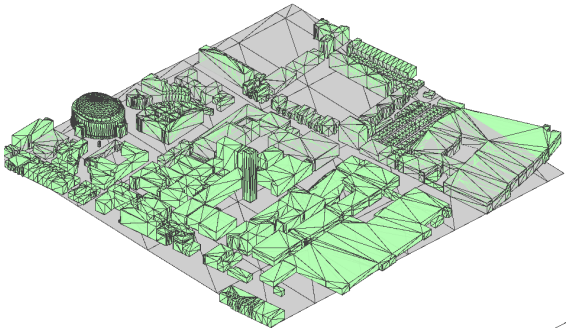
The first step in preparing the map for path-planning algorithms is to create a UAV scenario for each testing map using MATLAB's UAV Toolbox [19]. To do this, the terrain data is first added to obtain the ground elevation of the testing area. We use tiled global terrain derived from the GMTED2010 model by the U.S. Geological Survey (USGS) and National Geospatial-Intelligence Agency (NGA) [20]. Then, the building data of both testing areas was extracted from the OpenStreetMap [21] and added to the corresponding scenario. The UAV Scenarios of the first and second testing areas are shown in Figures 3.1b and 3.1d and Figures 3.2b and 3.2d, respectively. Although the UAV scenario gives a clear presentation of the testing environments, it is not suitable to apply path planning algorithms as they require 3D Occupancy maps. .



(a) (b) (c) (d)

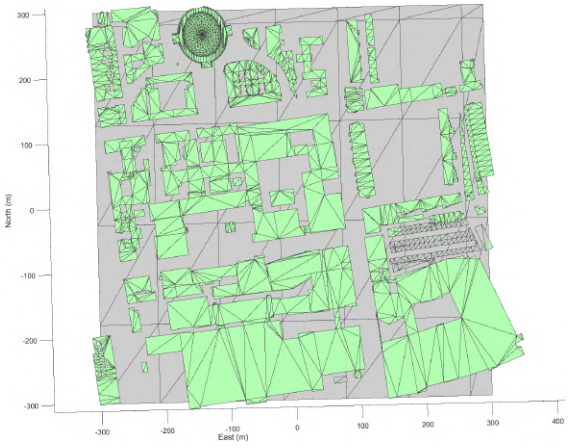
Figure 3.1: Scenario 1: Satellite imagery [17] and UAV scenario

(a)



(b)

(c)



(d)

Figure 3.2: Scenario 2: Satellite imagery [18] and UAV scenario

3.3 3D Occupancy Map

The occupancyMap3D is a MATLAB object [22] that stores a 3D representation of an environment using probabilistic values. This map is built on an octree data structure, which is highly efficient for spatial data management. An octree is a tree-based data structure ideal for representing three-dimensional space. It enhances memory efficiency and computational performance by pruning unnecessary or redundant data [23]. This representation is especially useful for large-scale 3D environments, such as urban environments.

In order to generate 3D occupancy maps of the testing areas, a LiDAR simulation was conducted on the UAV scenarios previously created. The coverage path was generated using the MATLAB function uavCoveragePlanner [24] with the map borders as coverage area. The resulting waypoints from the CoveragePlanner are shown in Figure 3.3.

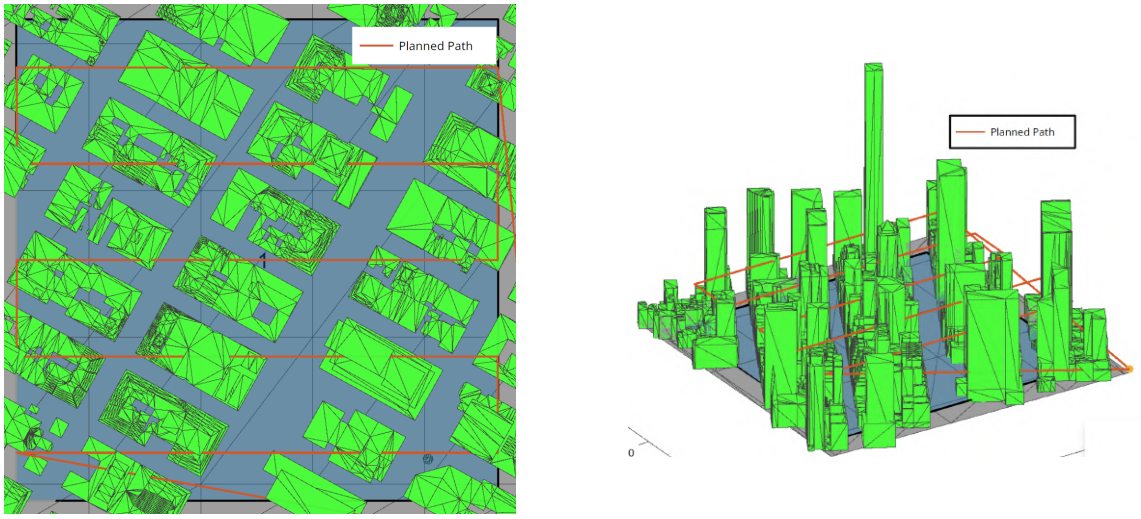
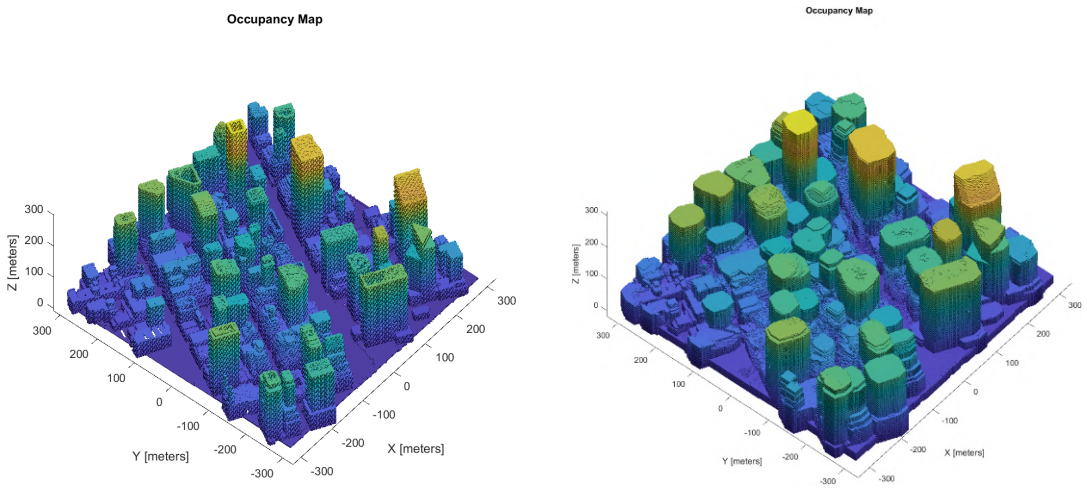


Figure 3.3: Coverage Path Generated for LiDAR Simulation – Scenario 1

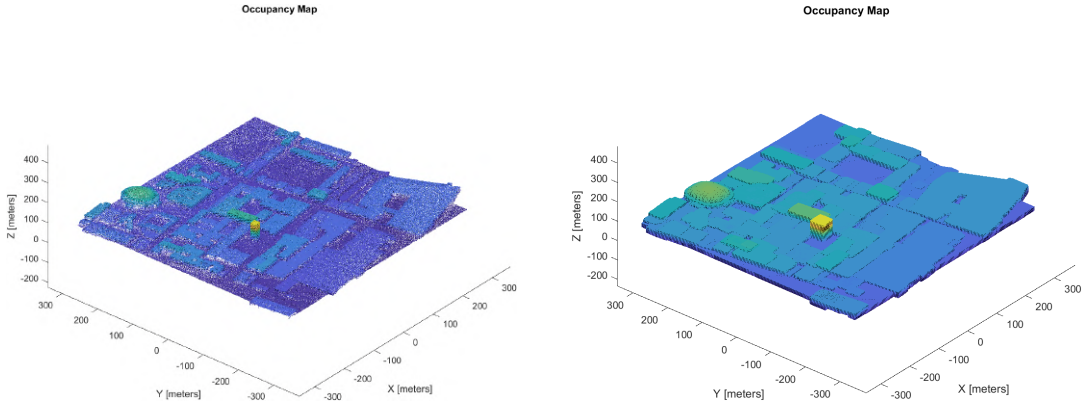
A UAV with a mounted LiDAR sensor was added to the UAV scenarios. A simulation was conducted to collect occupancy data from the LiDAR sensor readings. The data was then stored in a 3D occupancy map with a resolution of 2 metres per cell. The 3D occupancy maps of scenarios 1 and 2, which resulted from the simulation results, are shown in Figures 3.4a and 3.5a, respectively.



(a) (b)

Figure 3.4: Generated 3D Occupancy Map – Scenario 1

Finally, to increase path safety, a buffer area is created around the obstacles by inflating occupied cells (obstacles) by 10m in all directions. This inflation value is variable depending on the size of the biggest UAV in the set. Figures 3.4ba and 3.5b show the inflated occupancy maps.



(a) (b)

Figure 3.5: Generated 3D Occupancy Map – Scenario 2

4 The Improved RRT* Algorithm

Contents

4.1 Restricted Search Area 17
4.2 Path Pruning 19
4.3 The Shortest Path 22
4.4 Path Smoothing 25
4.5 Summary 25

The RRT* algorithm has the potential to solve the problem of path planning in complex high-dimensional space due to its strong space exploration ability. However, the random tree growth in the traditional RRT algorithm is relatively blind, affecting the algorithm's convergence speed and generating longer paths with poor flightability. In the following sections, several improvements are proposed to overcome these shortcomings of RRT* for UAV path planning.

4.1 Restricted Search Area

In this Section, we consider an improvement on the Rapidly-exploring Random Tree Star algorithm (RTT*) to avoid excessive time consumption when searching for a feasible path in an urban environment rich in obstacles.

The random sampling range for the RRT* algorithm is usually the full map range. In some cases, the full map is not needed to find a path between two points and, therefore, sampling the whole map to find the path will result in unnecessary computational efforts. By limiting the sampling area to the area of interest, the random tree can be expanded towards the target direction, thus

improving the search efficiency of the algorithm. Hence, the suggested improvement involves reducing the sampling range to a variable rectangle based on the start and target points. The vertices of the rectangle are decided as follows:

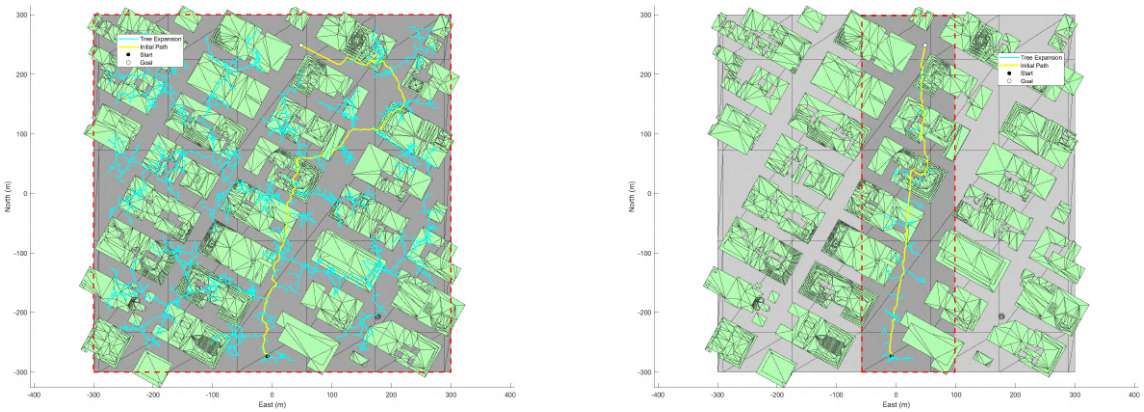
$$X_{\min} = \min(X_{\text{start}}, X_{\text{goal}}) + M, \tag{4.1}$$

$$X_{\max} = \max(X_{\text{start}}, X_{\text{goal}}) + M, \tag{4.2}$$

$$Y_{\min} = \min(Y_{\text{start}}, Y_{\text{goal}}) + M, \tag{4.3}$$

$$Y_{\max} = \max(Y_{\text{start}}, Y_{\text{goal}}) + M \tag{4.4}$$

where M is a variable value that decides the tightness of the restricted rectangular area. These values are then used to form the four vertices $(X_{\max}, Y_{\max})$, $(X_{\max}, Y_{\min})$, $(X_{\min}, Y_{\max})$, and $(X_{\min}, Y_{\min})$. This improvement allows higher search efficiency by reducing the sampling range. When no possible path is found within the limited range, the search range is expanded to cover the whole map. Two examples are shown in Figures 4.1 and 4.2 for a single UAV path planning in an area of the New York map. The map is a square with a side length of 600 metres. The start-target points in examples 1 and 2 are [-7.4 -273.8 100; 48.7 248.5 100] and [-279.9 -35.1 150;279.9 27.3 150] respectively. The RRT* algorithm was applied with a maximum connection distance of 10 meters and a maximum number of iterations of 5000, and the sampling area is shown in the red dashed rectangle.

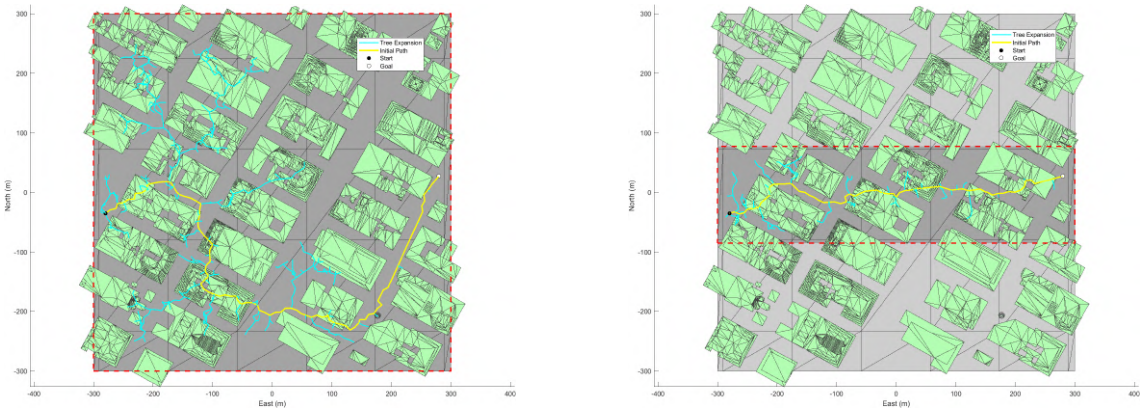


(a) Full map Sampling Range

(b) Reduced Sampling Range

Figure 4.1: RRT* Output Comparison: Full vs. Reduced Sampling Range – Example 1

In Figure 4.5a, we observe that the basic RRT* algorithm results in broader tree expansion with a larger number of sampling points x_{rand} compared to the improved RRT* shown in Figure 4.5a. The output path is also longer when using the basic RRT*. The comparison results in Table 4.1 show that the basic RRT* took 20.9 seconds and 6769 iterations to find a path of 930 m, which



(a) Full map Sampling Range (b) Reduced Sampling Range

Figure 4.2: RRT* Output Comparison: Full vs. Reduced Sampling Range – Example 2

is far from the optimal path between the start and the target. On the other hand, the improved algorithm only required 0.21 seconds to find a shorter path between the same start-goal pair. The simulation results for the second example in Figure 4.2b and in Table 4.1 confirmed these findings, with the improved algorithm producing a shorter path in less time.

	Example 1		**Example 2**	
Sampling Range	**Full map**	**Reduced**	**Full map**	**Reduced**
Path Cost (m)	930	620	1030	689
Time Cost (s)	20.9	0.21	0.33	0.12
Nb of iterations	6769	638	1287	479

Table 4.1: Table comparing path costs, time costs, and number of iterations for different examples and sampling ranges.

Comparing the two algorithms regarding path length, computation time, and the number of iterations demonstrates the efficiency of reducing the sampling range in improving the algorithm in terms of path length and computational time.

4.2 Path Pruning

Even though the RRT* is an effective and computationally efficient tool for path planning in complex environments, the solution is far from optimal due to its random space exploration. The algorithm's randomness results in returning some unnecessary waypoints in the path, which requires removing them.

Different approaches were developed to remove intermediate waypoints between two distant positions if this does not result in a collision with the environment. In [25], multi-stage Dijkstra's

algorithm is used to remove redundant waypoints from RRT*-generated paths and to generate a near-optimal path. Still, this method is computationally expensive, especially when working on multiple paths and in high-dimensional state space. Some other algorithms prioritise reducing the changes in the heading angle rather than getting the shortest path while removing intermediate points [26]. In this project, a new method was proposed to eliminate unnecessary points from the path.

Based on the fact that the shortest path between any two points in Euclidean geometry is a straight line connecting them, the simple but quite efficient proposed method can quickly find a path with the least number of waypoints possible to reach the goal. Let the original path of nodes from the start to the goal point be denoted $\{x_1, \ldots, x_N\}$, such that x_1 is the start point and x_N is the goal point. Let the pruned path initially be set with only x_1, and let $i = 1$ and $j = N$. The pruning operation is as follows: For each j decreasing from N to $i + 1$, check the line between (x_i, x_j) for a collision and stop at the first x_j without collision. Add x_j to the pruned path, let $i = j$, $j = N$, and repeat the process until $i = N$. The pseudo-code of this algorithm is shown in Algorithm 3.

Algorithm 3 Removal of Redundant Points

Input: $\sigma = \{x_1, x_2, \ldots, x_{n-1}, x_n\}$;
$i \leftarrow 1; j \leftarrow n; \sigma_{Pruned} \leftarrow \varnothing$;
Add x_1 to σ_{Pruned};
while $i \neq j$ **do**
  if CollisionFree(x_i, x_j) **then**
    Add x_j to σ_p;
    $i \leftarrow j$;
    $j \leftarrow n$;
  else
    $j \leftarrow j - 1$
  end if
end while
return σ_{Pruned};
=0

Figures 4.3 and 4.4 show the result of the path pruning algorithm applied to the initial RRT* path for two different pairs of start-goal points. The path in the first example initially has 122 nodes between the starting and target points, but the number of nodes is reduced to only six after the redundant waypoints are pruned.

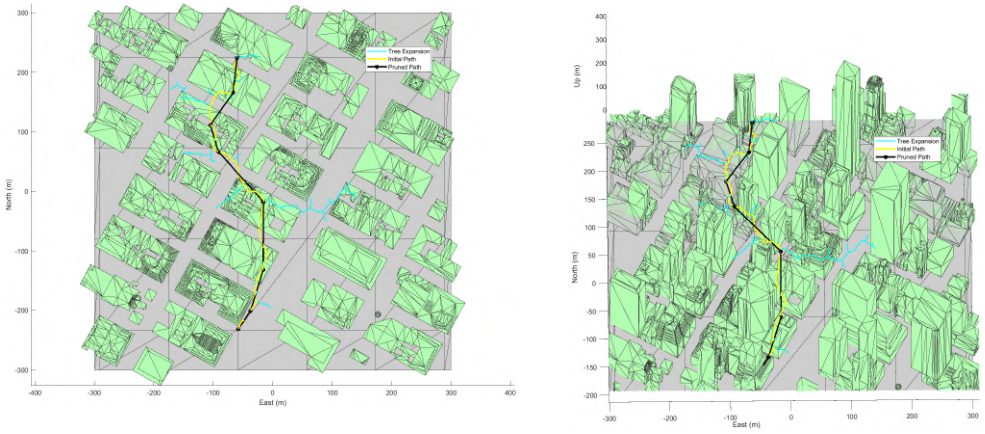


Figure 4.3: Path pruning - example 1

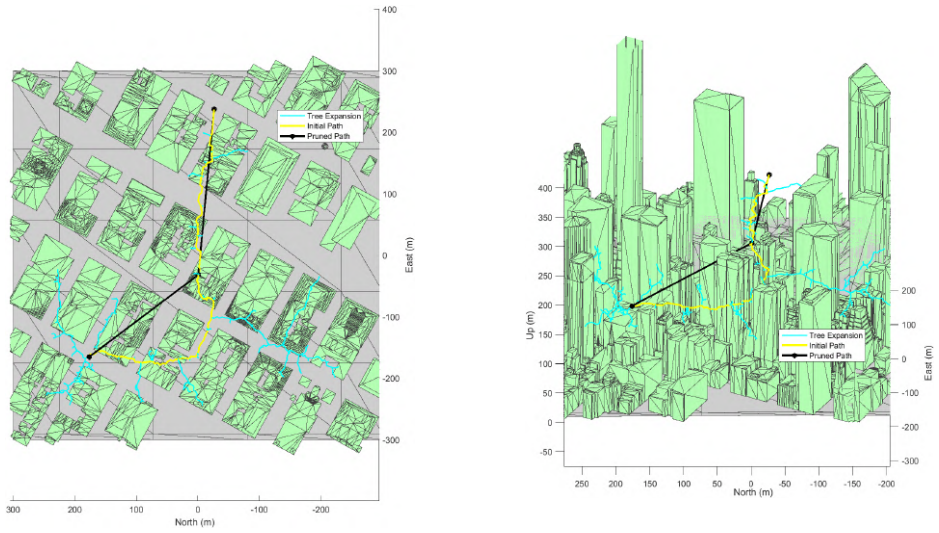
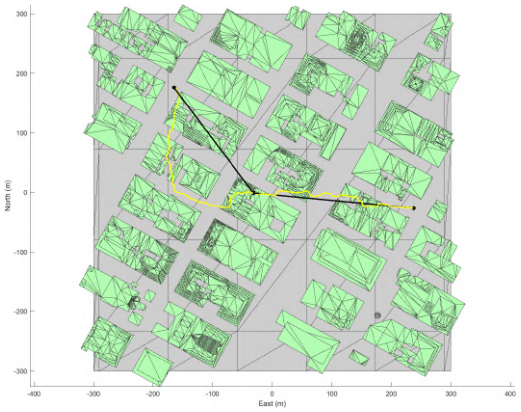
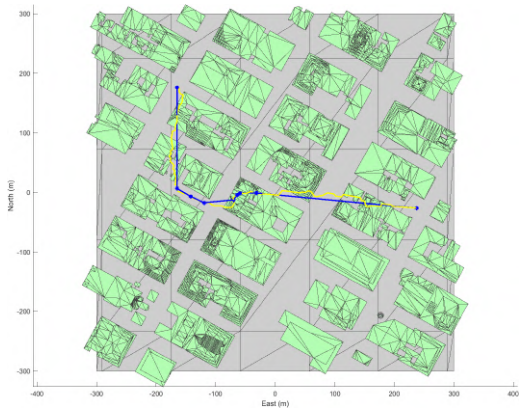


Figure 4.4: Path pruning - example 2

A similar algorithm used in MATLAB's smoothing function [27] works by checking for collisions between the start node and each point along the path to the end node. When a collision is found, the process stops and repeats until the path is pruned. Although this method reduces the number of waypoints, it cannot give the shortest path in cases with a direct collision-free path between the start and the end points, for example. Figures in 4.5 show a comparison in results between the proposed algorithm and the one used in the MATLAB function.



(a) Output of the proposed pruning algorithm



(b) Output of MATLAB pruning function

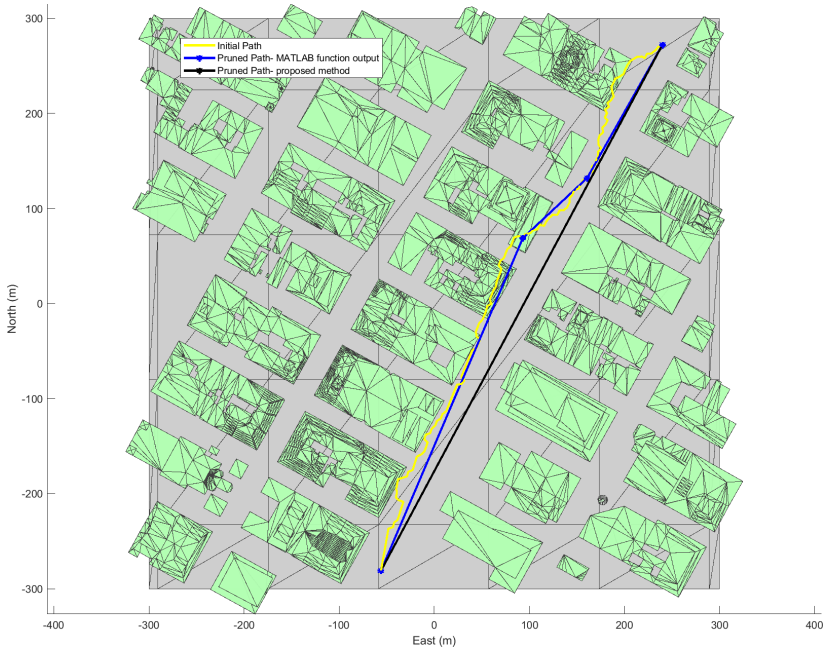


Figure 4.5: Comparison of results between proposed pruning algorithm and MATLAB smoothing function

4.3 The Shortest Path

Removing redundant points is a big step toward getting an optimal path. However, due to the random nature of the rapidly exploring random tree star, one path is not necessarily the shortest that the algorithm can produce. Another improvement was introduced to the RRT* algorithm to get closer results to the optimal path. The idea involves regenerating other paths for the same start and goal point by changing the random seed for every output. Then, the resulting paths are compared based on their length, and the path with the shortest distance becomes the algorithm's output. An input parameter was introduced to define the number of iterations to get

the path between two points. Although this approach will decrease the planning speed, it ensures the shortest output from a sampling-based path planning algorithm. Figures in 4.6 show the output of 5 iterations and the shortest path among them for two different pairs of start-goal points.

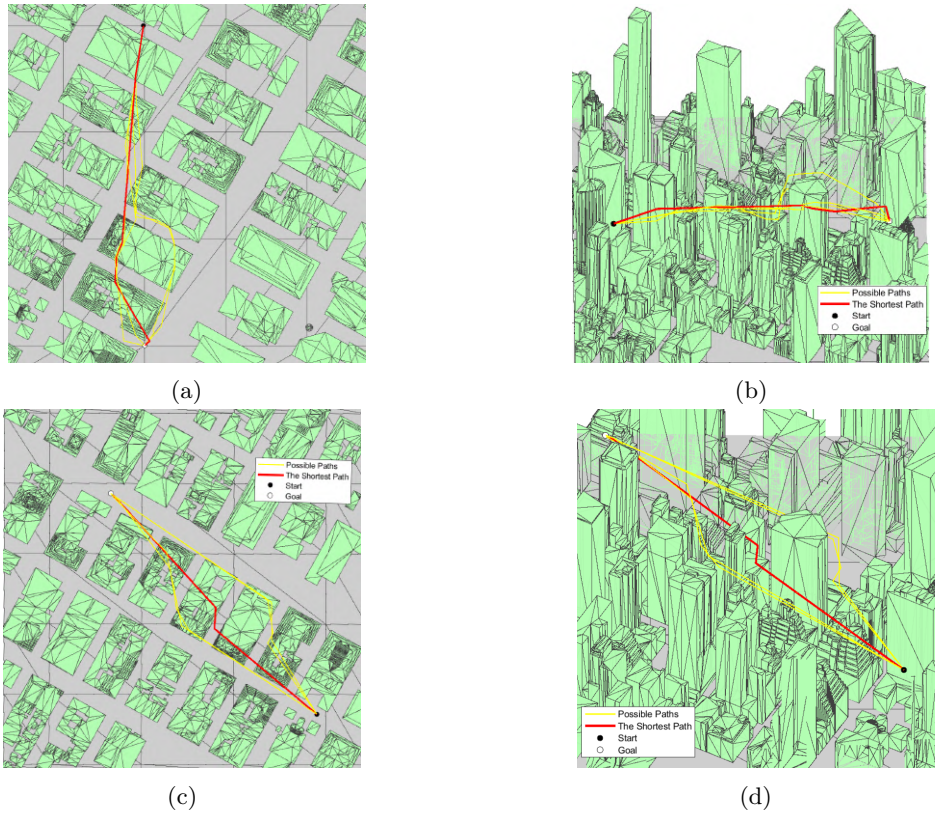
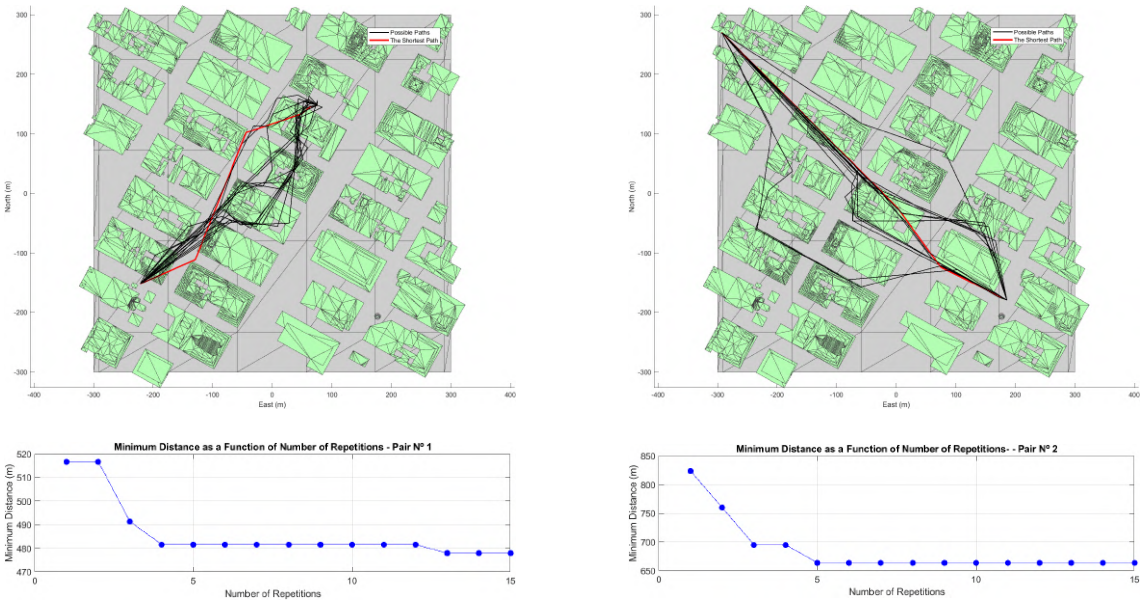


Figure 4.6: RRT* output from 5 iterations showing all possible paths and the shortest path selected for two start-goal pairs

Figure 4.7 displays the minimum path distance as a function of the number of iterations for four different start-goal pairs listed in Table 4.2. The results indicate that regenerating multiple paths by changing the seed improves the algorithm's performance, and that using five iterations yields the most effective path planning results. This number of iterations was used in the following work in the project.

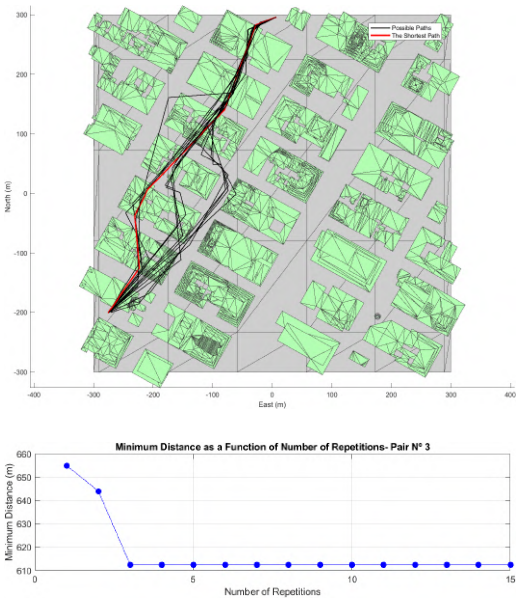
Start			**Goal**		
-222	-152	100	76	148	100
-293	270	100	186	-179	100
7	296	100	-276	-201	100
-282	22	100	261	130	100

Table 4.2: XYZ coordinates of Start-Goal pairs used for simulation results in Figure 4.7

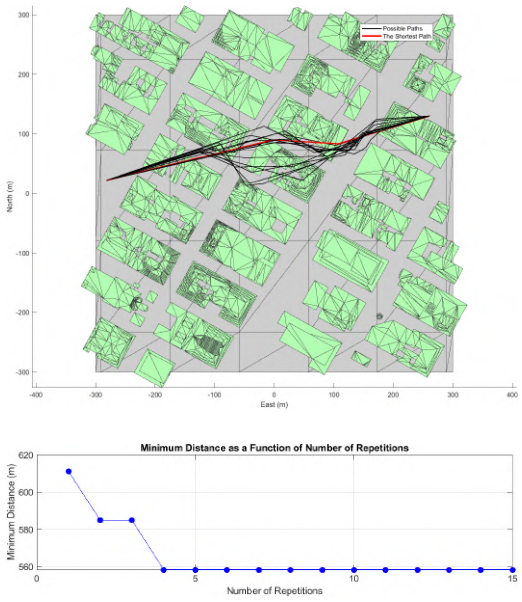


(a)

(b)



(c)



(d)

Figure 4.7: Minimum path distance versus number of iterations for different start-goal pairs

4.4 Path Smoothing

At this stage, the output path obtained from the algorithm is the best result in terms of path distance. However, the output path is piece-wise linear and characterised by edgy turns, making it not followable for a UAV. Therefore, it is necessary to make the path smooth to be suitable for UAVs dynamics. Different smoothing solutions for paths generated from sampling-based algorithms are presented in literature [28]. Dubin's algorithms are commonly applied for path smoothing in UAV path planning, as [29] uses this method to redistribute the position of waypoints obtained with the A*-based search process.

In this project, the smoothing function "*hobbysplines*" presented in [30] is used to get more flyable paths for UAVs. This function helps create smooth 3D cubic splines by only requiring key points that the curve should pass through, which are the pruned waypoints in the case of this work. The function eliminates the need for manually specifying Bezier control points by implementing Hobby's algorithm [31], which computes them automatically. It also allows for adjustments to the tension at each point to fine-tune the curve's shape. This approach simplifies creating smooth curves in 3D, making it highly useful in smoothing and enhancing flightability in UAV path planning applications. Figure 4.8 shows the resulting paths after applying the smoothing function hobbysplines with a tension value of 2 for two different pairs of start-goal points.

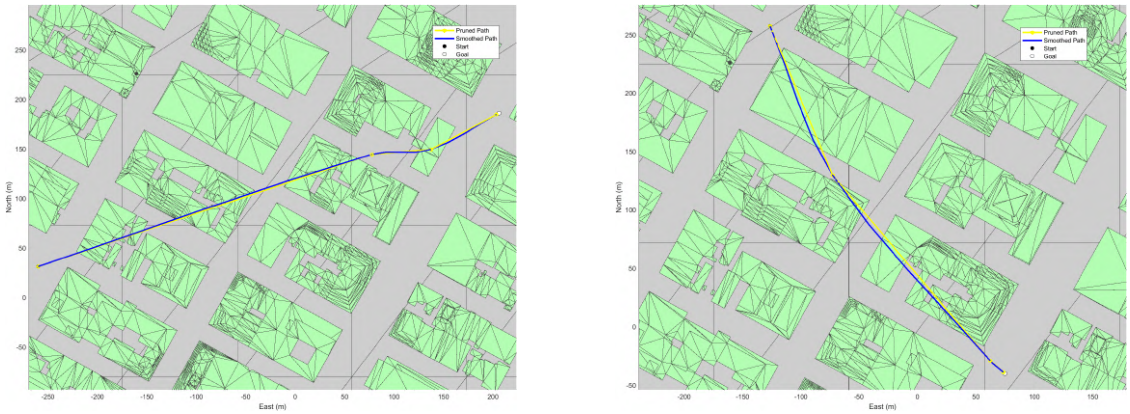
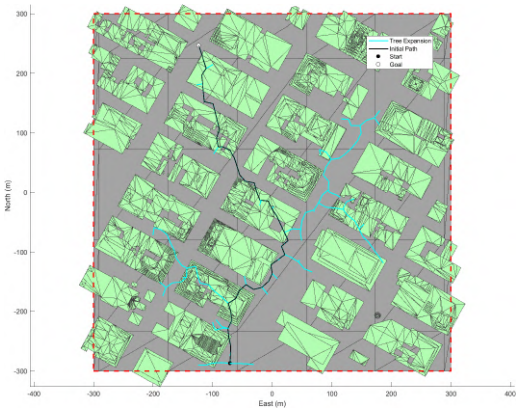


Figure 4.8: Output paths after applying the *hobbysplines* path smoothing function

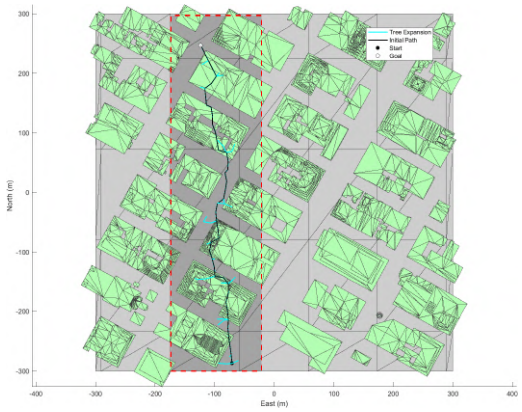
4.5 Summary

Several improvements were made to achieve a near-optimal UAV path from the RRT* algorithm. First, the sampling area was restricted to enhance computational effort and output quality. Then,

the path was pruned to remove all the redundant waypoints. The path generation process was repeated a number of times while changing the random seed to find the shortest path. Finally, the path was smoothed into 3D cubic splines to obtain a more feasible and realistic path. The figures below show all the steps in improving the RRT* path finding algorithm.

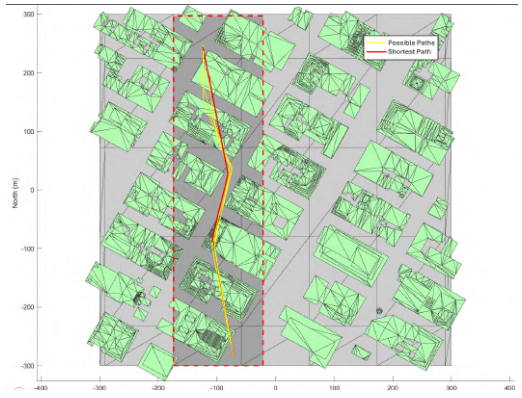


(a) Full Map Sampling Range

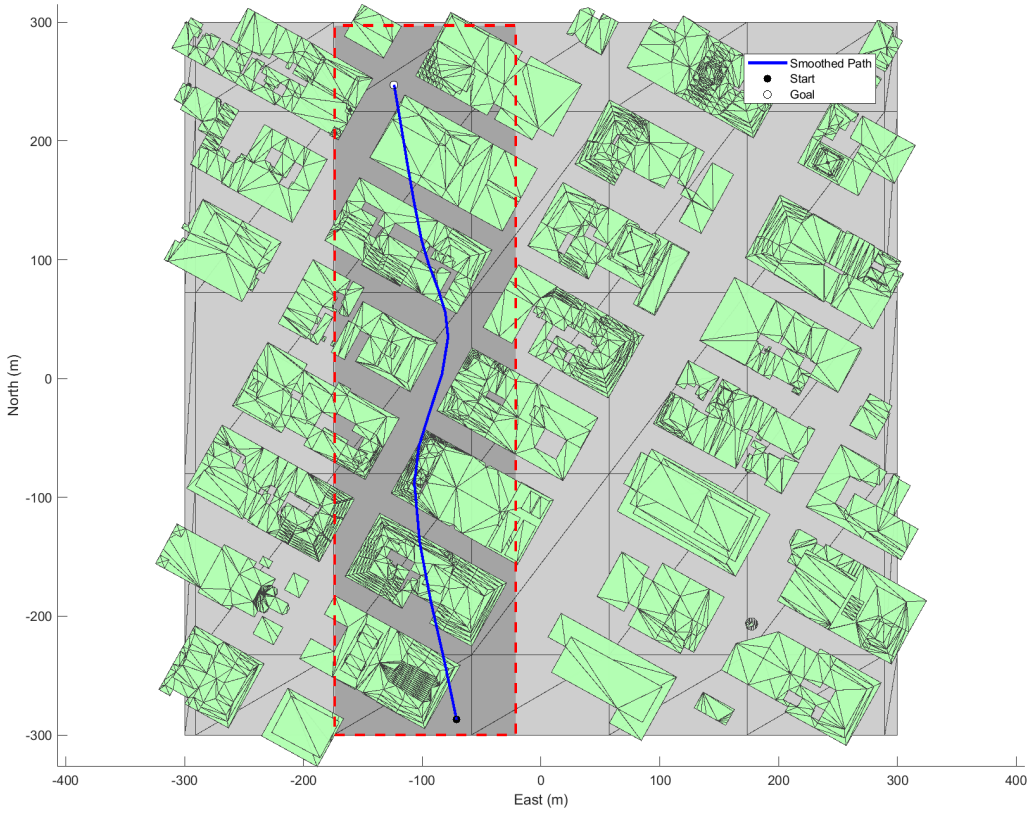


(b) Reduced Sampling Range

(c) Pruning



(d) Shortest Path



(e) Smoothing

Figure 4.9: The different improvements in the RRT* algorithm for single UAV path planning

4

5

Multi-UAV Path Planning - Algorithm

Contents

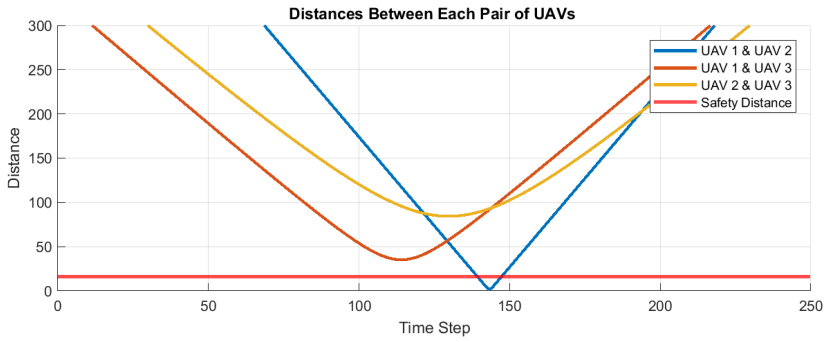
5.1 Generating Initial paths 29
5.2 Detecting Danger Points 31
5.3 Placing Pseudo-Obstacles 31
5.4 Local regeneration of safe path segments 32
5.5 User Guide 34

In the previous section, the RRT* algorithm was optimised to generate an improved path for solving the problem of path planning for a single UAV in urban environments. However, when planning for multiple UAVs, an additional complexity arises: maintaining a predefined safety distance between all the UAVs. To solve this problem, a path editing algorithm is developed, and it will be explained in the following sections.

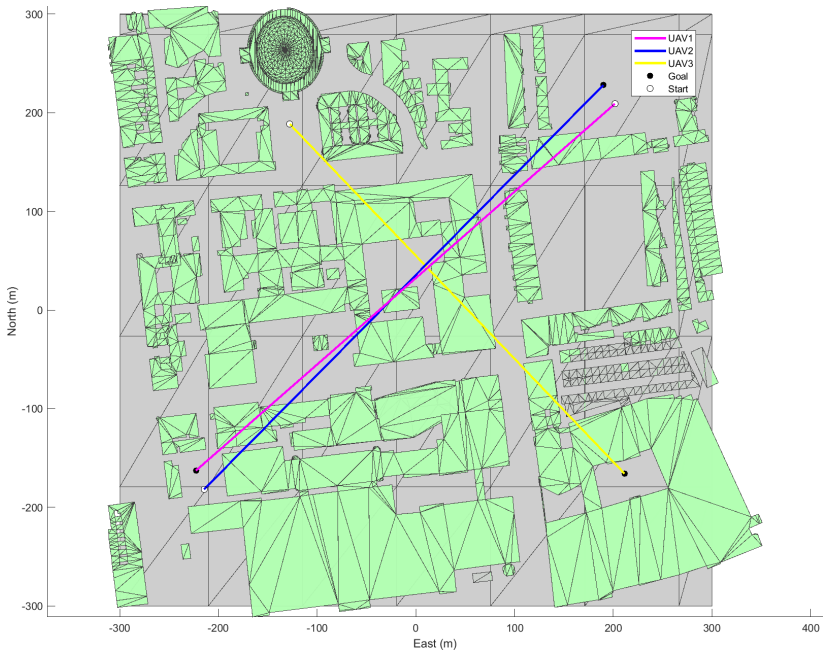
5.1 Generating Initial paths

The first step in generating collision-free paths for multiple UAVs is to create initial paths by applying the improved single UAV path planning algorithm for each UAV separately. Three different pairs of start-goal points were selected on the map of the Imperial College London campus (scenario 2). The single UAV path planning algorithm was then applied to all the pairs to generate

valid paths from the start to the goal points that avoid static obstacles on the map. The path planning results are illustrated in Figure 5.1b. The paths are then discretised with the same step to get their positions at every instant. Information on the speed of each UAV is needed for this step. Figure 5.1a shows the distances between each of the two UAVs of the group. By defining the safety distance with a value of 15 m, it is obvious from the distance plot that there are potential collisions between UAV1 and UAV2 with the closest distance of 1.4 metres. Thus, a path editing algorithm is required to ensure collision-free paths, avoiding collisions with other UAVs and static obstacles in the environment.



(a)



(b)

Figure 5.1: Initial paths for 3 UAVs in scenario 2 using the improved single UAV path planning algorithm

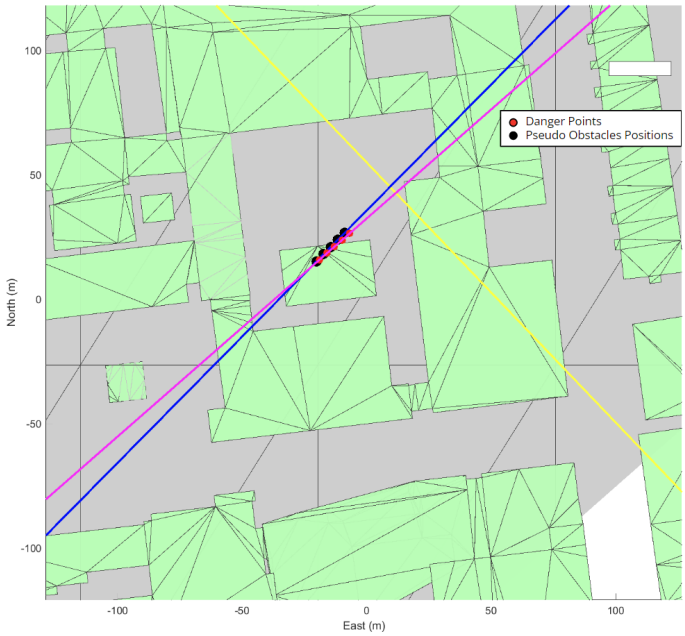


Figure 5.2: Danger points and pseudo-obstacles positions from the path editing algorithm for UAV1

5.2 Detecting Danger Points

The process of editing the initial paths begins by identifying potential collision points between UAVs on each path. These danger points are defined as the points of a path where the distance to the other paths at the same time instant is less than the safety distance. In the previous example and by taking the case of the path of UAV1 (in magenta), the detected danger points are illustrated in Figure 5.2.

5.3 Placing Pseudo-Obstacles

Once a danger point is identified on a path, all points from other paths that are within twice the safety distance from this danger point, at the same time instant, are added to the Obstacles_positions list. To simulate potential collisions, cubic pseudo obstacles of a side length of twice the safety distance and centres of the points in the Obstacles_positions list were added to the 3D occupancy map as occupied cells. Figure 5.2 shows the danger points of path 1 in red and all the points of the Obstacles_positions list in black. Figure 5.3 illustrates the placement of the pseudo obstacles in scenario 2. These pseudo obstacles are temporarily added to the occupancy map during the path editing process for each specific path and removed once the editing of that path is complete. This

ensures that these pseudo obstacles do not influence the editing of other paths, as each is subject to its unique set of pseudo obstacles.

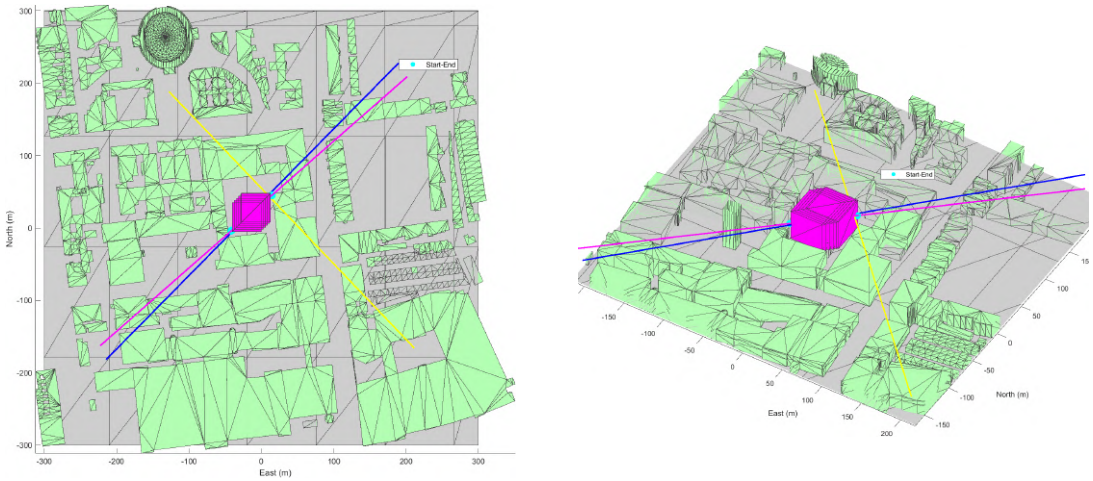


Figure 5.3: Placement of pseudo-obstacles for path editing of UAV1

5.4 Local regeneration of safe path segments

The next step in the algorithm is to define the segments of the path that need to be replaced. These path segments are identified by a start and an end point as shown with cyan in Figure 5.3. New collision-free segments are generated using the improved RRT* algorithm between start-end pairs. These path segments connect the start and end points while respecting a safety distance from the other UAVs. The regeneration of this new segment has no risk of getting closer to any other paths as the points of all the other paths within a distance of twice the safety distance are presented in the occupancy map as pseudo obstacles. So, the points of the newly generated segment cannot get closer than the safety distance from the other path. Once these safe path segments are generated, they are replaced in the initial path to create a collision-free path that respects a minimum safety distance from other UAVs' paths. The steps of the path editing algorithm are illustrated in Figure 5.4.

As can be noticed from Figure 5.5a, the edited path has sudden changes in direction to avoid the pseudo-obstacle. Two options were considered as possible solutions to address this issue. The first one was to run the pruning function developed in the previous chapter on the new path to remove any redundant waypoints resulting from the editing process. However, this method does not take into account the positions of other UAVs while pruning and may risk getting closer to them again. Therefore, the second solution involves expanding the path segment to be edited

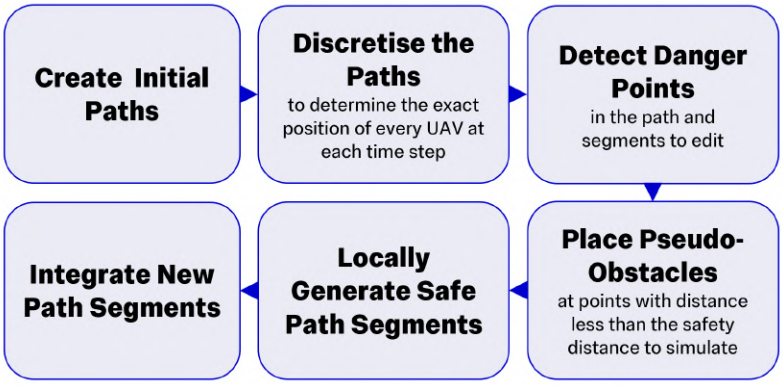
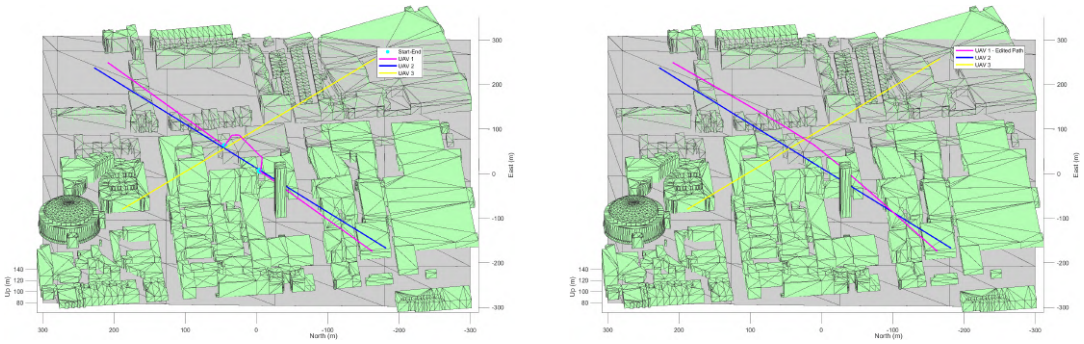


Figure 5.4: Sequential steps of the path editing algorithm

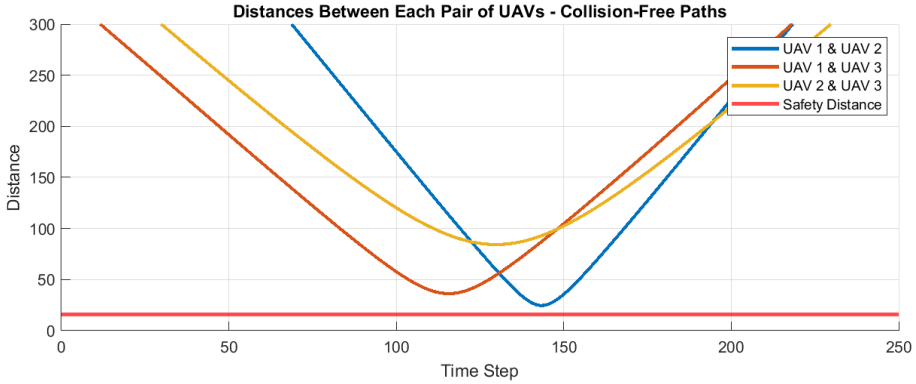
from both sides. This allows the path planning algorithm to find an alternative path to avoid the pseudo-obstacles without drastically changing the path direction. The range for the points of other paths to be added to the Obstacles_positions should increase according to the extension of the segment to be edited.



(a) (b)

Figure 5.5: Initial (a) vs. enhanced (b) versions of the edited path

Finally, the whole path is smoothed using the smoothing function implemented in the previous chapter to create a more natural path for a UAV. The final results are shown in Figures 5.5b and 5.6a.



(a)

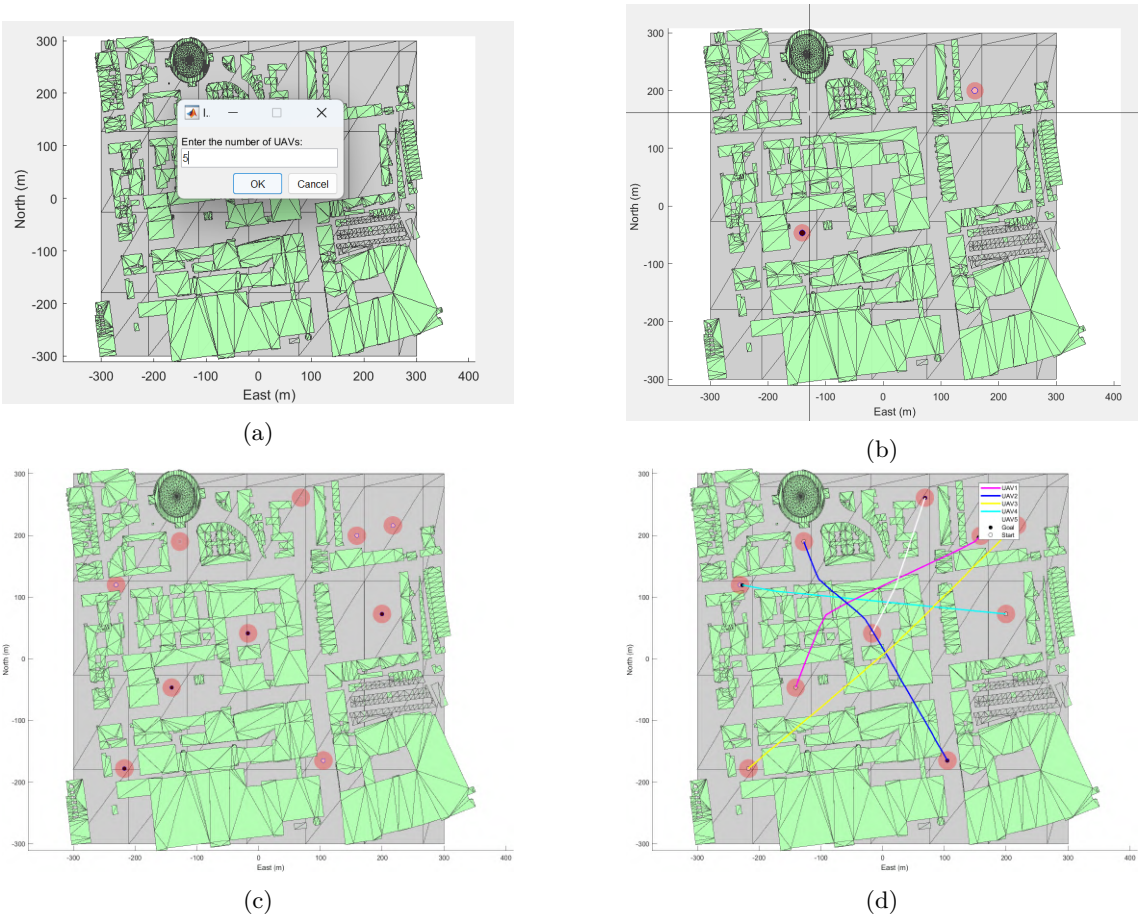
(b)

Figure 5.6: UAV1 collision-free path after applying the path editing algorithm

5.5 User Guide

Two tools have been designed to help users implement the proposed algorithm in this project. The first tool helps select the number of UAVs and their start and goal positions. First, users are asked to input the number of UAVs for which the algorithm will generate the required paths. They can then directly select the start and goal points on the map interface. The altitude for each selected point is automatically adjusted to the lowest available free cell on the map. Once these parameters are set, the final path is generated and displayed on the map. The process of selecting the starting and goal positions is shown in Figure5.7. The second tool is designed to generate a 3D UAV scenario and a 3D occupancy map for any desired area following the methods explained in Chapter 3. The **Get_3DPathPlanningEnv** function takes as inputs the OpenStreetMap data, the dimensions of the area and its geographical coordinates' centre. As output, the function

provides both the 3D occupancy map and the 3D UAV scenario, which can be used to implement the proposed multi-UAV path planning algorithm.



(a) (b) (c) (d)

Figure 5.7: Selection of the number of UAVs and their start and goal positions using the designed tool

5

6

Multi-UAV Path Planning-Simulation

Contents

6.1 Scenario 1: New York **38**
6.2 Scenario 2: Imperial College London Campus **42**

In this chapter, we present simulation results to demonstrate the performance of the proposed algorithm. The multi-UAV path planning algorithm developed in the previous section is applied in the two scenarios prepared in Chapter 2: an area of New York City and the Imperial College London campus area in South Kensington, London. All algorithms were implemented in MATLAB and ran on a computer with a 3.30 GHz processor and 16GB RAM on a Windows operating system.

The parameter settings for the path planning algorithm in both scenarios are as follows:

- Maximum connection distance in the exploring random tree: 15 metres
- Maximum number of iterations for the path search: 5000
- Minimum distance to be considered within the goal region: 5 metres
- Number of iterations to obtain the shortest path: 5
- Smoothing tension value: 2
- Safety distance: 25 metres
- UAV speed (assumed constant): 20 m/s

6.1 Scenario 1: New York

Ten different start-goal pairs were selected on the map of scenario 2, as shown in Figure 6.1. The coordinates of these points are listed in Table 6.1.

UAV	Start (X, Y, Z)			Goal (X, Y, Z)		
UAV1	-60	224	95	-58	-232	100
UAV2	-207	-218	88	157	124	100
UAV3	185	-183	100	-235	70	80
UAV4	-15	120	90	-90	-223	100
UAV5	11	182	85	193	186	80
UAV6	-293	270	100	186	-179	90
UAV7	7	296	100	-276	-201	100
UAV8	-45	-278	85	-297	164	95
UAV9	-282	22	100	261	130	100
UAV10	241	265	80	26	-281	50

Table 6.1: 3D coordinates of start and goal points in Figure 6.1 for the simulation in Scenario 1

The initial path planning results are presented in Figure 6.2a, and the distance plots for each pair of UAVs are shown in Figure 6.2b. To ensure clarity, the distance plot only shows lines for pairs where the separation between the two UAVs falls below 50 metres at any point during their trajectories. The plots show potential collisions involving four pairs of UAVs: UAV1 & UAV2, UAV1 & UAV3, UAV2 & UAV2, and UAV6 & UAV7 as the distance at certain points along their paths is less than the required safety distance.

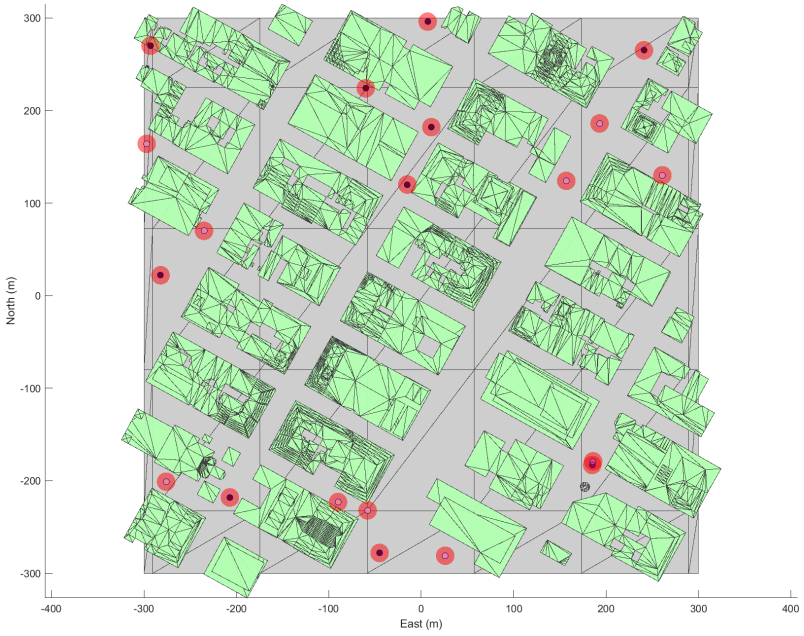
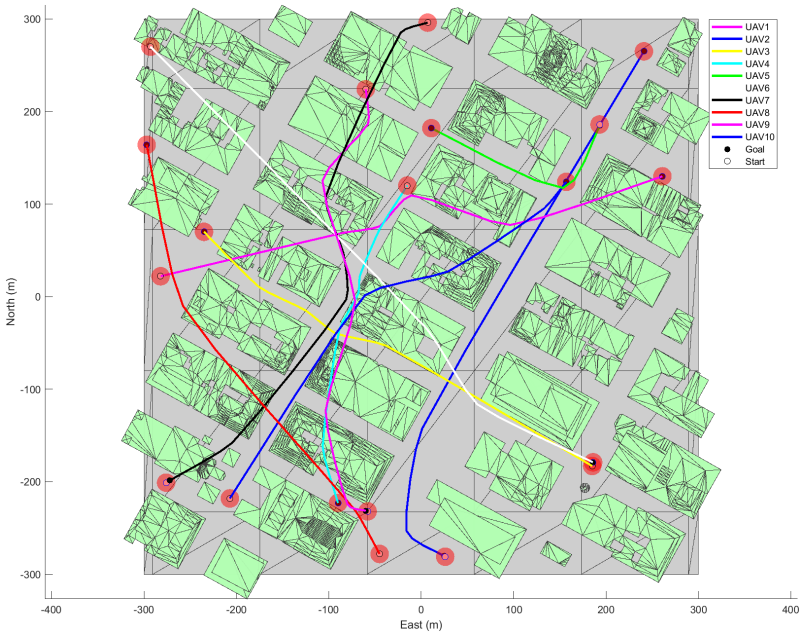
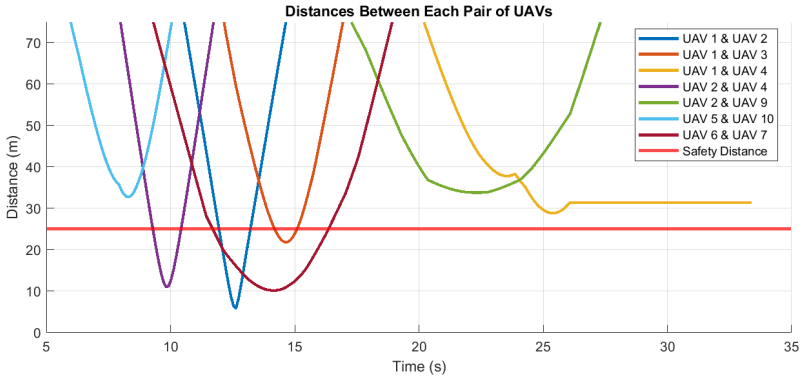


Figure 6.1: Selected start-goal pairs for the simulation in Scenario 1

The results of the path editing algorithm are illustrated in Figures 6.3a and 6.3b, which present the final collision-free paths. The corresponding distance plot in Figure 6.3c confirms the safety of these paths, as each UAV maintains the required safety distance from all other UAVs throughout its trajectory. The initial planning phase for ten paths required 8.95 seconds, while the path editing phase took 40.08 seconds. Only 3 of the ten paths were edited (UAV1, UAV2, and UAV6). The path-planning process for the 10 UAVs was completed in 49.03 seconds, including 22 seconds allocated for inserting and removing pseudo-obstacles within the 3D Occupancy map.

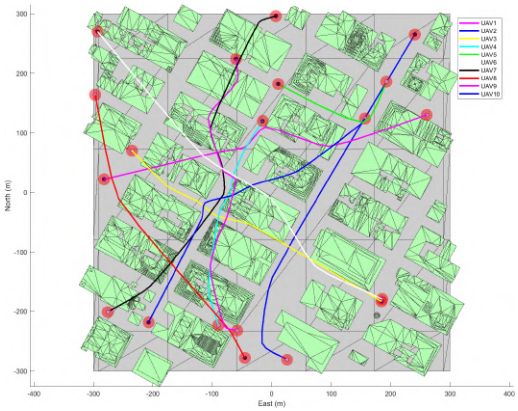


(a)

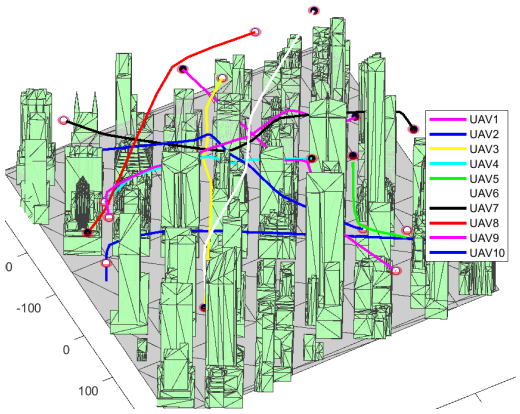


(b)

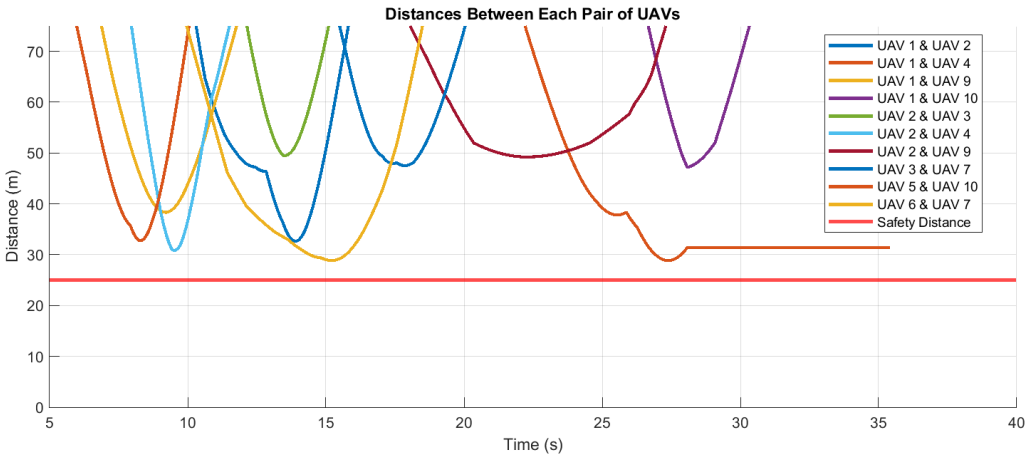
Figure 6.2: Initial path planning output for the ten UAVs in Scenario 1



(a)



(b)



(c)

Figure 6.3: Final path planning output for the ten UAVs in Scenario 1

6.2 Scenario 2: Imperial College London Campus

In Scenario 2, eight distinct start-goal pairs were selected on the map, similar to the arrangement in the first scenario. These pairs are illustrated in Figure 6.4, and their coordinates are provided in Table 6.2.

UAV	Start (X, Y, Z)			Goal (X, Y, Z)		
UAV1	257.1	257.3	93.5	-249.6	-253.0	88.9
UAV2	-266.5	-212.0	89.1	237.8	203.0	92.9
UAV3	214.9	23.2	90.9	-213.4	82.4	92.4
UAV4	-206.2	25.7	111.7	207.6	72.7	91.4
UAV5	115.9	-274.7	88.3	12.2	276.6	98.0
UAV6	-71.0	238.0	107.4	241.4	-169.8	98.9
UAV7	-131.4	-78.1	91.1	73.7	242.8	95.4
UAV8	-119.3	166.8	95.4	46.0	-177.0	89.7

Table 6.2: 3D coordinates of start and goal points in Figure 6.4 for the simulation in Scenario 2

The results of the initial path planning are shown in Fig. 6.5a, with the distance plots between each pair of UAVs shown in Fig. 6.5b. For clarity, the distance plots only include lines where the separation between two UAVs is below 50 metres at any point during their trajectories. These plots show potential collisions between four pairs of UAVs: UAV1 & UAV2, UAV3 & UAV4, UAV4 & UAV8, and UAV7 & UAV8. The distance between these UAV pairs is below the required safety distance at some points along their paths.

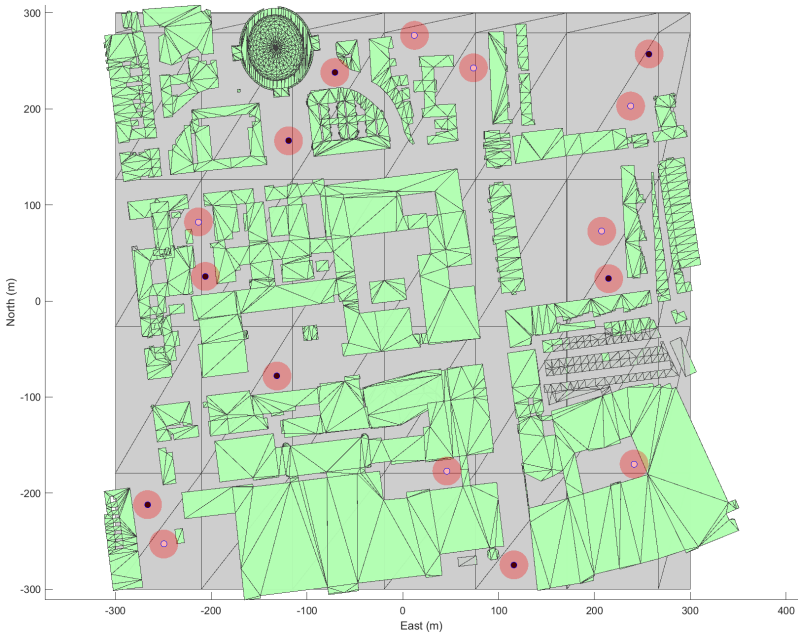
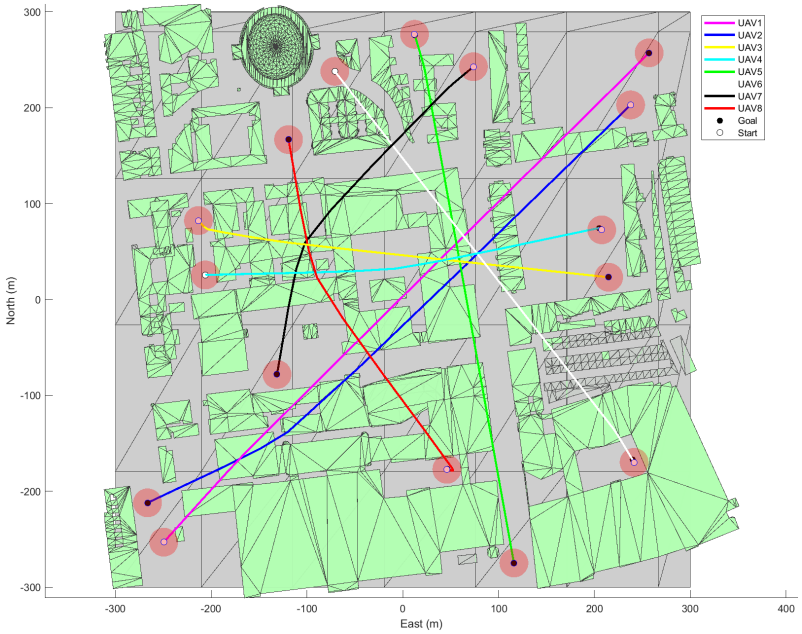
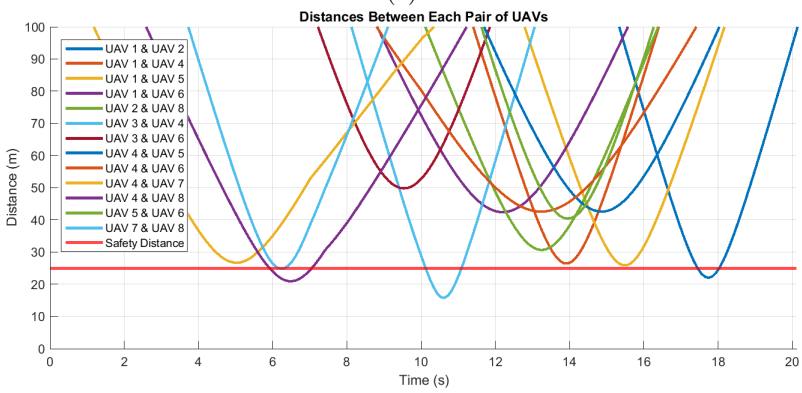


Figure 6.4: Selected start-goal pairs for the simulation in Scenario 2



(a)

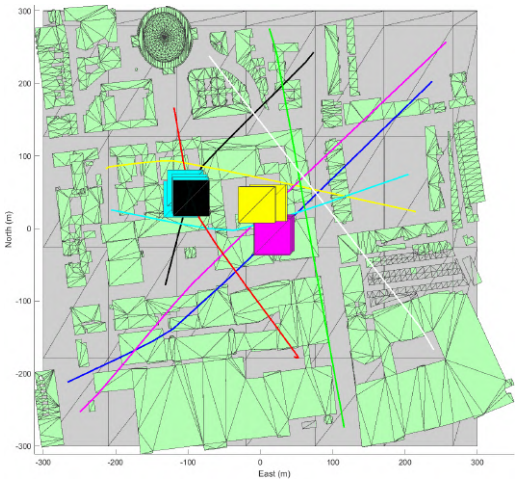


(b)

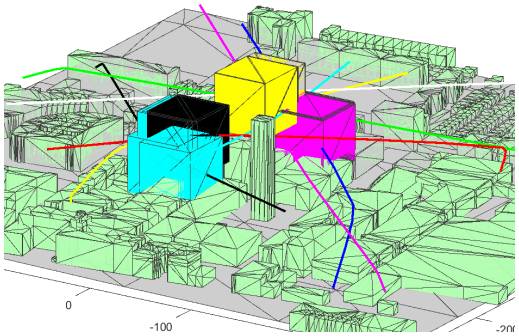
Figure 6.5: Initial path planning output for the eight UAVs in Scenario 2

After running the path editing algorithm, the final collision-free paths are shown in Figures 6.7a and 6.7b. The distance plot in Figure 6.7c confirms the safety of these paths, as each UAV maintains the required safety distance from all other UAVs at every step of its trajectory. The process of adding pseudo-obstacles is illustrated in Figure6.6, with the obstacles colour-coded to correspond to the paths they influence

The initial planning process took 2.458 seconds, while the path editing process took 3.1783 seconds. Only four out of eight paths were edited (UAV1, UAV3, UAV4, and UAV7). In total, the entire path-planning process for eight UAVs took 5.6363 seconds.

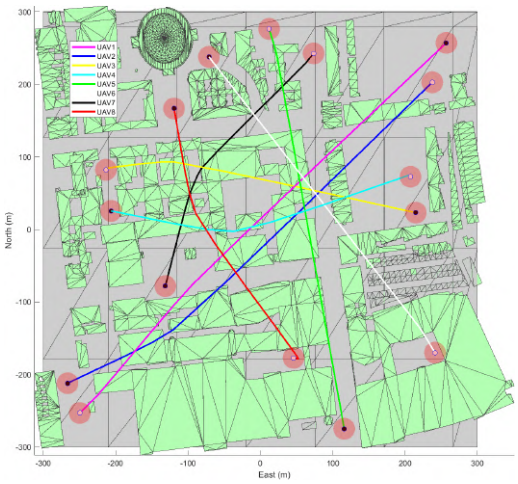


(a)

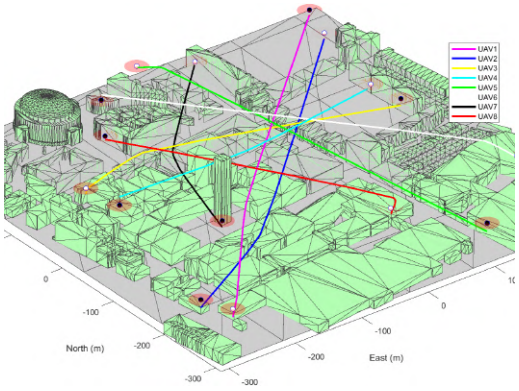


(b)

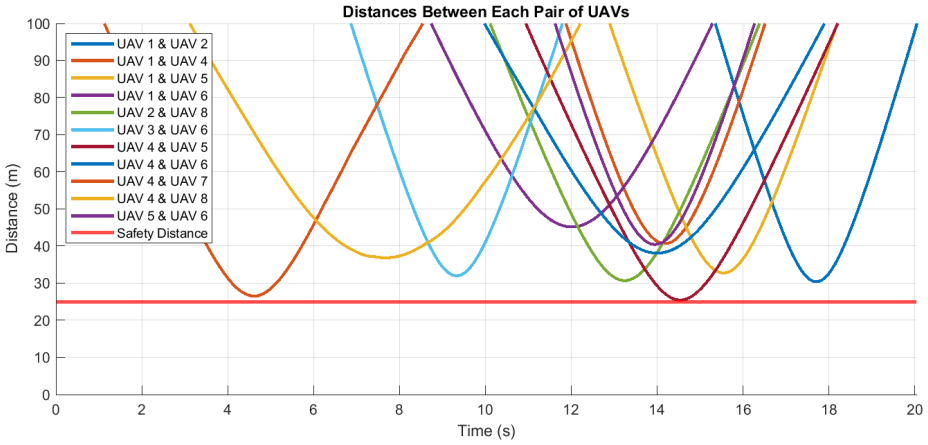
Figure 6.6: Placement of pseudo-obstacles for path editing of the eight UAVs in Scenario 2



(a)



(b)



(c)

Figure 6.7: Final path planning output for the eight UAVs in Scenario 2

Conclusions

This thesis presented a novel framework for 3D path planning in urban environments for unmanned aerial vehicles (UAVs), focusing on offline path planning for multiple UAVs in real urban scenarios. An enhanced RRT*-based algorithm was developed to address several limitations of the traditional RRT* algorithm, such as poor directionality and non-smooth paths, making generated paths more suitable for UAV flight in urban environments. These improvements included constraining the sampling area, pruning redundant waypoints, finding the shortest path, and smoothing the path. The improved algorithm demonstrated the ability to generate more optimized paths. Additionally, this research introduced an original path editing algorithm based on implementing pseudo-obstacles in the environment to ensure a minimum safety distance between UAVs throughout their trajectories. This algorithm was validated through simulations in two different real urban environments. Results proved the algorithm's efficiency in generating collision-free trajectories for multiple UAVs, even in dense urban environments. However, at its current stage of development, the proposed algorithm is considered computationally expensive, especially when scaling to larger maps handling multiple potential collisions in the UAVs' trajectories. This opens up new possibilities for future research and further improvements.

6.3 Future Work

While the proposed framework has made significant progress in solving the multi-UAV path planning problem in urban environments, there are still several areas that require further exploration:

- **Improving Computational Efficiency:** Future work should explore optimisations to reduce computational efforts to handle larger-scale maps more effectively.
- **Integration of Dynamic Obstacles:** The current algorithm only accounts for dynamic obstacles created by other UAVs in the group. Future work could expand on this by incorporating real-time sensing and adaptive planning to handle dynamic environments more effectively.

A

Supplementary Materials

A.1 Project Code

All code files related to the algorithms discussed in this thesis and figures and graphs included in the report can be accessed on GitHub. The repository is at the following link: `https://github.com/HamzaSaddour/Path-planning-for-Multi-UAV-.git`.

A

Bibliography

[1] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, and M. H. Alsharif, "Towards the unmanned aerial vehicles (uavs): A comprehensive review," *Drones*, vol. 6, no. 6, -06-15 2022. DOI: 10.3390/drones6060147.

[2] G. Skorobogatov, C. Barrado, and E. Salamí, "Multiple uav systems: A survey," *Unmanned Systems*, vol. 08, no. 02, p. 149, -04-02 2020. DOI: 10.1142/s2301385020500090.

[3] L. E. Parker, "Multiple mobile robot teams, path planning and motion coordination in," in Springer New York, 2009, p. 5783. DOI: 10.1007/978-0-387-30440-3_344.

[4] F. Aljalaud, H. Kurdi, and K. Youcef-Toumi, "Bio-inspired multi-uav path planning heuristics: A review," *Mathematics*, vol. 11, no. 10, -05-18 2023. DOI: 10.3390/math11102356.

[5] A. A. Saadi, A. Soukane, Y. Meraihi, A. B. Gabis, S. Mirjalili, and A. Ramdane-Cherif, *Uav path planning using optimization approaches: A survey*, -04-18 2022. DOI: 10.1007/s11831-022-09742-7.

[6] L. Yang, J. Qi, J. Xiao, and X. Yong, "A literature review of uav 3d path planning," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*, 2014, pp. 2376–2381. DOI: 10.1109/WCICA.2014.7053093.

[7] C. Zammit and E.-J. V. Kampen, "Comparison between a* and rrt algorithms for 3d uav path planning," *Unmanned Systems*, vol. 10, no. 02, p. 129, -09-23 2021. DOI: 10.1142/s2301385022500078.

[8] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, p. 1135, -04 2016. DOI: 10.1109/tits.2015.2498841.

[9] F. Schøler, A. la Cour-Harbo, and M. Bisgaard, *Generating approximative minimum length paths in 3d for uavs*, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:15627186.

[10] Y. Liu, Z.-L. Gu, C. Li, B.-G. Wang, H.-N. Wu, and W.-J. Liu, *Uav path planning algorithm based on improved rrt*, -02-22 2023. DOI: 10.1117/12.2667637.

[11] S. M. LaValle and J. J. Kuffner, *Randomized kinodynamic planning*, May 2001. DOI: 10.1177/02783640122067453.

[12] S. LaValle, *Rapidly-exploring random trees : A new tool for path planning*, 1998. [Online]. Available: https://www.semanticscholar.org/paper/Rapidly-exploring-random-trees-%3A-a-new-tool-for-LaValle/d967d9550f831a8b3f5cb00f8835a4c866da60ad.

[13] J. Tian, T. Chao, M. Yang, J. Zhu, and S. Wang, "A path planning algorithm based on improved rrt* for uavs," IEEE, 2022-10-28. DOI: 10.1109/icus55513.2022.9986963.

[14] S. Karaman and E. Frazzoli, *Optimal kinodynamic motion planning using incremental sampling-based methods*.

[15] X. Jiang and B. Huang, *Global path planning of fixed-wing uav based on improved rrt * algorithm*. DOI: 10.6180/jase.202310_26(10).0009.

[16] S. Karaman and E. Frazzoli, *Sampling-based algorithms for optimal motion planning*, -05-05 2011.

[17] G. Earth, *Satellite imagery of new york city, midtown manhattan*, *Google Earth*, 40.7528° N, 73.9859° W, elevation 0 m. Accessed: Sep. 9, 2024, Available: https://earth.google.com/, 2024.

[18] G. Earth, *Satellite imagery of imperial college london, south kensington*, *Google Earth*, 51.4953° N, 0.1826° W, elevation 0 m. Accessed: Sep. 9, 2024, Available: https://earth.google.com/, 2024.

[19] T. M. Inc., *Uav toolbox (r2024a)*, Natick, Massachusetts, United States, 2024. [Online]. Available: https://www.mathworks.com.

[20] U. G. Survey, *Gmted2010 | u.s. geological survey*, 2024. [Online]. Available: https://www.usgs.gov/coastal-changes-and-impacts/gmted2010#science.

[21] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2017.

[22] T. M. Inc., *Create 3-d occupancy map*, Natick, Massachusetts, United States, 2022. [Online]. Available: https://uk.mathworks.com/help/nav/ref/occupancymap3d.html.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, p. 189, -02-07 2013. DOI: 10.1007/s10514-012-9321-0.

[24] T. M. Inc., *Path planner for uav space coverage*, Natick, Massachusetts, United States, 2024. [Online]. Available: https://uk.mathworks.com/help/uav/ref/uavcoverageplanner.html.

[25] J. N. Amin, J. Bokovic, and R. K. Mehra, "A fast and efficient approach to path planning for unmanned vehicles," in American Institute of Aeronautics and Astronautics, -06-15 2006. DOI: 10.2514/6.2006-6103.

[26] X. Lan and D. Cairano, "Continuous curvature path planning for semi-autonomous vehicle maneuvers using rrt*," 2015-07-85.

[27] I. T. MathWorks, *Motion planning with rrt for fixed-wing uav*, 2024. [Online]. Available: https://uk.mathworks.com/help/uav/ug/motion-planning-with-rrt-for-fixed-wing-uav.html?searchHighlight=UAV%20fixed%20wings%20path%20planning&s_tid=srchtitle_support_results_1_UAV%20fixed%20wings%20path%20planning#MotionPlanningWithRRTForAFixedWingUAVExample-9.

[28] K. Yang, "Real-time continuous curvature path planning of uavs in cluttered environments."

[29] L. D. Filippis, G. Guglieri, and F. Quagliotti, "A minimum risk approach for path planning of uavs," *Journal of Intelligent amp; Robotic Systems*, vol. 61, no. 1-4, p. 203, -11-12 2010. DOI: 10.1007/s10846-010-9493-9.

[30] W. Robertson, *Smooth 3d bezier curves with implicit control points*, 2024. [Online]. Available: https://uk.mathworks.com/matlabcentral/fileexchange/42302-smooth-3d-bezier-curves-with-implicit-control-points.

[31] J. D. Hobby, "Smooth, easy to compute interpolating splines," *Discrete Computational Geometry*, vol. 1, pp. 123–140, Jun. 1986. DOI: 10.1007/bf02187690. (visited on 11/01/2020).