# ABSTRACT

MultiUAV path planning has gained significant attention, driven by the applications of UAVs in commercial contexts. However, prevailing research often limits the full spectrum of real-world constraints inherent in this complex problem. This report investigates efficient path planning of agents navigating urban environments. Each agent is tasked with delivery responsibilities, necessitating movement to starting points and subsequent goal positions while navigating obstacles and avoiding collisions with other agents.

Two distinct approaches are introduced: the Two Step Interplanner and the Two Step Interplanner with Conflict Resolution. These methodologies meticulously consider potential points of conflict and various constraints. Through experimentation, these algorithms are simulated on a pre-generated 3D occupancy map. The latter algorithm is particularly pertinent in scenarios where real-time computation is paramount. Initially tested in a 2D environment with multiple agents assigned specific tasks and goals, these methodologies were extended to a 3D map, accounting for UAV-specific constraints such as maximum roll angle and forward flight speed.

In addition to the proposed approaches, a case study on task reallocation is presented, which is a critical aspect of multi UAV operations. While most existing studies focus on static task allocation, the investigation conducted addresses the dynamic nature of task execution. Unforeseen events like UAV failures or environmental changes can disrupt the original task allocation, impeding task completion. We propose a Partial Reassignment Algorithm (PRA) to solve this challenge. The PRA selectively reallocates tasks among UAVs, optimizing task success rates and alleviating communication and computation burdens. By integrating this algorithm into the framework, the adaptability and robustness of multi UAV operations in dynamic environments is enhanced.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

Multiple Unmanned Aerial Vehicle or MultiUAV systems involve many scenarios, from mapping to search and rescue. In each of these scenarios, the task of control and planning is crucial to ensure the accomplishment of the end goal. Among these scenarios, the problem statement of managing the urban airspace, specifically 'Urban Air Mobility,' is paramount. Urban Air Mobility refers to using small, highly automated aircraft to carry passengers or cargo at lower altitudes in urban and suburban areas developed in response to traffic congestion. It encompasses existing and emerging technologies such as traditional helicopters, vertical takeoff and landing vehicles (VTOLs), and unmanned aerial vehicles (UAVs).[1]

In this context, the rapid growth of drone technology will create corridors of high drone traffic. These corridors will be localized and integral to last-mile delivery systems. The inherently dense network established by this system will lead to exponential demand for airspace resources. Therefore, managing this airspace by investigating planning algorithms to avoid collisions and ensure efficient flights is critical.

## 1.1 Challenges Involved

The implementation of autonomous drones involves a range of intriguing hurdles that need to be tackled before they can be integrated into our daily lives.

A few key challenges are listed here:

1. **Obstacle Detection and Avoidance:** To ensure safe flights, drones need to per-

ceive their surroundings and identify static and dynamic obstacles. This demands sensors like LiDAR, RADAR, and computer vision algorithms to interpret the collected data and make decisions based on it.

2. **Path Planning and Decision Making:** Optimal routes must be planned according to obstacles, goal positions, and mission objectives. Post-planning drones must accurately follow the planned trajectories and be capable of replanning based on any environmental disturbances.

3. **Localisation and Mapping:** Drones must know their precise location to navigate the environment correctly. This task requires GPS and sensor-based localisation that is robust to environmental changes.

4. **Fail-Safe Mechanisms:** Strong fail-safe mechanisms must be integrated into autonomous drone systems to account for vulnerable situations like propeller damage, sensor and motor malfunctions, and software bugs.

5. **Computing Power and Resources:** Onboard hardware needs to be compact and computationally efficient to process sensor data and make quick decisions.

6. **Battery Limitations:** Current battery technology limits the flight times and range of autonomous drones. However, advancement in this technology can improve the efficiency and practicality of missions.

In multiUAV scenarios, all these challenges are applicable. However, the major issue is coordinating the entire system to avoid collisions and plan efficient trajectories for each UAV according to its goal. Therefore, this report focuses on the path-planning aspect of multiUAV systems in Urban Air Mobility scenarios.

## 1.2 Summary of the Report

This report explains the steps required for robot autonomy, reviews literature on planning techniques for UAVs in urban scenarios, and references fundamental path planning algorithms like A-Star (A*), and Rapidly Exploring Random Trees (RRT).[2][3] It then proceeds to mathematically define the problem statement, elaborating on the methods and

resources required to develop a solution. Consequently, it proposes two algorithms, the Two Step Interplanner and Two Step Interplanner with Conflict Resolution, and demonstrates the results using simulations on MATLAB. It also presents a case study on task reallocation, an essential aspect of continuous, reliable, and autonomous multiUAV operations. The last section draws conclusions from the results and presents possible directions for future work.

In the following subsection, applications of autonomous drones in the electrical domain are listed and related to the work on path planning of multiUAV systems.

## 1.3 Applications in Electrical Domain

Autonomous drones can play an essential role in electrical engineering. Multi UAV teams can help with various tasks in the electrical domain, ranging from inspections to surveillance. A few of those applications are listed below:

1. **Inspection of Power Lines and Infrastructure:** Drones equipped with sensors and cameras can be employed for inspecting power lines, transmission towers, and other critical infrastructure in electrical power systems (Fig. 1.1). Efficient path-planning algorithms can optimize drone routes, ensuring comprehensive infrastructure coverage while minimizing the time required for inspection.

**Fig 1.1**: Power line inspection using drones

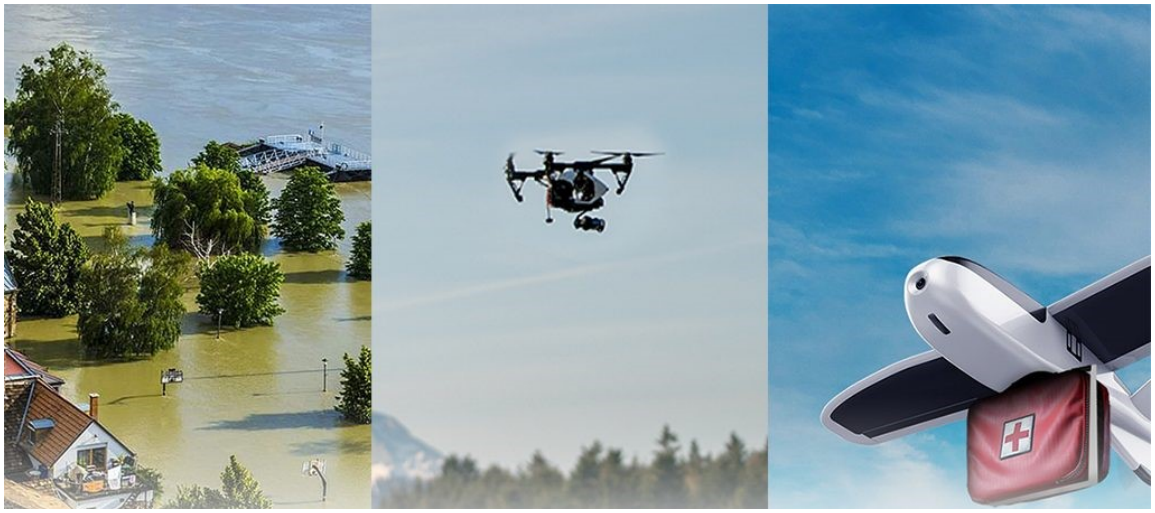


**Fig 1.2**: Drones used for disaster response and recovery

2. **Fault Detection and Maintenance:** Drones can identify faults, damages, or abnormalities in electrical power systems. Drones can systematically traverse the power grid by utilizing path-planning algorithms and inspecting components such as transformers, switches, and substations. Timely detection of faults can facilitate proactive maintenance and reduce downtime.

3. **Environmental Monitoring:** Drones can monitor environmental conditions around power generation facilities, substations, and transmission lines. Path-planning algorithms can enable drones to collect data on factors such as air quality, temperature, and vegetation encroachment, helping utilities mitigate environmental impacts and ensure compliance with regulations.

4. **Security and Surveillance:** Drones can enhance security measures at power generation plants, substations, and other critical facilities. Path-planning algorithms can be utilized to coordinate surveillance patrols, allowing drones to monitor perimeters, detect intrusions, and respond to security threats promptly.

5. **Remote Sensing and Mapping:** Drones equipped with specialized sensors, such as LiDAR or thermal imaging cameras, can facilitate remote sensing and mapping of electrical infrastructure. Path-planning algorithms can optimize flight paths for data collection, enabling utilities to generate accurate 3D models of power lines, substations, and surrounding terrain for planning and maintenance.

6. **Disaster Response and Recovery:** In natural disasters or emergencies affecting electrical power systems, drones can assist in disaster response and recovery efforts (Fig. 1.2). Path-planning algorithms can help coordinate drone fleets to assess damage, locate survivors, and deliver critical supplies to affected areas, restoring power infrastructure.

By integrating efficient path-planning algorithms with drone technology, electrical power systems can benefit from enhanced reliability, safety, and efficiency in operations and maintenance activities. These applications demonstrate the potential of leveraging advanced technologies to address the challenges faced by modern electrical power systems.

# Chapter 2

# Background & Problem Definition

## 2.1 Autonomy

Autonomous navigation of ground or aerial robots involves multiple steps.[4] They can broadly be classified into four steps, as shown in Fig2.1 :

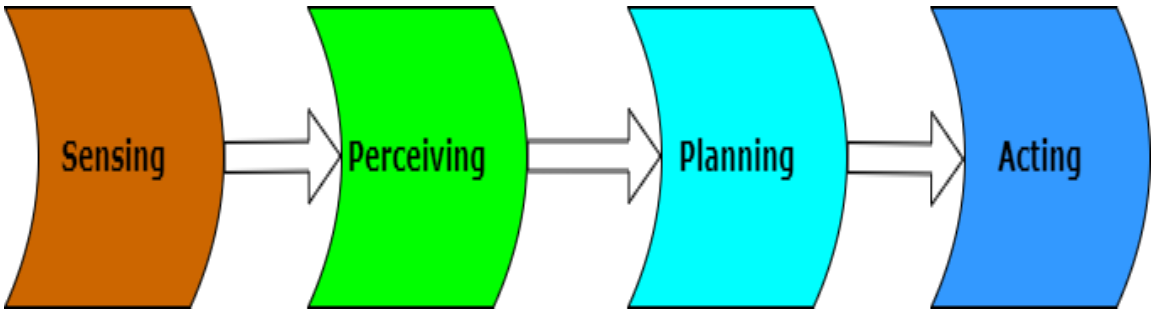



**Fig 2.1**: Four Steps of Autonomy

**Sensing** refers to the task of collecting data from the environment, including the environment's map consisting of static and dynamic obstacles.

**Perceiving** is the primary processing step in autonomy. It involves interpreting the collected data and tracking objects. It also involves building models for further steps like estimating the robot's state and localising it in the environment. Since it involves multiple sub-steps within itself, perception plays a vital role in robot autonomy. The level of perception also determines the degree of autonomy of the robot.

**Planning** refers to developing a path towards the goal. The data processed in the perception step enables us to create an efficient, reliable, and safe path towards the goal. The path parameters can be curated based on other task requirements.

**Acting** means to control the vehicle to follow the path determined in the planning step. This involves using control algorithms like PID or LQR control to allow the robot to follow the desired trajectory accurately.

Based on these steps, approaches for path planning can be divided into two major categories:

1. **Heuristic Approach:** It follows a practical set of rules, and does not require complete information from the environment. However it does not guarantee optimality
2. **Optimal Approach:** It requires more environment information and optimal planning results are obtained

.

## 2.2 A* and RRT Algorithms

Motion planning is a fundamental problem in robotics, where the goal is to compute a collision-free path for a robot from its initial position to a desired goal position within its environment. This task becomes challenging due to obstacles the robot must avoid while navigating. Two popular algorithms used for motion planning are A-Star (A*) and Rapidly Exploring Random Trees (RRT), each with its approach to solving the problem.

### A-Star (A*) Algorithm

A* is a widely used graph search algorithm that finds the shortest path between nodes in a graph. It is particularly suitable for problems where the space of possible solutions is searchable in the form of a graph. In the context of motion planning, the environment is typically represented as a grid or a graph, where each node corresponds to a robot's configuration, and edges represent possible transitions between configurations. An instance of A* is depicted in Fig. 2.2

1. **Heuristic Evaluation:** A* uses a heuristic function to estimate the cost of reaching the goal from any given node. This heuristic efficiently guides the search towards the goal. In motion planning, the heuristic might be based on distance metrics like

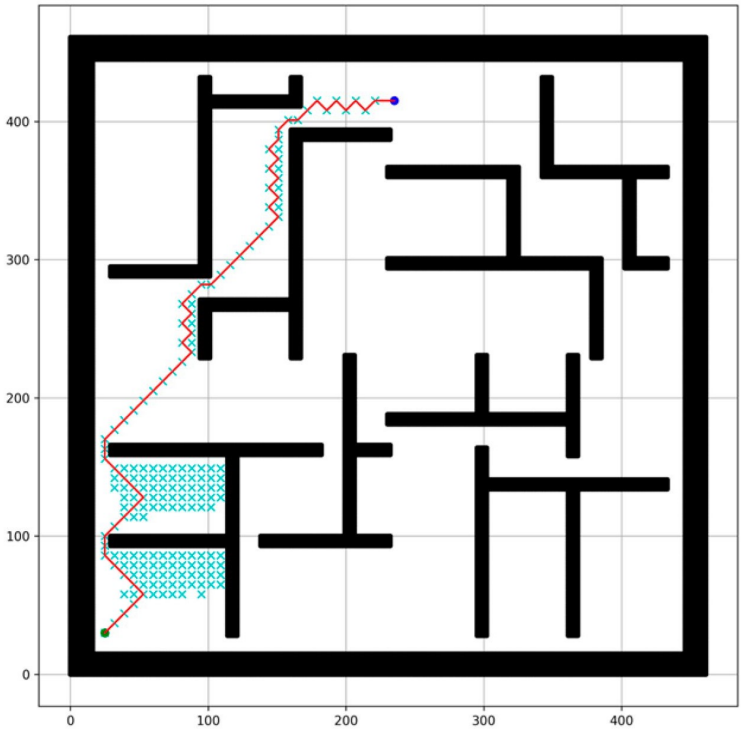



**Fig 2.2**: A* algorithm

Euclidean distance or Manhattan distance between the current configuration and the goal configuration.

2. **Cost Function:** A* maintains a cost function representing the cost of reaching each node from the start node. The cost function is updated as the algorithm explores the graph.

3. **Search Process:** A* conducts a systematic search starting from the initial configuration of the robot and iteratively explores neighbouring configurations. It prioritises configurations based on their estimated total cost, the sum of the cost to reach the configuration from the start and the heuristic cost to achieve the goal from that configuration.

### Rapidly Exploring Random Trees (RRT) Algorithm

RRT is a probabilistically complete algorithm for motion planning in high-dimensional configuration spaces. It incrementally builds a tree structure rooted at the initial configuration, expanding towards unexplored regions of the space as shown in Fig. 2.3.

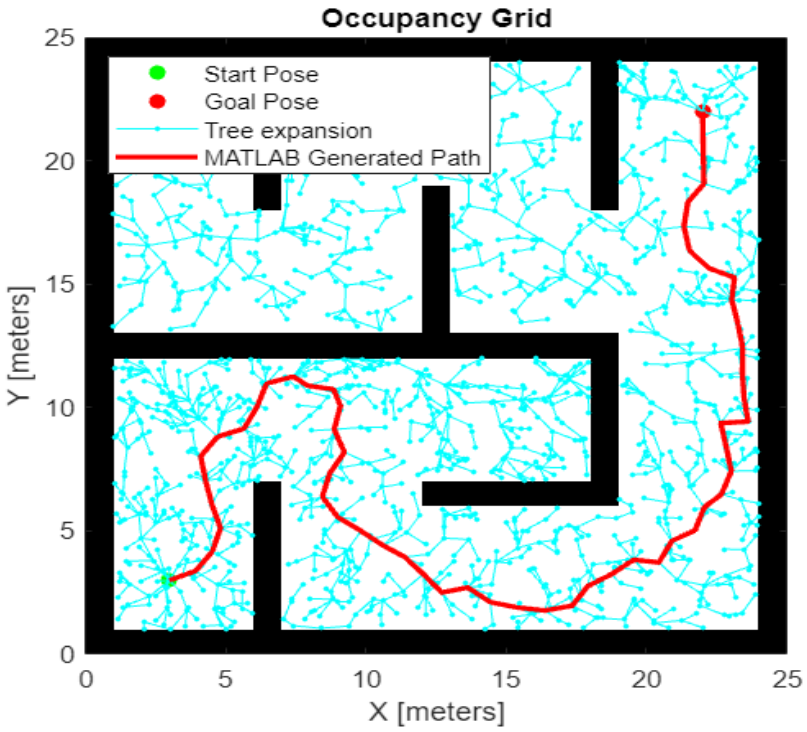



**Fig 2.3**: RRT algorithm

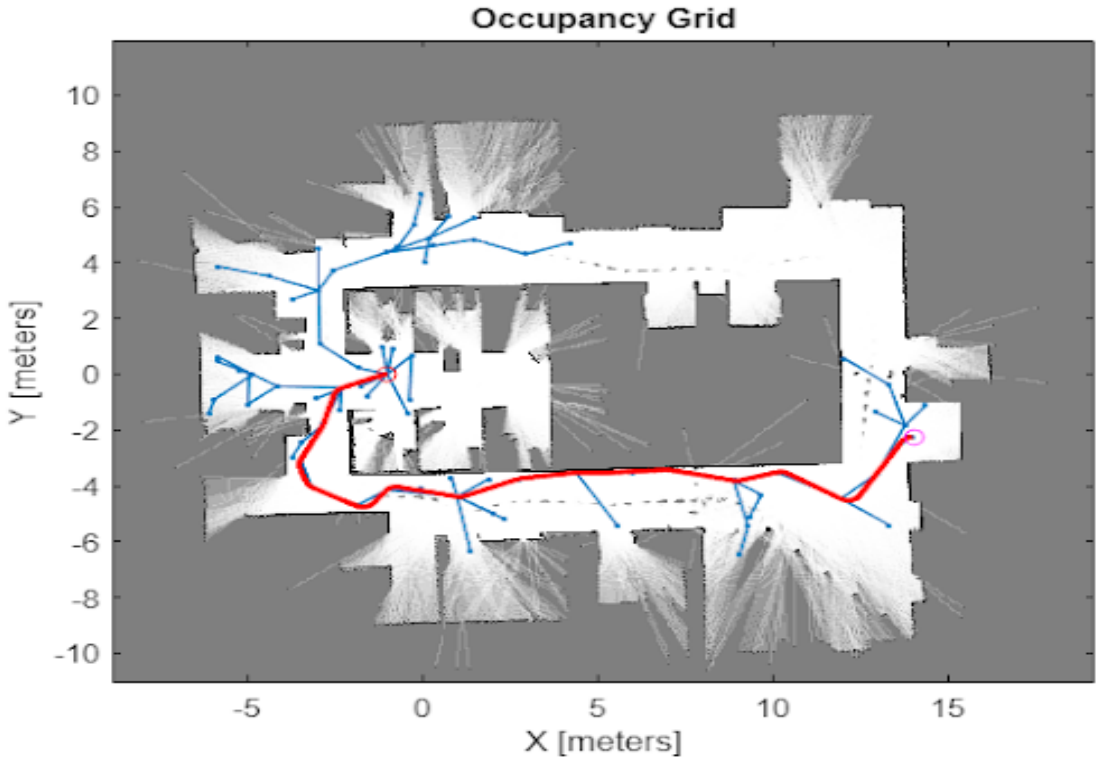



**Fig 2.4**: RRT implementation on ground robot

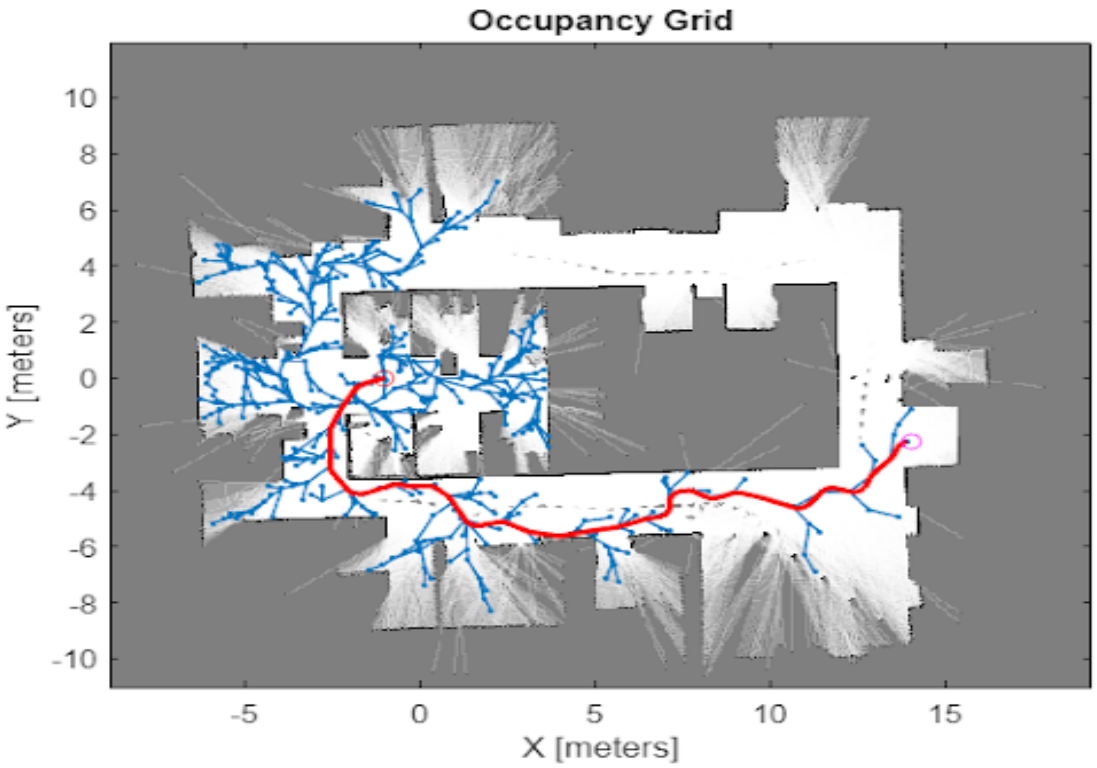



**Fig 2.5**: Smoother path with reduction in maximum inter node distance

1. **Random Sampling:** RRT starts by generating random configurations in the robot's configuration space. These configurations are sampled uniformly or based on some probability distribution.

2. **Tree Expansion:** RRT incrementally grows a tree by connecting each sampled configuration to the nearest configuration in the existing tree. This step ensures that the tree efficiently explores the space, focusing on regions closer to unexplored areas.

3. **Collision Checking:** RRT checks whether the path between the new configuration and its nearest neighbour in the tree is collision-free at each step. The algorithm discards the sampled configuration if a collision is detected and repeats the sampling process.

4. **Goal Biasing:** RRT often includes a bias towards sampling configurations near the goal to guide the exploration towards the goal. This bias helps in quickly finding a path towards the goal.

5. **Termination:** RRT terminates when the goal configuration is reached, or a prede-

fined maximum number of iterations is reached.

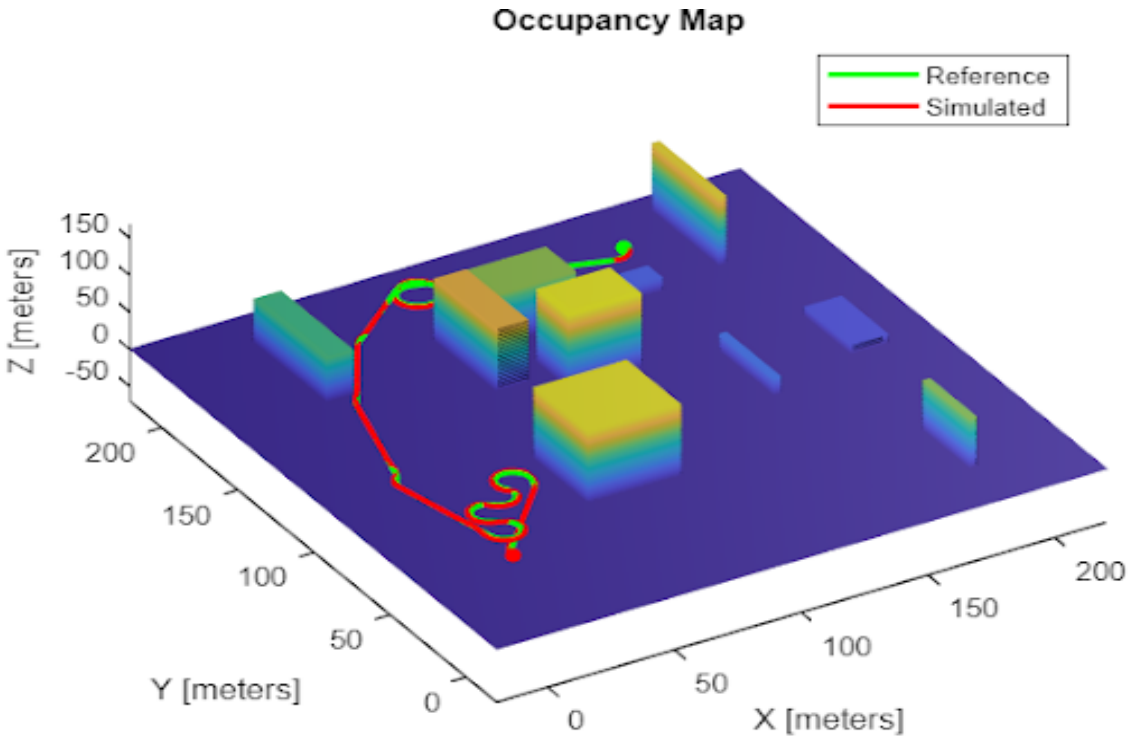



**Fig 2.6**: Implementation of RRT on drone in an 3D environment

**Comparison between A* and RRT algorithm**

1. **Efficiency:** A* tends to be more efficient in low-dimensional spaces with a known grid structure. At the same time, RRT is well-suited for high-dimensional continuous spaces where explicit enumeration of states is impractical.

2. **Optimality:** A* guarantees to find the optimal path if certain conditions are met, while RRT does not provide such guarantees, although it often considers near-optimal paths in practice.

In summary, while A* is a systematic search algorithm suitable for discrete spaces with a known structure, RRT is a probabilistic algorithm tailored for high-dimensional continuous spaces, making it ideal for many robotic motion planning tasks.

Based on the above comparison, the report uses RRT algorithm for simulations since it is more suitable for use in real-world multi-UAV applications.

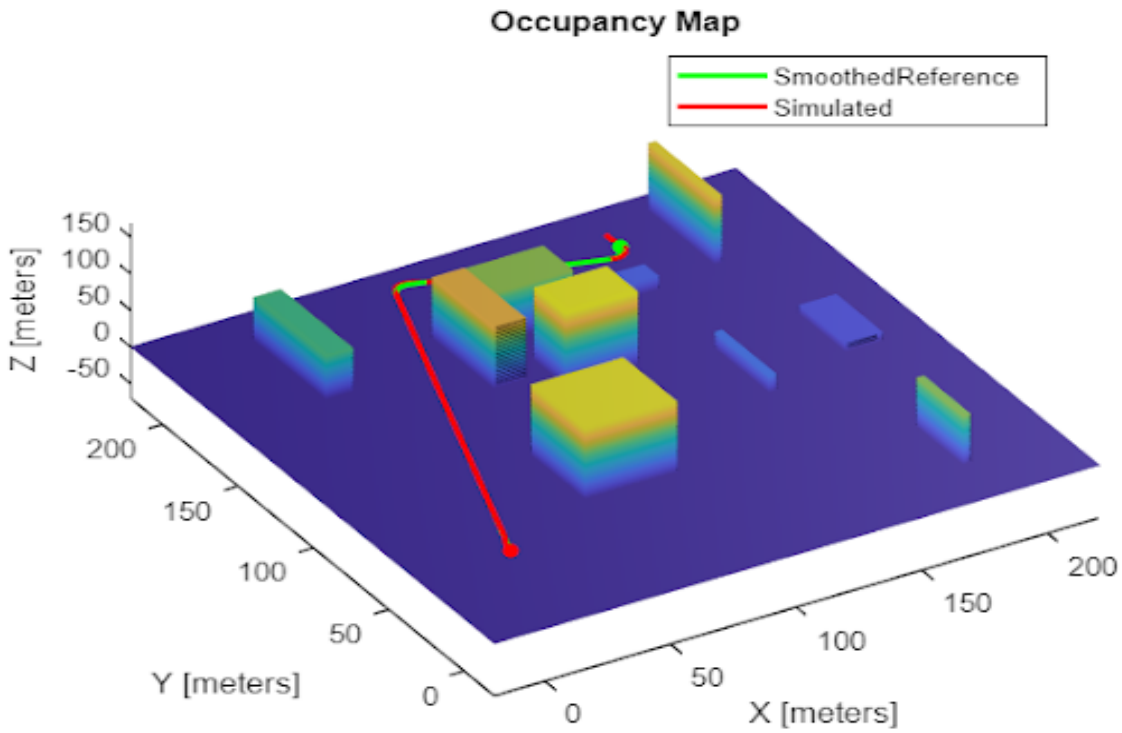



**Fig 2.7**: Smoother path with reduction in maximum node distance and increased iterations

RRT is implemented on a ground robot in a 2D environment. From Fig. 2.4 and Fig.2.5, it can be seen that with a decrease in the maximum inter-node distance and increased iteration, a smoother path can be generated. This algorithm is extended to a 3D map and plans a path for a UAV, considering all its constraints, such as maximum roll angle, maximum forward speed, etc. Fig. 2.6 depicts a path from the starting to the goal position. This path was improvised later by increasing the iterations and decreasing the maximum node distance, as shown in Fig. 2.7

## 2.3 Centralised Tracking

Centralised Tracking essentially means having a central storage point for data of all your system's UAVs. This is important in multi UAV systems since you need to track all your agents to plan their trajectories efficiently and avoid potential collisions. This data may include their initial position, goal position, and current coordinates. This can also be extended to include data like total flight time or the amount of fuel left.

Centralised tracking can be implemented in two ways. One method is depicted in

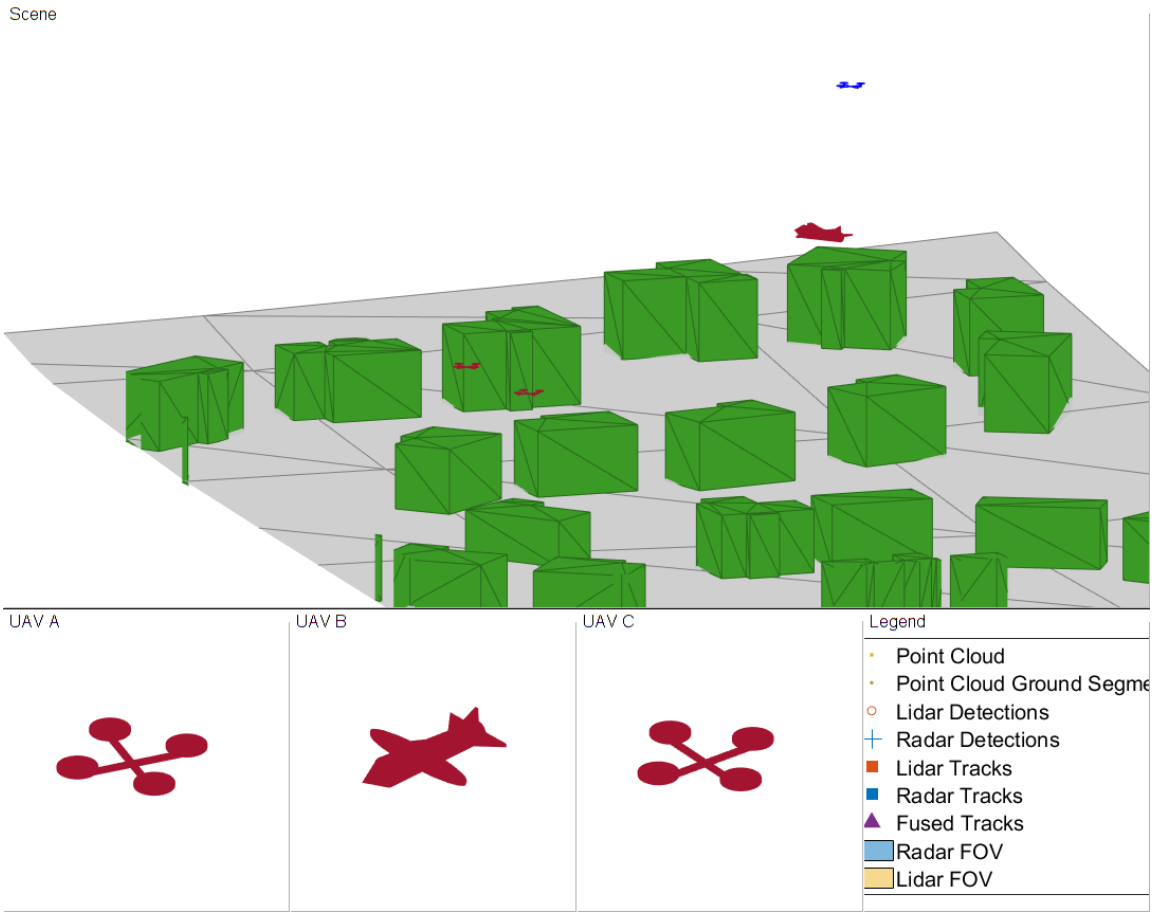



**Fig 2.8**: UAV Designer Scenario for Centralised Tracking

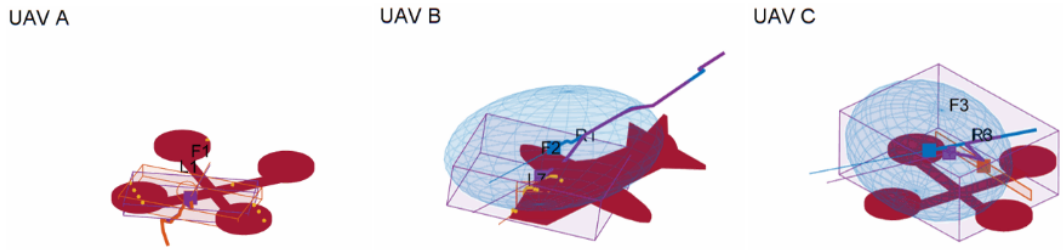



**Fig 2.9**: Three Tracked UAVs

Fig2.8. A central UAV (represented in blue) hovers above the map area and observes the other 3 UAVs (represented in red) in the scenario, described in Fig2.10. In this specific example, it is equipped with LiDAR and RADAR.[5] By fusion of the data from the two sensors, the UAV can obtain the current positions of the 3 UAVs and keep track of them as shown in Fig2.9.

The central UAV mentioned here can be a satellite, keeping track of every UAV in the

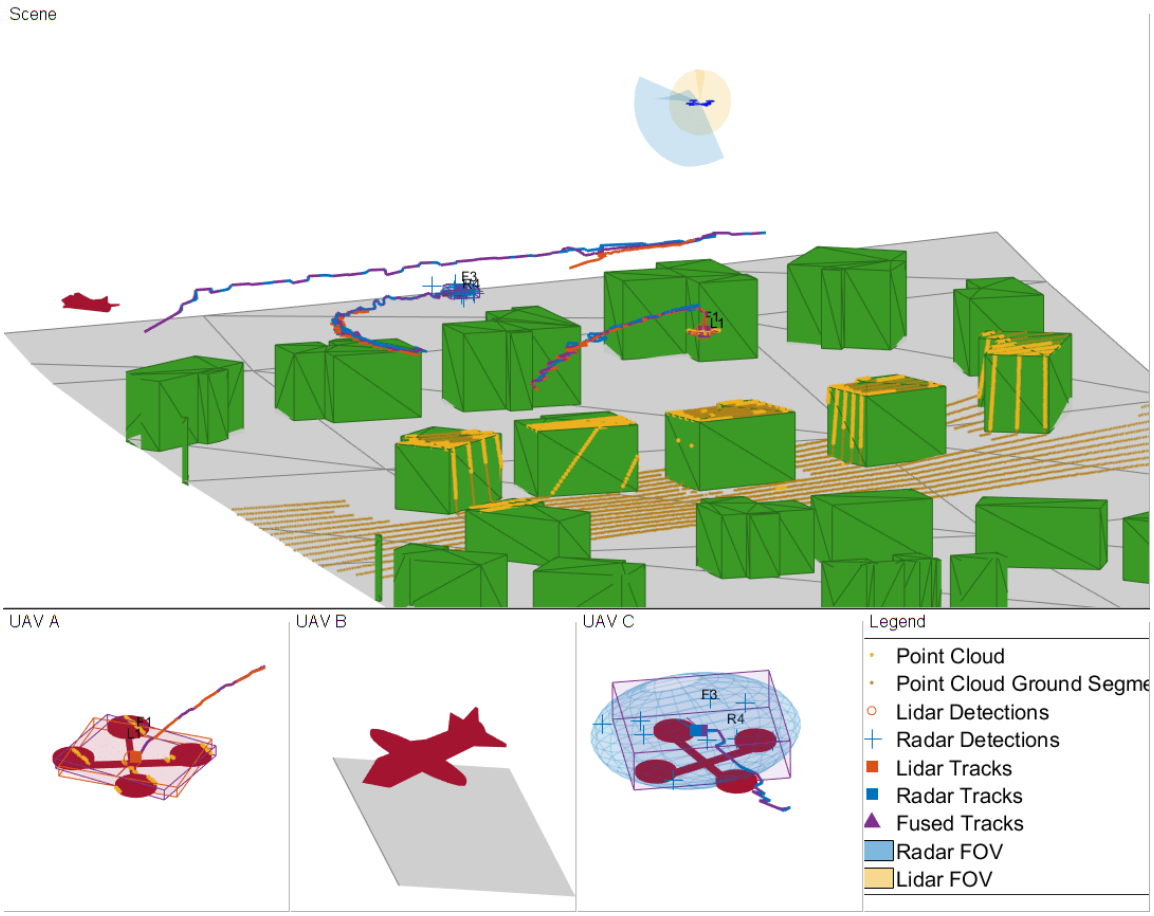



**Fig 2.10**: UAV states being tracked by Central UAV

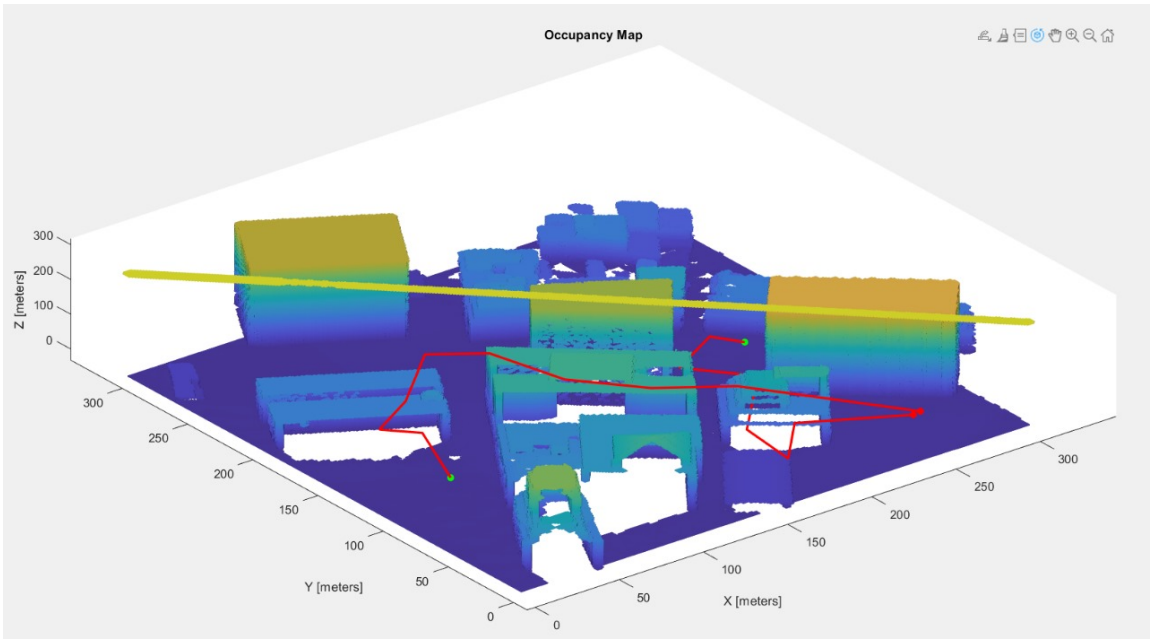



**Fig 2.11**: Generation of 3d occupancy grid from osm file

environment. Since this ego vehicle will be at a certain height, the precision in mapping via LiDAR and RADAR will be reduced. This approach, though, is best applicable in cases where the 3D occupancy grid has not been generated prior.

The second method of implementing centralized tracking is simply using a central data structure of a vector or array. So via simple code the current state of multiple UAVs in the scenario can be updated and kept track of. MATLAB provides a UAV Scenario App that allows us to draw custom maps.[6] Using this map, multiple drones can be introduced in simulation, with an option to modify their initial and goal positions.

## 2.4 Occupancy Grid

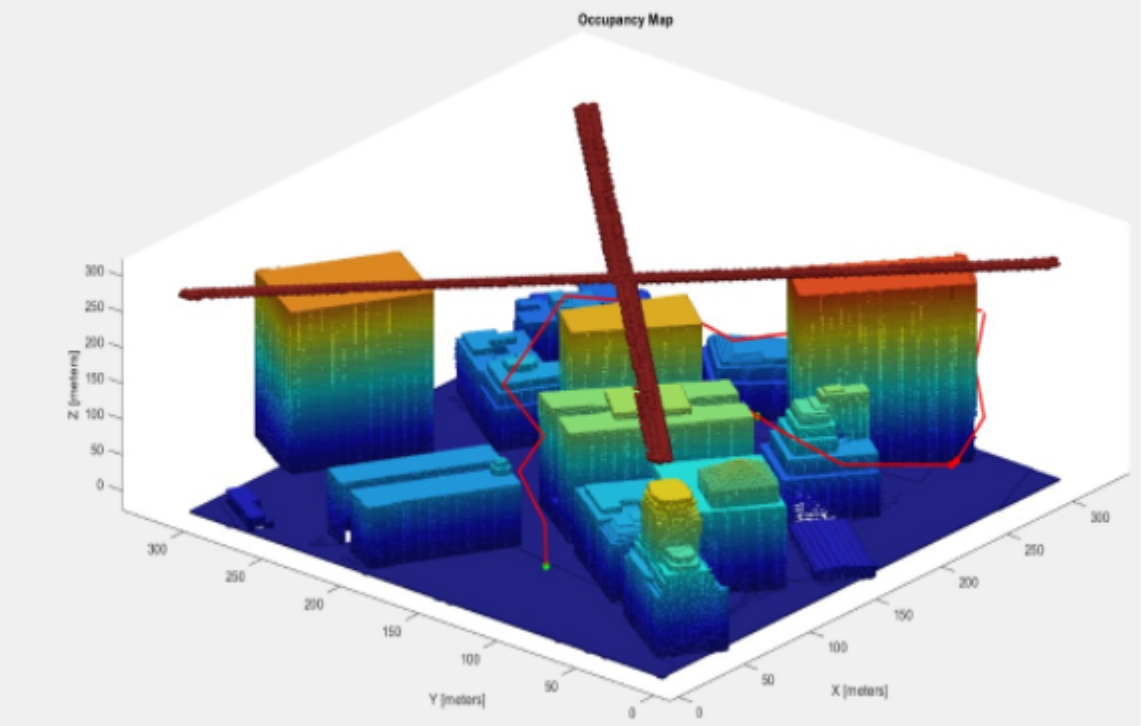



**Fig 2.12**: Improved quality of 3d grid with increased iteration and two drones

According to [7], cccupancy grids are used to represent a robot workspace as a discrete grid. Information about the environment can be collected from sensors in real time or be loaded from prior knowledge. Laser range finders, bump sensors, cameras, and depth sensors are commonly used to find obstacles in the robot's environment.

Occupancy grids are used in robotics algorithms such as path planning. They are

used in mapping applications for integrating sensor information in a discrete map, in path planning for finding collision-free paths, and for localizing robots in a known environment. Maps with different sizes and resolutions to fit the specific application can be created.

For 2-D occupancy grids, there are two representations: **Binary occupancy grid** and **Probability occupancy grid**.

A binary occupancy grid uses true values to represent the occupied workspace (obstacles) and false values to represent the free workspace. This grid shows where obstacles are and whether a robot can move through that space.

A probability occupancy grid uses probability values to create a more detailed map representation. This representation is the preferred method for using occupancy grids. This grid is commonly referred to as simply an occupancy grid. Each cell in the occupancy grid has a value representing the probability of the occupancy of that cell. Values close to 1 represent a high certainty that the cell contains an obstacle. Values close to 0 represent certainty that the cell is not occupied and obstacle free. The probabilistic values can give better fidelity of objects and improve performance of certain algorithm applications.

Binary and probability occupancy grids share several properties and algorithm details. Grid and world coordinates apply to both types of occupancy grids. The inflation function also applies to both grids, but each grid implements it differently. The effects of the log-odds representation and probability saturation apply to probability occupancy grids only.

In order to implement RRT on UAVs, 3D Occupancy Grid Map of the environment is required. Since actual maps of the city are used to plan the paths for the UAVs, it was not possible to directly convert the Open Street Map (OSM) file into a 3D Occupancy Grid. Once the grid is generated, RRT could be quickly implemented.

To tackle this problem, drones equipped with LiDAR are moved around the city to generate the 3D Occupancy Grid.[8] In the depicted Fig 2.11, a drone generating the 3D Occupancy Grid can be seen, albeit the grid is of low quality. To enhance the map quality, two drones are used with an increased iteration time. The resultant improved map is shown in Fig2.12.

# Chapter 3

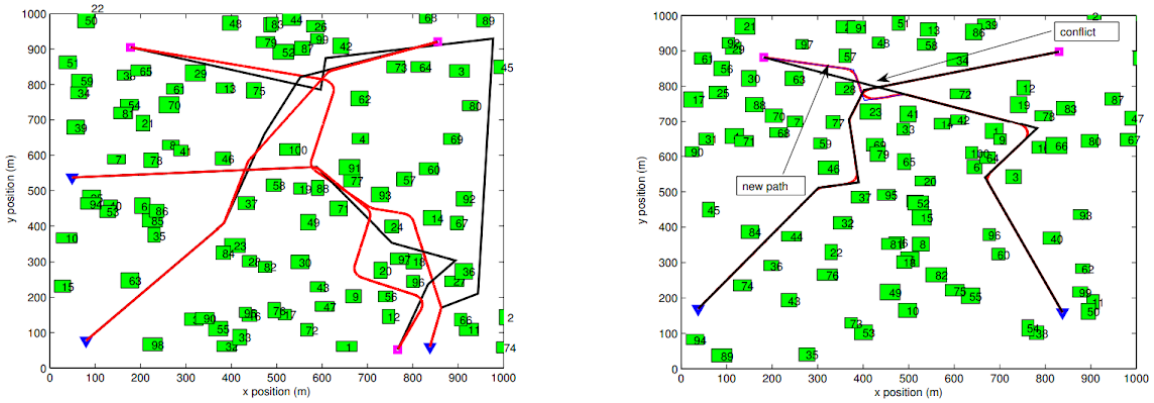
# Related Work

1. Conflict-free Four-dimensional Path Planning for Urban Air Mobility considering Airspace Occupancy [9]: The paper proposes a path planning model for conflict-free urban air mobility operation. The model includes a point mass model for aircraft speed estimation and higher priority flight-plans are considered as dynamic obstacles. Mathematical representations for the avoidance of both static and dynamic obstacles are presented. The main algorithm - Conflict-Free A* (CFA*) was developed for 4D path planning based on First-come-First-served (FCFS) scheme. This algorithm includes a novel design of heuristic function as well as a decision-making process for collision avoidance. The algorithm was analysed by numerical experiment carried out in an urban airspace in Singapore. Results show that the algorithm successfully resolves a large number of potential conflicts with acceptable flight-time cost and computational time. Fig.3.1 shows the trajectories of A* and CFA* in different scenarios.

2. Probabilistically Guaranteed Path Planning for Safe Urban Air Mobility Using Chance Constrained RRT* [10]: The algorithm presented is considered as an extension of standard RRT*. It improves the way of parent node selection, and introduces the notion of cost function to select the node as the parent node which has the minimum cost within the neighbourhood of the node extended. At the same time, the nodes in the existing tree will be reconnected after each iteration, so as to ensure the computational complexity and asymptotic optimality. It applies trajectory wise constraints checking, allowing the incorporation of probabilistic constraints. Sev-


(a) (b)

**Fig 3.1**: Trajectories of A* and CFA* in different scenarios




(a) (b)

**Fig 3.2**: RRT trajectories with Static and Dynamic obstacles

eral simulations on cases like single static uncertain obstacle and multiple static uncertain obstacles are performed to verify the feasibility of the algorithm using a single UAV. Simulation results show that a path for the vehicle satisfying probabilistic feasibility can be identified by the proposed algorithm. Fig.3.3 depicts a simple diagram of CC-RRT* planning.

3. Strategic and Tactical Path Planning for Urban Air Mobility - Overview and Application to Real World Use Cases [11]: A planning framework is presented, which encompasses two phases - (i) strategic phase, which is aimed at evaluating an optimised trajectory for the UAV before the flight, and (ii) tactical phase, which continuously checks the trajectory during the flight and takes action in case an unpre-

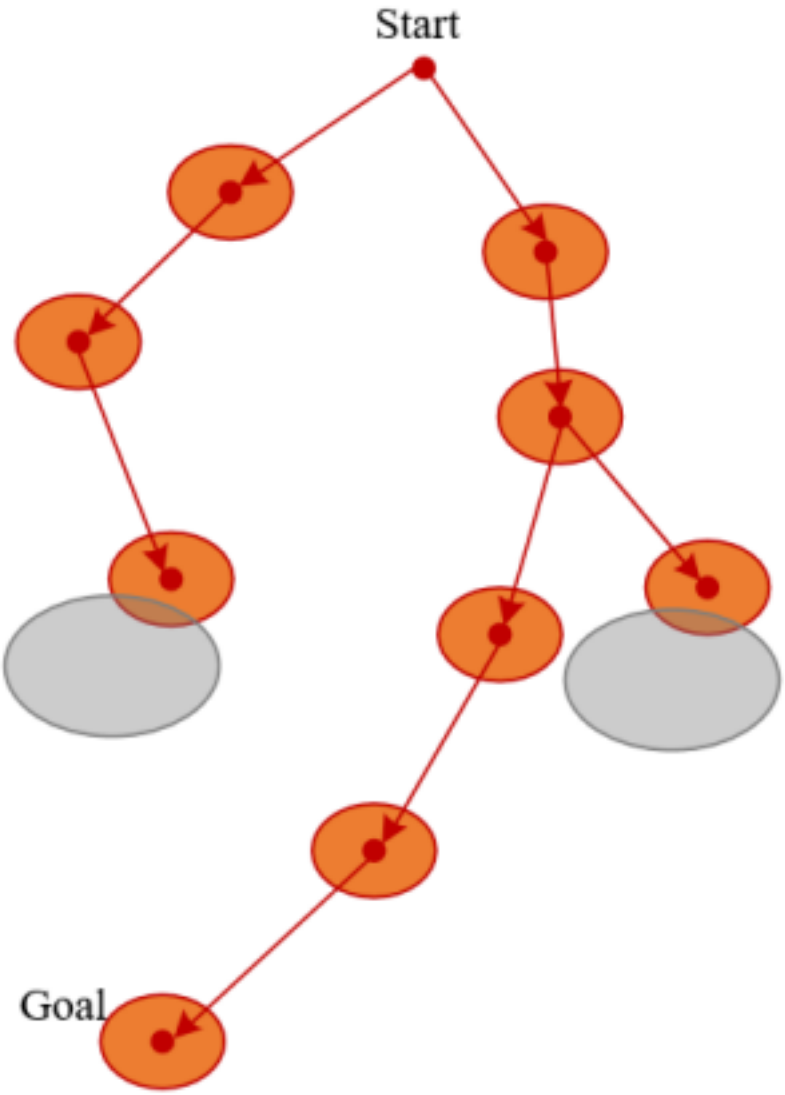



**Fig 3.3**: Diagram of CC-RRT* planning

dicted event occurs that compromises the trajectory safety and effectiveness. The need for decomposing the planning approach is justified for guaranteeing the path optimality while reducing the computational cost during flight, as tactical replanning is only demanded at finding deviations from nominal trajectory. To ensure the reduction of computational cost, a strategic path planner must be carried out with the largest amount of available information, following the better informed, better-planned logic. The algorithm is tested on two scenarios that involve an air taxi in a dense urban environment, and medical delivery from mainland to island. The results demonstrate the optimised and time-saving trajectories yielded by the algorithm.

4. Multi-UAV Path Planning in Obstacle Rich Environments Using Rapidly-exploring Random Trees [12]: The paper details techniques using RRTs to generate paths for multiple unmanned aerial vehicles in real time, from given starting locations to goal locations in the presence of static, pop-up and dynamic obstacles. To accommodate challenges like short-time window and turn radius constraints, a path is first quickly generated using RRT by considering the kinematic constraints. Consequently, in order to generate a low cost path, an anytime algorithm is developed that yields a path whose quality improves as flight proceeds. A dynamic obstacle is avoided based on a set criteria by the path planner. The generated paths are tracked with a guidance law based on pursuit and line of sight. Simulation studies are carried out in an obstacle rich environment containing static, dynamic and pop up obstacles. Trajectories with these static and dynamic obstacles are shown in Fig.3.2.

# Chapter 4

# Proposed Algorithms

An instance of the problem consists of $m$ agents in a defined environment. The environment consists of static obstacles represented using occupancy grids analogous to residential or commercial buildings found in an urban scenario. Let $P_n(t)$ belong to $P_n$ be the location of agent n at timestamp $t$, where $P_n$ is a path of nodes from initial to goal position that agent $n$ can move along. At each time stamp $t$, it has to be ensured that no two agents are in the exact location. To avoid collisions, agents either wait for a particular location to be empty before passing through it or replan their path around the occupied location to avoid collisions with other agents. Each agent is also assigned a priority number depending on the importance of the task assigned. This priority number will be useful in deciding the agent which will be allowed to move first, in case of a path conflict between the agents. Based on this setup two algorithms are proposed, namely, Two Step Interplanner and Two Step Interplanner with Conflict Resolution.

## 4.1 Two Step Interplanner

In this algorithm, initial path planning is done using the rapidly exploring random trees (RRT) algorithm. To elaborate, if an agent is given a start position and goal position based on the assigned task, the optimal path considering static obstacles between start and goal position will be calculated using Global RRT Planner as depicted in Fig.4.1 . This generated path is stored in an array. However, this static planning is not enough to account for other agents in the environment which will behave like dynamic obstacles.

Therefore, the RRT algorithm is used twice in the Two Step Interplanner approach. For simplicity, consider two agents, Agent A and Agent B. Let A1, A2, A3 and B1, B2, B3 be the waypoints of an agent as calculated by the Global Planner.

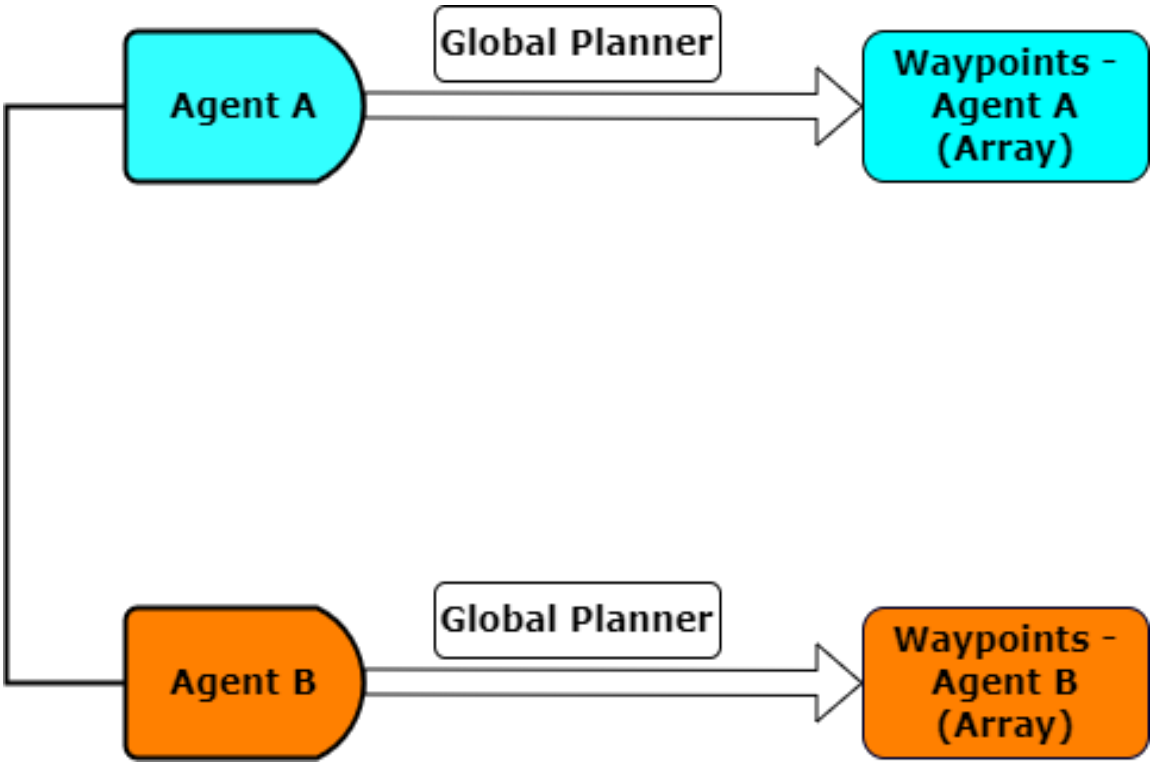



**Fig 4.1**: Interplanner Step 1

In the second step, the RRT algorithm is used again but to plan between the nodes for an agent. At t = 0 if agent A is at start position, the algorithm assumes all other agents at rest and as static obstacles for that particular instance. Then the Turn by Turn Planner plans the path from start position to node A1, with a smaller connection distance as compared to Global Planner. When agent A has reached node A1, agent B plans its path from its start position to B1, considering the position of A1 as an obstacle. In this manner, turn by turn both agents A and B reach their goal positions depicted in the flowchart of Fig.4.2. After a series of simulations, it was evident that due to the number of redundant computations, as the number of agents increases, the computation time will increase significantly.

Simulation was performed on a 2D occupancy map of a warehouse in MATLAB.[13] The algorithm was tested on two agents by randomly allocating pickup and goal locations. Fig.4.3 visualises the starting and goal positions of two agents in the occupancy grid. The

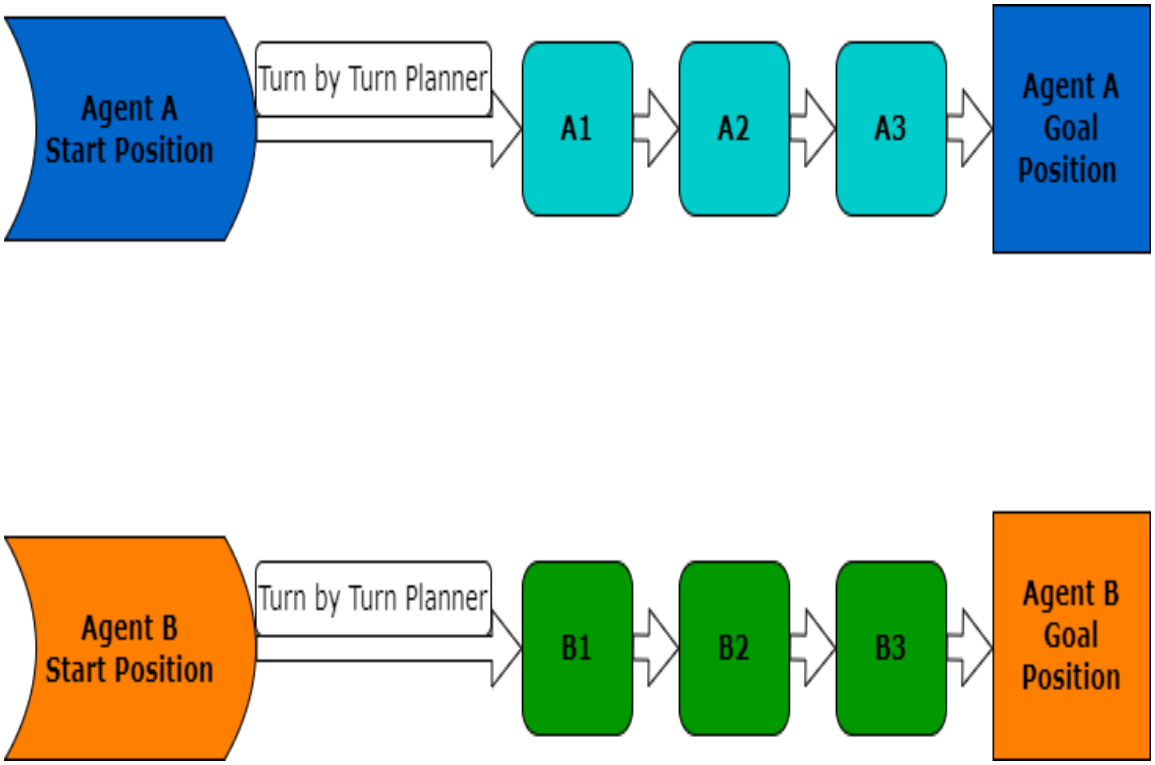



**Fig 4.2**: Interplanner Step 2

heading angles of every agent can be varied, as per user input. The state space of the vehicle was specified using a stateSpaceDubins object in MATLAB. State bounds were specified as parameters of this object. The object limits the sampled states to feasible Dubins curves for steering a vehicle within the state bounds. A small turning radius allows for tight turns in this small environment. The RRT algorithm samples random states within the state space and attempts to connect a path. These states and connections need to be validated or excluded based on the map constraints. The vehicle must not collide with obstacles defined in the map. The validation distance discretises the path connections and checks obstacles in the map. The algorithm was tested for various sets of maximum connection distances between consecutive nodes and number of iterations. It was observed that decreasing the maximum connection distance and increasing the iterations generated a smoother trajectory. Fig.4.4 shows the planned trajectories for two agents using Two Step Interplanner. Table 4.1 shows the values of the relevant parameters of simulation.

After validating the results for 2D simulation, a 3D simulation using UAV Agents A and B is performed.[14] The results are presented in Fig 4.5 based on the parameters

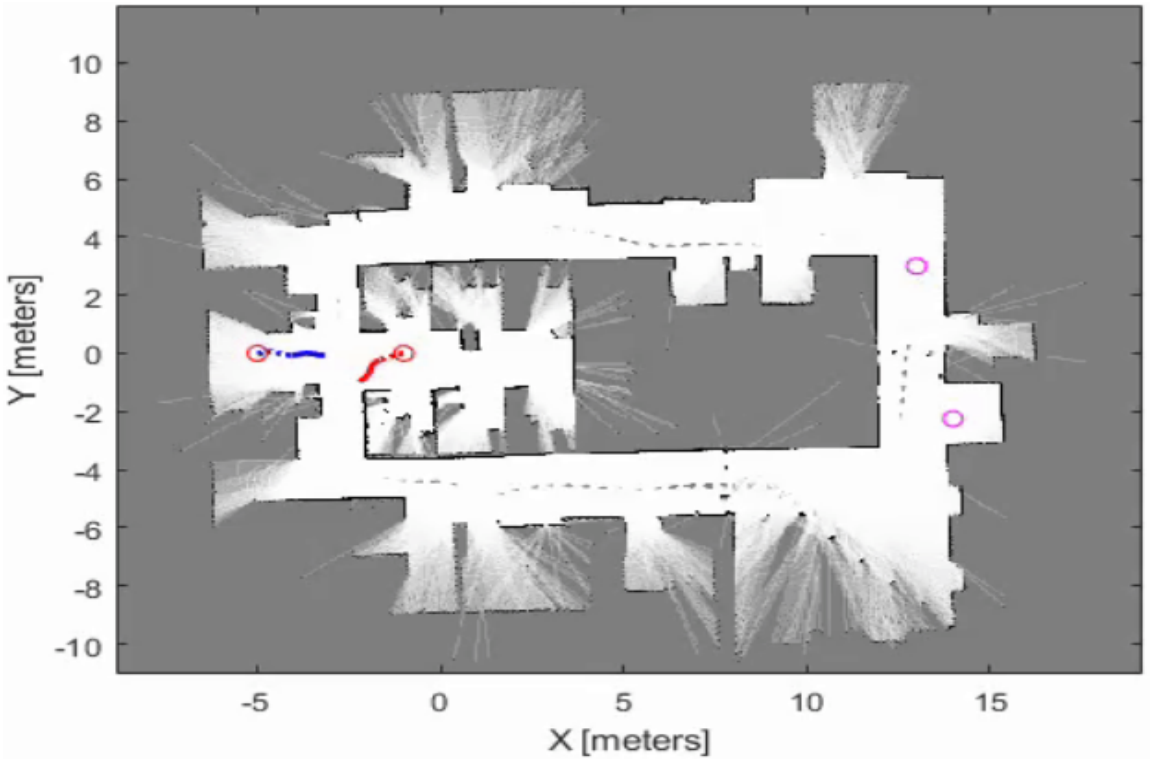



**Fig 4.3**: Start and Goal Positions of Agent A and Agent B

| Parameters | Values |
|---|---|
| Minimum turning radius | 0.4m |
| Validation distance | 0.05m |
| Maximum connection distance | 1m |
| Maximum iterations | 30000 |

Table 4.1: Parameters for 2D simulation

indicated in Table 4.2

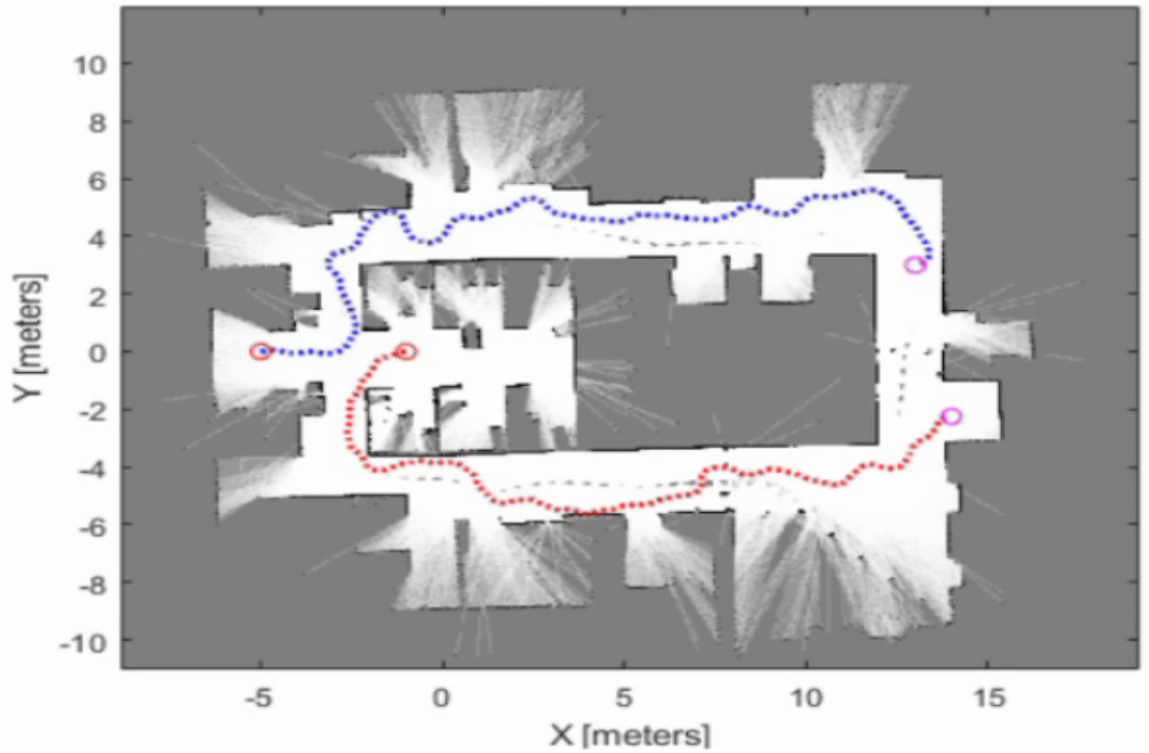



**Fig 4.4**: Trajectories plotted for Agent A and Agent B

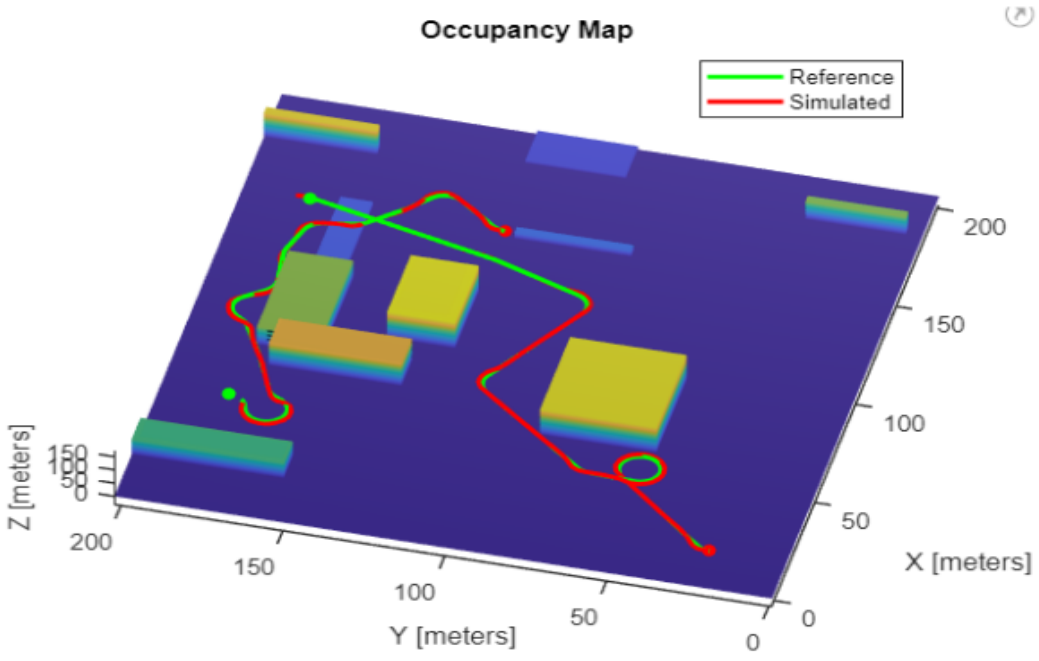



**Fig 4.5**: Trajectories plotted for UAV A and UAV B

| Parameters | Values |
|---|---|
| Maximum roll Angle | pi/6 |
| Airspeed | 6 m/s |
| Bounds | [-20 220; -20 220; 10 100; -pi pi]) |
| threshold | [(goal-0.5)' (goal+0.5)'; -pi pi] |
| Flight Path Angle Limit | [-0.1 0.1] in radians |
| Maximum Connection Distance | 10 m |
| Maximum Iterations | 4000 |

Table 4.2: Parameters for 3D simulation

## 4.2 Two-Step Interplanner with Conflict Resolution

To account for the previous shortcomings, the Two Step Interplanner was improved by introducing certain constraints. These constraints decreased the number of redundant computations, reducing the computation time.

To elaborate, consider 3 agents Agent A, Agent B and Agent C. Similar to the previous algorithm, path planning for each agent from start to goal position is done using the Global Planner. However, the usage of the Turn by Turn Planner is carefully decided. Instead of replanning for every iteration, the paths are replanned only if the proximity of two agents comes under a certain threshold. The edge case of one agent reaching the goal position before the other agent is also handled, as the agent which reaches its goal position is treated as a static obstacle for any other moving agents. Fig4.6 shows the flowchart of Two Step Interplanner with Conflict Resolution.

Along the lines of simulation of Two Step Interplanner, the simulation of Two Step Interplanner with Conflict Resolution is performed in a 2D environment. The parameters are described in Table 4.3 and Fig4.7 depicts the trajectories generated.

| Parameters | Values |
|---|---|
| Minimum turning radius | 0.4m |
| Validation distance | 0.05m |
| Maximum connection distance | 1m |
| Maximum iterations | 30000 |

Table 4.3: Parameters for 2D simulation with Conflict Resolution

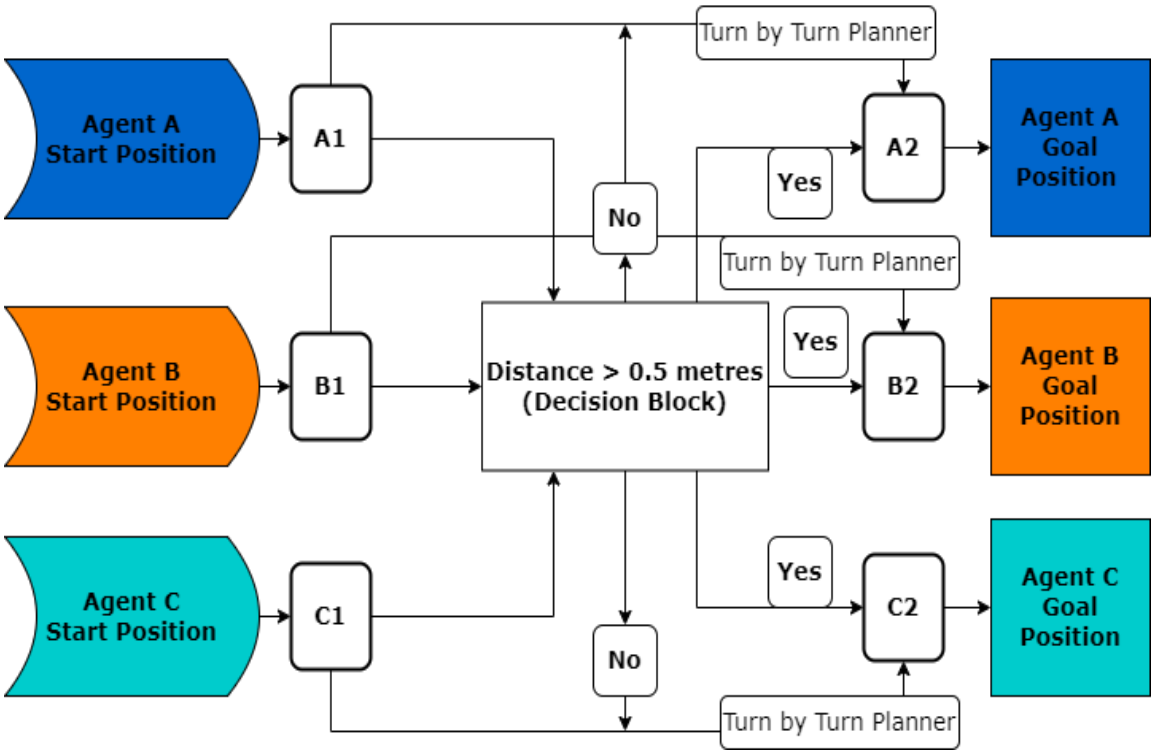



**Fig 4.6**: Two Step Interplanner with Conflict Resolution

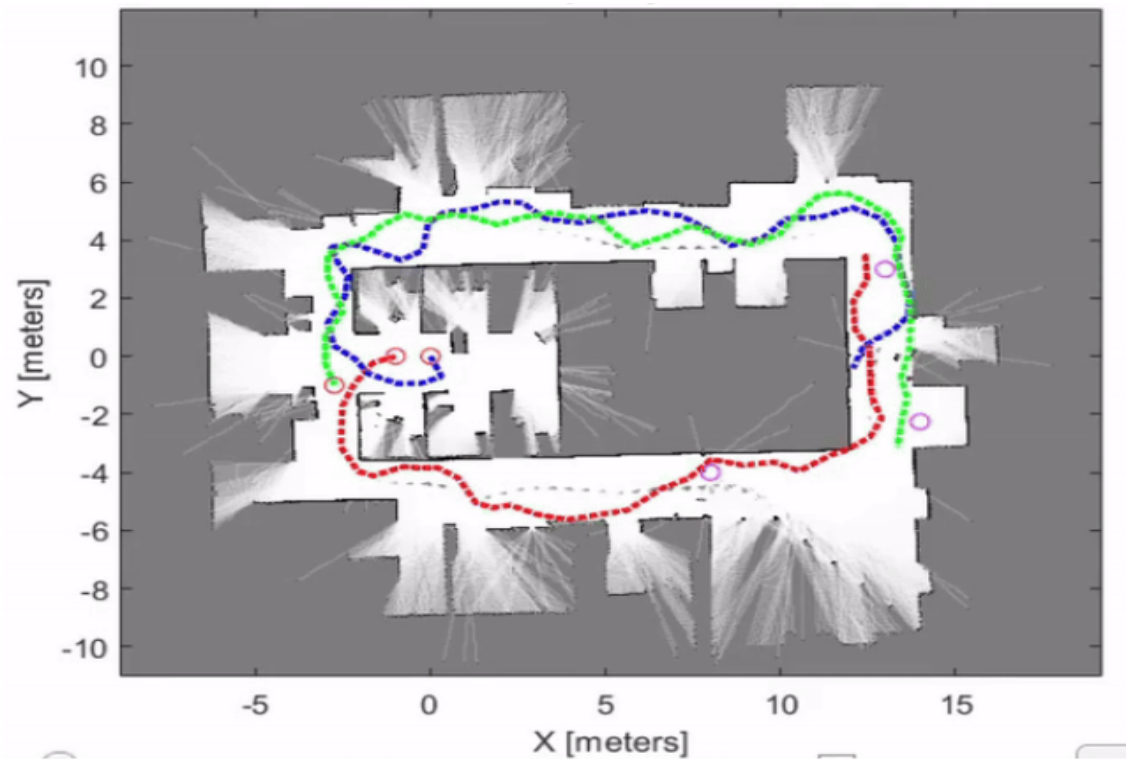



**Fig 4.7**: Trajectories plotted for Agent A, Agent B and Agent C

# Chapter 5

# Dynamic Task Reallocation: Case Study

## 5.1 Introduction

In the realm of unmanned aerial vehicles (UAVs), task allocation stands as a pivotal challenge, crucial for the seamless execution of missions ranging from surveillance and reconnaissance to disaster response and beyond. However, the application of drones in real-world scenarios faces a diverse array of limitations, particularly in the realm of task assignment. Traditional algorithms for task allocation operate under the assumption of a static environment, a paradigm that starkly contrasts with the dynamic and unpredictable nature of real-world operations.

UAVs, often deployed in dynamic and volatile environments, are ill-equipped to handle unforeseen events that may arise during mission execution. Conventional task allocation frameworks suffer from rigidity, lacking the agility necessary to adapt to changing circumstances. Tasks are typically predetermined and known at the outset of a mission, leaving little room for flexibility to accommodate emergent needs or unexpected developments. Furthermore, communication between UAVs and centralized stations may be unreliable, further exacerbating the challenges associated with task allocation.

At the heart of these challenges lies the task allocation problem: the imperative task of assigning a set of tasks to UAVs while ensuring adherence to constraints and minimizing conflicts. While existing studies have made strides in addressing this issue, they primarily focus on static versions of task allocation, overlooking the dynamic nature of real-world environments. The inability to effectively cope with dynamic events, such as UAV failures

or sudden environmental changes, can lead to mission failure or suboptimal outcomes.

In light of these limitations, there is a pressing need to reevaluate traditional approaches to task allocation and develop methodologies that are robust, adaptive, and capable of responding to the dynamic nature of real-world operations. By addressing the shortcomings of existing frameworks and embracing the challenges posed by dynamic environments, the way for more resilient and effective UAV missions can be paved.

## 5.2 Related Work

There have been various notable works addressing the multi-UAV task allocation problem, typically categorized into two main approaches: optimization-based and market-based [15].

### 5.2.1 Optimization-based Approach

1. These methods aim to find the optimal solution for task allocation in multi-UAV systems [16].

2. Optimization-based techniques exhibit excellent exploration performance, but they are predominantly utilized in centralized systems[17].

#### Limitations of Centralized Approaches

1. Designing appropriate local decision rules for dynamic environments poses a significant challenge.

2. Centralized approaches incur heavy communication burdens, limiting mission coverage area.

3. High computational demands are imposed as these approaches track every UAV's progress throughout the mission.

4. Centralized systems are susceptible to single points of failure, compromising mission reliability.

5. Mathematical modeling of dynamic task allocation is complex due to the presence of multiple uncertain constraints in real-world scenarios.

### 5.2.2 Market-based Approach

1. Market-based techniques draw inspiration from economic theory, offering an effective means to coordinate UAVs.

2. Auction algorithms are a prevalent example of market-based approaches, commonly employed in large-scale task allocation problems

Some of the approaches that researchers have implemented with varying degrees of success, are briefly discussed here:

1. Auctions for multi-robot task allocation in communication limited environments [18]: In the proposed approach, a UAV serves as the auctioneer, orchestrating task bids from participating UAVs, with the highest bidder being awarded the assignment(Fig. 5.1). Furthermore, the approach can be adapted to decentralized architectures, though it may introduce additional complexity. It delves into the challenge of multi-robot task allocation in scenarios with limited communication. It investigates the impact of imperfect communication on solution quality within auction-based approaches. Through analytical and experimental exploration, the study reveals that different auction algorithms degrade uniquely as communication quality diminishes. Consequently, it emphasizes the critical importance of carefully selecting auction algorithms for systems operating in environments with lossy communication.

2. Consensus-based decentralized auctions for robust task allocation [19]: This paper considers the task allocation problems in a distributed multi-robot system under critical time constraints. It proposed a Consensus-based bundle algorithm (CBBA). UAVS bids on bundles rather than individual tasks in CBBA. The algorithm runs on each UAV individually without an auctioneer. A dynamic grouping allocation method is proposed. It builds upon the state-of-the-art consensus-based auction algorithms, extending them in both task inclusion phase and consensus phase

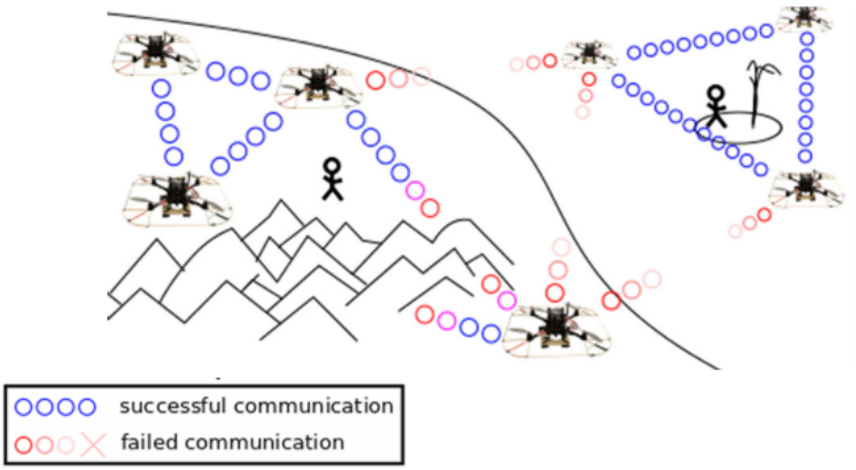



**Fig 5.1**: Auctions in communication limited environments

3. A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenarios [20]: It proposed a distributed heuristic algorithm known as the Performance impact (PI) algorithm. which aims to minimize the overall waiting time. PI extracts some specific search rules based on the properties(as depicted in Fig.5.2) of the task allocation problem to obtain optimal or suboptimal solutions rapidly. The proposed method is simple and effective. It directly aims at optimizing the mathematical objective defined for the problem. A new concept of significance is defined for every task and is measured by the contribution to the local cost generated by a vehicle, which underlies the key idea of the algorithm. The whole algorithm iterates between a task inclusion phase, and a consensus and task removal phase, running concurrently on all the vehicles where local communication exists between them

These approaches heavily rely on much more computation power and reliable communication between a UAV and its neighbour. Since, every UAV must communicate with others to make decisions in distributed system, the communication network may have great impact on the final assignment solution. Additionally, these approaches only considers the static version of task allocation and are ineffective for dynamic changes in environment.

Dynamics events are classified under two categories:

Discrete dynamics include newly adding and deleting tasks, whereas continuous dy-

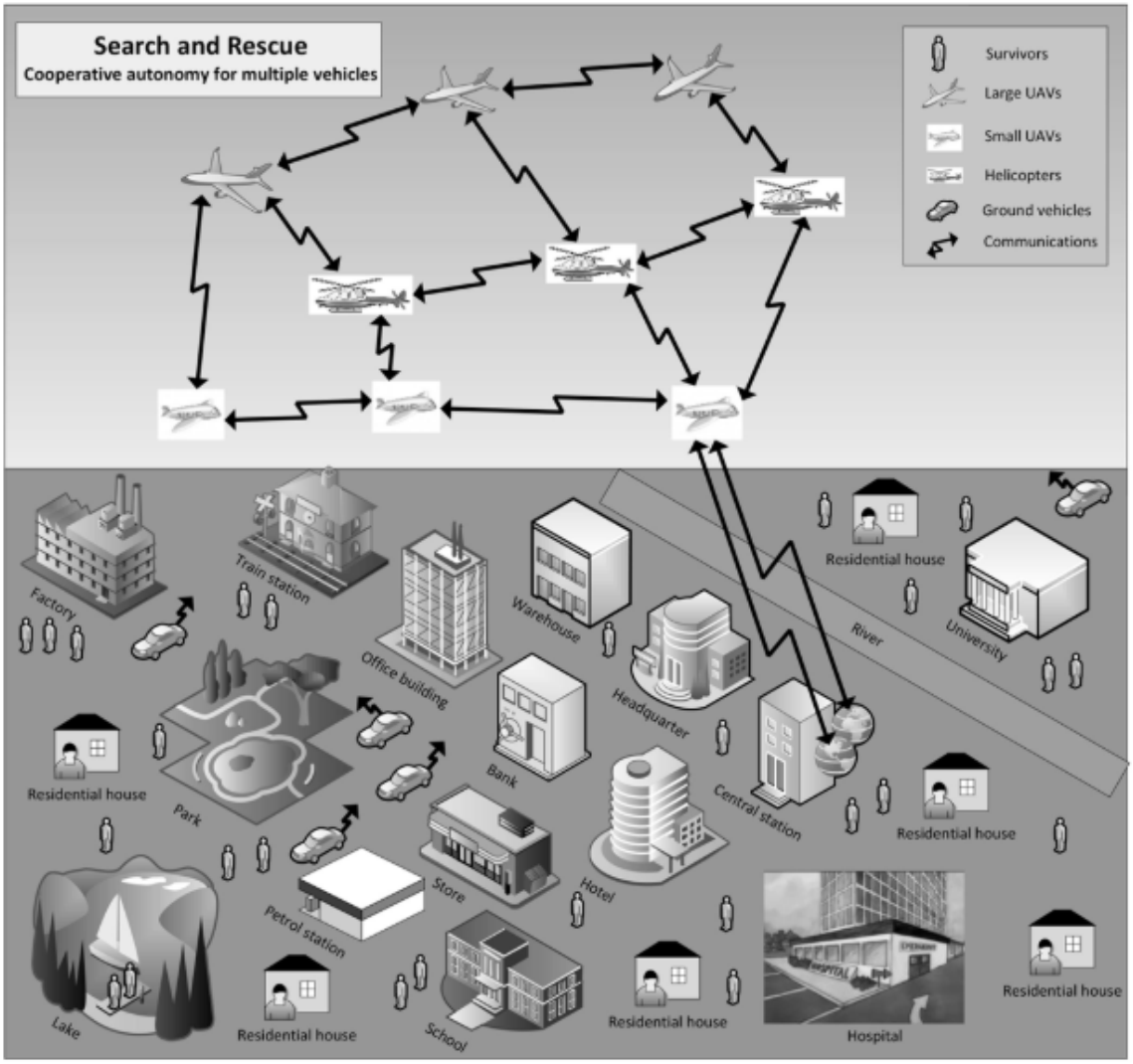



**Fig 5.2**: Heterogeneous unmanned vehicles, aerial and ground, collaborate in a search and rescue mission, gathering survivor data and coordinating for efficient rescue operations

namics include updating the task location, task deadline, and task duration. The emergence of new tasks and the disappearance of tasks are one of the most common dynamic events. Fig. 5.6 explains the influence of the other three types of continuously changed tasks on the original task allocation solution which is shown in Fig. 5.6(a). If the travel time from t1 to t2 is extended by a green slash rectangle due to a changed position, t2 cannot be started before its deadline as shown in Fig. 5.6 (b). Likewise, the advanced deadline denoted by a green dotted line in Fig. 5.6 (c) and extended execution duration represented by a green rectangle in Fig. 5.6 (d) may also result in that t2 cannot be started before its deadline.

To illustrate this concept, consider a scenario where drones inspect suspension in-

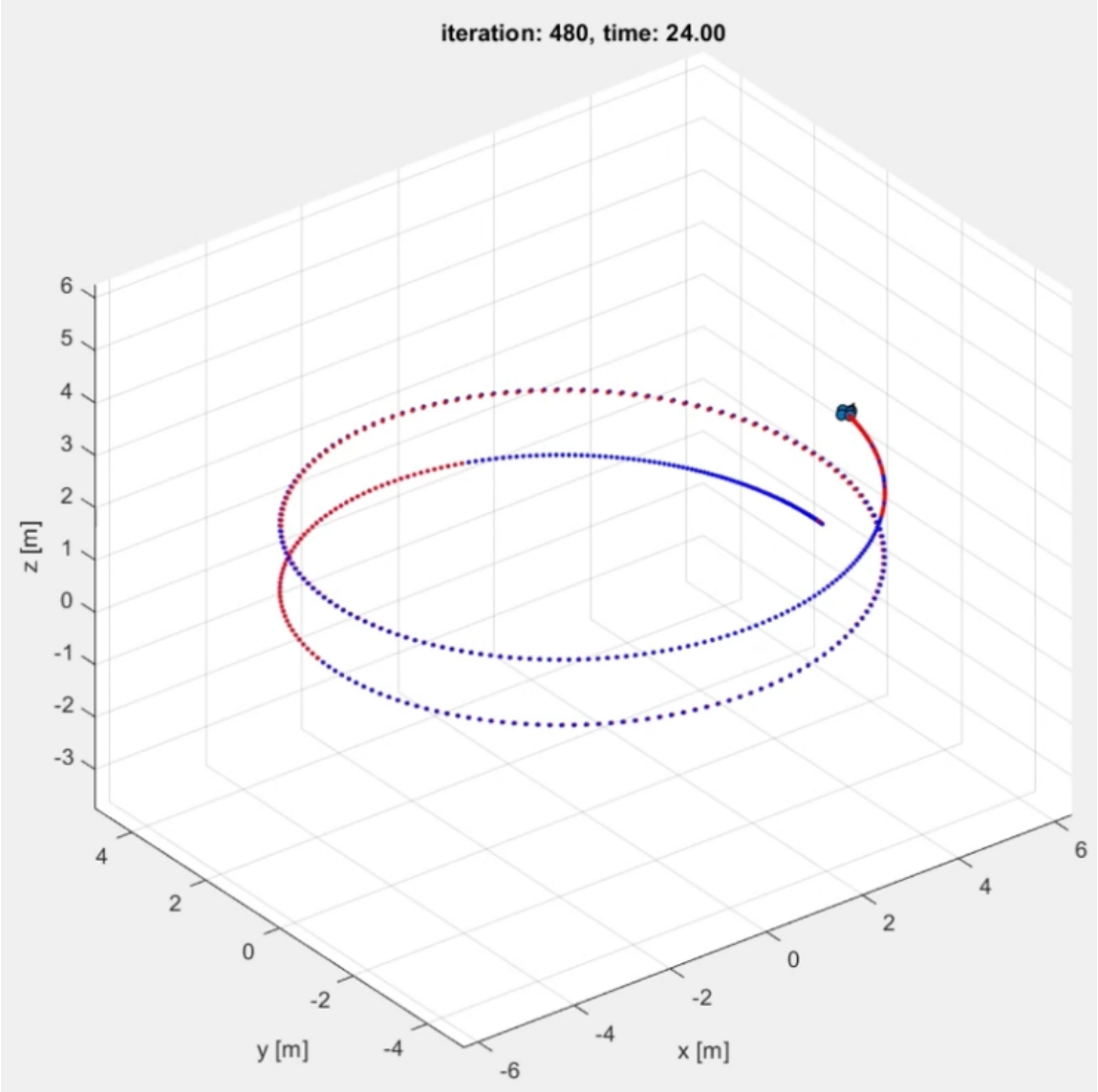



**Fig 5.3**: UAV following a helical trajectory with generated waypoints

sulators on transmission towers using a helical trajectory while maintaining a constant focus on the insulator. This scenario is simulated in MATLAB, where waypoints for the UAV to follow a helical trajectory are generated, as seen in the Fig.5.3.

The aerial vehicle is also simulated in Mission Planner to better visualize the Region of Interest(ROI). The red and orange line in Fig.5.4 and Fig.5.5 depicts the heading of the drone which constantly points in the center ( in our case, the Suspension insulator).

Now, for instance, the task location may shift to a different insulator on the tower, the deadline for completion may be revised, or the duration of circling the insulator may be extended for better data collection. This example vividly demonstrates the need for dy-

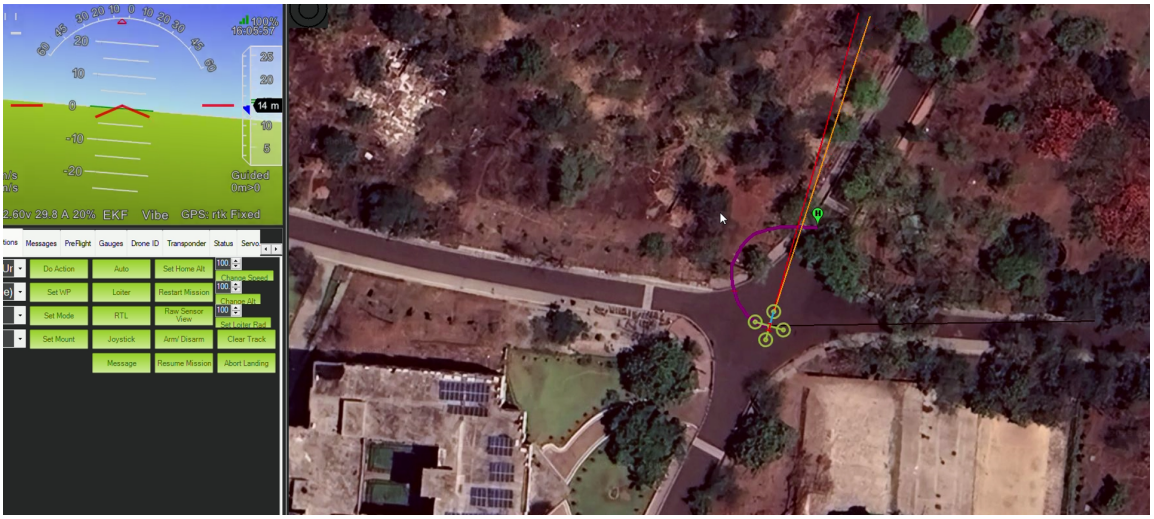


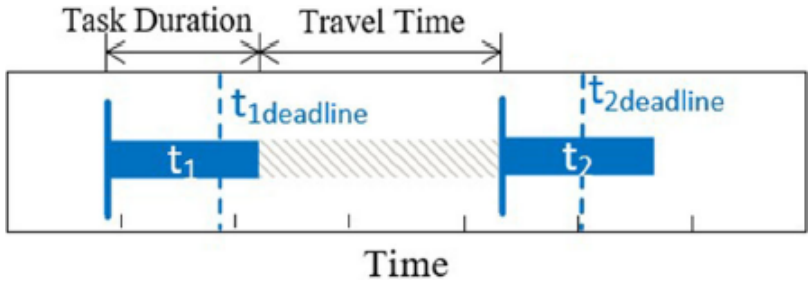

**Fig 5.4**: Simulation of drone in Mission Planner with ROI(Region of Interest) at center


**Fig 5.5**: Completion of one circle with gradually decreasing height to maintain a helical trajectory

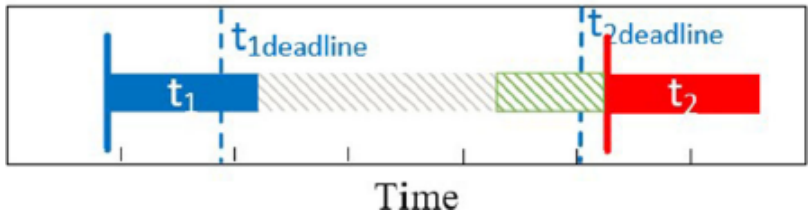
namic task reallocation and its potential impact on optimizing UAV missions in dynamic environments.

**A distributed task reassignment method in a dynamic environment for the multi-UAV system: [21]** proposes a distributed method to cope with dynamic events that occur online during the execution of original schedules. First, a distributed framework for determining the processing strategy according to the types of dynamic events is introduced. Second, a partial reassignment algorithm (PRA) is proposed to support the framework and an incremental subteam formation mechanism and a partial releasing mechanism are developed to release the computation and communication burden. Fur-

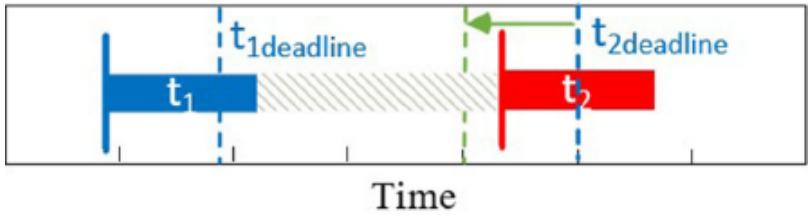
thermore, a modified inclusion phase to maximize assignment (MIP-MA) is also proposed in PRA to maximize the number of task allocations.



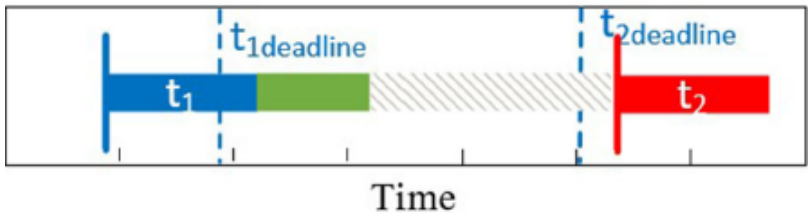

(a) Initial Schedule




(b) Changed Position




(c) Advanced Deadline




(d) Extended Execution Duration

**Fig 5.6**: Schematic representation of the influence of three continuous dynamics on the original task allocation solution: changed position, advanced deadline and extended execution duration

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

From the previous chapters, following conclusions can be drawn:

1. The Two Step Interplanner exhibits a clear advantage over existing works by actively considering potential points of collision and vehicle constraints during path planning.

2. Simulation conducted in 2D and 3D environments, encompassing ground and aerial vehicles, demonstrates the algorithm's robustness.

3. The Two Step Interplanner with Conflict Resolution addresses the major limitations of redundant computations and increased computation time observed in the initial approach.

4. A comprehensive case study on Dynamic Task Reallocation actively provides insights into managing tasks in real-time dynamic environments, offering system flexibility to adapt mid-mission.

5. The introduction of a Partial Releasing Mechanism alleviates computation and communication burdens from UAVs, enhancing their capability to handle dynamic tasks.

## 6.2 Future Work

In this work, a novel algorithm is proposed to improve the path-planning performance of multiple UAVs in an urban city environment. Limitations of centralised tracking in the approaches includes:

1. It tracks every UAV, and as the mission progresses, the communication burden increases, thereby reducing the mission coverage area
2. Centralised approaches are vulnerable to a single point of failure
3. Mathematical modeling of dynamic task allocation is formidable as the real world consists of multiple uncertain constraints.

There is a vast scope of research in this field, primarily in decentralised and market-based methods. Market-based techniques are inspired by economic theory and provide an effective way to coordinate UAVs. The dynamic nature of the environment suggests that UAVs will have to be equipped with robust control and planning algorithms in the future. The detailed case study on task reallocation describes the importance of having flexibility for dynamic tasks according to the requirement in real time.

# Bibliography


[1] Wikipedia contributors. *Urban air mobility — Wikipedia, The Free Encyclopedia*. [Online; accessed 24-April-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Urban_air_mobility&oldid=1209006806.

[2] Peter Hart, Nils Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/tssc.1968.300136. URL: https://doi.org/10.1109/tssc.1968.300136.

[3] Steven M. LaValle. "Rapidly-exploring random trees : a new tool for path planning". In: *The annual research report* (1998). URL: https://api.semanticscholar.org/CorpusID:14744621.

[4] Brian Douglas. *Autonomous Navigation*. MATLAB YouTube. 2020. URL: https://www.youtube.com/playlist?list=PLn8PRpmsu08rLRGrnF-S6TyGrmcA2X7kg.

[5] MATLAB. *LiDAR and RADAR Fusion using in Urban Air Mobility Scenario*. 2024. URL: https://in.mathworks.com/help/uav/ug/lidar-and-radar-fusion-in-an-urban-air-mobility-scenario.html.

[6] MATLAB. *UAV Scenario Designer App*. 2024. URL: https://in.mathworks.com/help/uav/ref/uavscenariodesigner-app.html.

[7] MATLAB. *Occupancy Grids*. [Online; accessed 24-April-2024]. 2024. URL: https://in.mathworks.com/help/robotics/ug/occupancy-grids.html.

[8] MATLAB. *Map Environment for Motion Planning using LiDAR*. 2024. URL: https://in.mathworks.com/help/uav/ug/map-environment-motion-planning-using-uav-lidar.html.

[9] Wei Dai, Bizhao Pang, and Kin Huat Low. "Conflict-free four-dimensional path planning for urban air mobility considering airspace occupancy". In: *Aerospace Science and Technology* 119 (2021), p. 107154. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2021.107154. URL: https://www.sciencedirect.com/science/article/pii/S1270963821006647.

[10] Pengcheng Wu et al. *Probabilistic Guaranteed Path Planning for Safe Urban Air Mobility Using Chance Constrained RRT*. 2022. arXiv: 2109.04606 [cs.RO].

[11] Flavia Causa, Armando Franzone, and Giancarmine Fasano. "Strategic and Tactical Path Planning for Urban Air Mobility: Overview and Application to Real-World Use Cases". In: *Drones* 7 (Dec. 2022), p. 11. DOI: 10.3390/drones7010011.

[12] Mangal Kothari, Ian Postlethwaite, and Da-Wei Gu. "Multi-UAV path planning in obstacle rich environments using Rapidly-exploring Random Trees". In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. 2009, pp. 3069–3074. DOI: 10.1109/CDC.2009.5400108.

[13] MATLAB. *Plan Mobile Robots path using RRT*. 2024. URL: https://in.mathworks.com/help/nav/ug/plan-mobile-robot-paths-using-rrt.html.

[14] MATLAB. *Motion Planning with RRT for Fixed-Wing UAV*. 2024. URL: https://in.mathworks.com/help/uav/ug/motion-planning-with-rrt-for-fixed-wing-uav.html.

[15] Mohamed Badreldin, Ahmed Hussein, and Alaa Khamis. "A Comparative Study between Optimization and Market-Based Approaches to Multi-Robot Task Allocation". In: *Advances in Artificial Intelligence* 2013 (Jan. 2013), p. 11. DOI: 10.1155/2013/256524.

[16] Farouq Zitouni, Saad Harous, and Ramdane Maamri. “A Distributed Approach to the Multi-Robot Task Allocation Problem Using the Consensus-Based Bundle Algorithm and Ant Colony System”. In: *IEEE Access* 8 (2020), pp. 27479–27494. DOI: 10.1109/ACCESS.2020.2971585.

[17] Na Geng et al. “How Good are Distributed Allocation Algorithms for Solving Urban Search and Rescue Problems? A Comparative Study With Centralized Algorithms”. In: *IEEE Transactions on Automation Science and Engineering* 16.1 (2019), pp. 478–485. DOI: 10.1109/TASE.2018.2866395.

[18] Michael Otte, Michael Kuhlman, and Donald Sofge. “Auctions for multi-robot task allocation in communication limited environments”. In: *Autonomous Robots* 44 (Mar. 2020), pp. 1–38. DOI: 10.1007/s10514-019-09828-5.

[19] Han-Lim Choi, Luc Brunet, and Jonathan P. How. “Consensus-Based Decentralized Auctions for Robust Task Allocation”. In: *IEEE Transactions on Robotics* 25.4 (2009), pp. 912–926. DOI: 10.1109/TRO.2009.2022423.

[20] Wanqing Zhao, Qinggang Meng, and Paul W. H. Chung. “A Heuristic Distributed Task Allocation Method for Multivehicle Multitask Problems and Its Application to Search and Rescue Scenario”. In: *IEEE Transactions on Cybernetics* 46.4 (2016), pp. 902–915. DOI: 10.1109/TCYB.2015.2418052.

[21] Mi Yang et al. “A distributed task reassignment method in dynamic environment for multi-UAV system”. In: *Applied Intelligence* 52 (Jan. 2022). DOI: 10.1007/s10489-021-02502-3.