ScuDo

Scuola di Dottorato ~ Doctoral School

WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation

Doctoral Program in Electronics and Telecommunications Engineering (31^{th}cycle)

Communication-Aware UAV Path Planning

By

Afshin Mardani

******

Supervisor(s):

Prof. Marcello Chiaberge, Supervisor

Prof. Paolo Giaccone, Co-Supervisor

Doctoral Examination Committee:

Prof. Igor Bisio , Referee, University of Genova

Prof. Paolo Gay, Referee, University of Torino

Politecnico di Torino

2019

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Afshin Mardani
2019

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my loving parents

Acknowledgements

And I would like to express my sincerest gratitude to my supervisors, Prof. Marcello Chiabrge and Prof. Paolo Giaccone for providing me the opportunity to work with them on this thesis. I appreciate their constant guidance and motivation through my PhD period. I would like to specifically express my deepest appreciation to my co-supervisor Prof. Paolo Giaccone who has supported me throughout this work with his patience, vast knowledge, and amazing brilliance. I greatly appreciated the kindness and honesty that were part of every interaction we had. I also would like to thank my friends and PhD students of LIM for holding valuable discussions which helped me immensely.

Finally, I would like to thank my beloved family for their support throughout my studies. There are no words that can fully express my gratitude to my parents, my sister and her husband, and my dear girlfriend Farsima. This thesis is dedicated with love to them.

Afshin Mardani, 2019

Abstract

Autonomous air drones, known as Unmanned Aerial Vehicles (UAVs), often accomplish missions with real-time video streaming leveraging the available cellular network. Video streaming is a typical application with stringent Quality of Service (QoS) requirements, which are not always supported in the whole mission area. In this thesis, we address the offline path planning problem of finding the optimal path from a source to a destination in 2D space in order to maximize the communication quality, given a cellular coverage, and thus providing throughput guarantees to the video streaming. In addressing the problem, the restricted amount of available energy, the wind effect, and the path post-smoothing problem are considered. We propose two innovative path planning algorithms and we show that our algorithms outperform classical approaches that are oblivious of communication network coverage. Both algorithms are variants of classical A* algorithm and they optimize the path jointly in terms of distance and of the experienced throughput by the drone: in this way, the quality of the video streaming along the path is optimized while preserving the energy budget for the flight. We describe both of the algorithms and investigate their performance. Moreover, we introduce a novel path smoothing method that outperform classing approaches in terms of distance and computation cost. Finally, in order to prove that our algorithms can be practically utilized in the real-world path planners, we integrate our proposed path planning algorithms into the popular QGroundControl (QGC) control station.

Contents

Chapter 1

Introduction

1.1 Introduction

Autonomous air drones are becoming increasingly popular and typically are equipped with real-time video streaming for surveillance and safety purposes. In recent years, the UAVs' excessive variety of applications including delivery, search and rescue, and inspections, require a highly secure and reliable connection. Focusing on this objective leads to support the UAVs on cellular network coverage. Wireless technology can bring many privileges to UAVs such as ubiquitous coverage, high-speed mobile support, robust security, high reliability and quality of service (QoS). This aim has been attracted the attention of industry and companies [1], to provide cellular connectivity for UAVs. For instance, telecommunications leading companies like Qualcomm and AT&T have set up to develop the connectivity technologies, including optimization of LTE networks and advancement of 5G technology for drones [2]. The 3rd Generation Partnership Project (3GPP) is started the study on enhanced LTE support for UAVs in 2017 [3]. Accordingly, UAVs can leverage the cellular network infrastructure to support the video streaming communication, in case of long-distance flights. When planning an autonomous path between a set of distant way-points, we advocate the adoption of path-planning algorithms which are aware not only of the obstacles (as in classical approaches) but also of the cellular coverage, in order to guarantee the stringent requirements of Quality of Service (QoS) in the communication for video streaming.

In this work, an offline path planning problem between two generic way-points, taken into account the provided budget of energy and the required bandwidth in terms of either average or minimum throughput is addressed. In addition, the cellular coverage map and possible effects of the wind that could have an impact on the path trajectory is considered. Moreover, we address the smoothing problem of the resulting paths of our path-planning algorithms, for the purpose of compensating for the spatial sampling of the graphs needed for computation and obtaining straight flying trajectories.

Our main contributions are manyfold.

1. We propose two innovative path-planning algorithms, which are variants of the popular A* algorithm. The main focus of AT-PP (Average Throughput Path Planning) algorithm is to maximize the average throughput along the path, with the purpose of guaranteeing an average level of communication QoS. The main focus of MT-PP (Minimum Throughput Path Planning) algorithm is instead to maximize the minimum throughput along the path, with the purpose of guaranteeing a minimum level of communication QoS. Each of the two algorithms take into account the on-board accessible amount of energy for the flight and lengthen the path for the sake of optimizing the communication quality.

2. By simulation, we evaluate the performance of the MT-PP and AT-PP algorithms, in both presence and absence of the wind. It is numerically shown that our communication aware approach is capable of outperforming classical approaches (oblivious of cellular coverage) significantly, concerning throughput, without exceeding the constraint of energy. Furthermore, we investigate the effect of energy budget on the resulting paths of MT-PP and AT-PP, and we show how the energy budget can affect the resulting path.

3. A novel path post-smoothing approach (IPS) is proposed that leads to a reduction in the computation time comparing with classical approaches.

4. We integrate seamlessly our path planning algorithms into the open-source QGroundControl control station [4], in order to modify the path between way-points considering the cellular coverage map. We prove that our algorithms are practical, by employing them in a real-world path planner.

This thesis is organized as follows. In Chapter 2, we provide a review to the required preliminaries for a better understanding of the presented work. First we investigate the universal applications of the UAVs. We present some statistics referring to the rapidly growing commercial market and academic interest about the drones. We describe the fundamental path planning algorithms which are basically utilized in this work. Next, we concentrate on the environmental disturbances in path planning and specifically we investigate the effect of wind, and how drones are dealing with these disturbances. Some popular techniques to compensate the wind effect is presented and how we compensated for the wind is specifically explored. Moreover, we explore the effect of restricted on-board energy budget on the path planning methods, and the factors that should be considered in energy calculation for path planning are described. The effect of wind in energy consumption is explored as well. Finally, employing cellular network technology to provide support for drones in order to enhance communication QoS is investigated. The reasons of requiring a much more advanced communication for drones and how cellular network technology fulfills these requirements are explained. Furthermore, the challenges that UAVs are facing, the communication requirements for UAVs, and the factors to enhance the cellular network performance for establishing a productive coexistence between the terrestrial and aerial users are described.

In Chapter 3, our novel communication-aware path planning algorithms are introduced. We describe the model of the flight area, the drone mobility model, the energy constraints, and the mobility planning of drone between each two nodes are explained. We introduce the AT-PP algorithm that maximizes the average throughput along the path and the MT-PP algorithm that maximizes the minimum throughput along the path, both subject to the energy budget. The algorithms are well explained and the pseudo-codes of them are provided in detail. Finally, the post smoothing methods to smooth the resulting paths including our novel method are explained.

In Chapter 4, the integration of our algorithms in a real-world path planner is explored. We investigate the QGroundControl (QGC) as apowerful open-source control station and its communication with the autopilot. The data flow between the QGC and autopilot, some specific commands, and the communication sequence of uploading a mission to a vehicle is explained in more detail. Then, we introduce our designed proxy for monitoring the data which are passing between the QGC and vehicle. The two faced behavior of the proxy, acting transparently or not, is analyzed in depth.

In Chapter 5, The numerical results of our experiments and the performance of algorithms are presented. Some comparisons based on the path length, computation time, and energy consumption are presented. Our results are compared with the-state-of-the-art algorithms.

Finally in Chapter 6, we conclude the thesis.

Chapter 2

Preliminaries

In this chapter, first we concentrate on the extensive applications of drones. We show that the drone commercial market and the academic interest about drones are growing rapidly. A prediction of drone potential market value of business services is presented. Next, a taxonomy of the existing path planning methods is presented. We introduce the path planners in three groups of graph search based algorithms, sampling based algorithms and combinatorial algorithms. We explain some fundamental path planning methods that we have used throughout this work. We describe Dijkstra's algorithm [5], that is a graph based algorithm and finds the shortest path from a single-source to any nodes of the graph. A* [6] algorithm is introduced which is an extension of Dijkstra's algorithm and calculates the shortest path in a much shorter computation time than the canonical Dijkstra's algorithm, by considering a smaller search subspace. Then, we explore the wind effect in path planning. We discuss about the conditions and the most important challenge, i.e. wind, that the UAVs are facing when they operate in real-world environments and how UAVs are dealing with it. We introduce some popular techniques to compensate for the effect of wind. After that, we investigate the significance of energy consumption and energy restrictions in path planning. Specially, the flying status of UAVs such as acceleration, deceleration, velocity, and turning angle on the consumption of energy is investigated. We show that energy consumed in turns, taking off and landing is not negligible. In addition, we specifically explore the effect of wind on the energy consumption of the UAVs and describe the method that we used in our work to anticipate the energy consumption in windy situations. Finally, the employment of cellular network technology for drones and how the leading companies have setup

to develop this technology in order to provide support for the drones enhancing safety, control, and communication QoS is investigated. We describe the reasons that a much more advanced communication is required for modern drones and the strong points of cellular network technology that fulfills these requirements are explained. Additionally, we mention the network coverage and interference as the most important challenges that UAVs are facing in exploiting the cellular networks. The communication requirements for UAVs and how the cellular network is meeting those requirements are studied. Furthermore, we look into some factors in order to enhance the cellular network performance for enabling a productive coexistence between the terrestrial and aerial users.

2.1 Applications of Unmanned Aerial Vehicles

In recent years, the accessibility and performance of Unmanned Aerial Vehicles (UAVs) has improved vastly, due to the numerous applications of this new technology. Currently, in the US and EU, the operations of drones are restricted significantly (e.g. keeping the vehicle in the operator's line of sight) in order to safeguard the airspace, people, and their properties. Since the the UAV technology is improving (e.g. improvement in collision avoidance systems), and as interested parties gain experience in UAV technology, a significant relaxation in the regulations is expected. Thus, predictably a new range of interesting applications for UAVs will be opened. Base on the market forecast, commercial drone market is developing rapidly. As an example, the Teal Group Corporation which is under the supervision of Federal Aviation Administration (FAA), predicts that the commercial drones operation fleet in the US will be about 3.3 billion dollars in purchase value, by 2020 [7]. Fig. 2.1 indicates the growing academic interest about the aerial vehicles topic [8].

Uncrewed technology can be beneficial to many industries, since it reduces the cost. This technology can lower the weight of the vehicle, and consequently the energy consumption of the UAVs. Moreover, it reduces the cost by making the vehicle's cockpit and its facilities unnecessary. UAVs are able to operate in unreachable and hazardous environments for humans. Fig. 2.3 illustrates some of the civil applications [9]. Drones have already performed some commercial applications [10], [11], and new ideas for new applications are appearing in the media. Some of the applications are discussed as following:

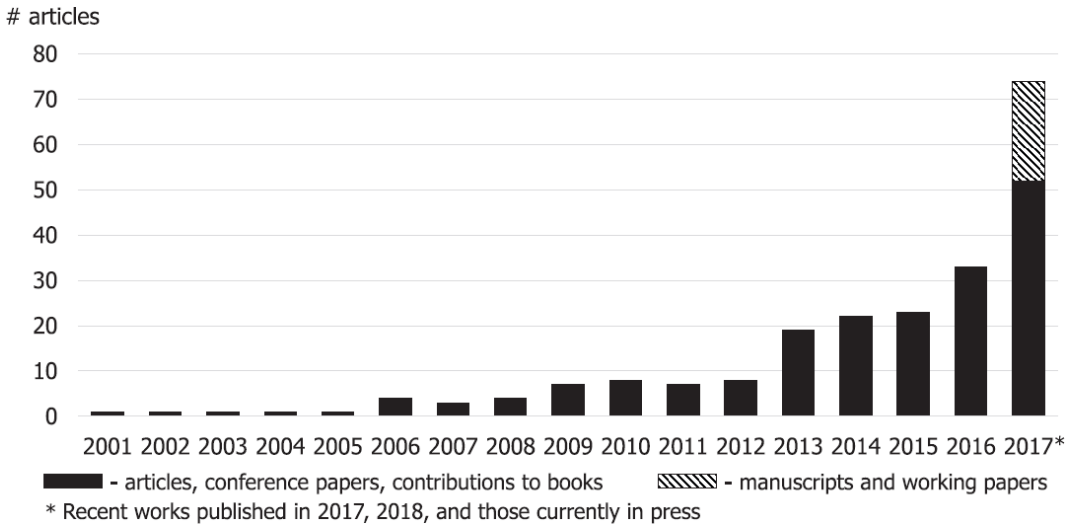

Fig. 2.1 Academic interest in aerial vehicles is growing dramatically. Reproduced from [8].

- UAVs have already been employed for accessible infrastructures like roads, oil and gas pipes, and railways in order to track the construction progress, and to regularly inspect for maintenance. In addition to providing a high quality video recordings, UAVs offer a safe operation in risky environments. For instance, the first pilot project of Lufthansa and its partner DJI, utilized UAVs to inspect rotor blades of wind turbines [12].

- Drone technology is very beneficial in agriculture. Drones are able to monitor crops, investigate crop health, spray fertilizers, and assess the soil. Market analysts predict that the largest application of drones in the US will be in the agriculture in the few coming years [13]. Fig. 2.2 shows the YAMAHA RMAX Japanese agriculture drone that have been used for over 20 years in countries like Australia and South Korea. It can carry a payload of up to 31 kg and has a flight time of up to 1 hour.

- Delivery and transport is another interesting application of drones. Medical supplies like vaccines and blood can be delivered very fast with a low cost. The quality of medical service can be improved significantly by using drones in developing countries. Rwanda, for instance, has begun Zipline commercial drone deliveries from 2016 [10]. This operation is done by fixed-wing drones that fly to destinations automatically. Drones release the packages without landing at the delivery point. The deliveries are faster than previously been possible by road.

Fig. 2.2 RMAX, a very popular drone which has been utilized for many years in agricultural applications. Reproduced from [14].

Fig. 2.3 Examples of UAV civil applications. Reproduced from [9].

There is a huge potential market value in all of the drone applications and in order to take advantage of this potential, companies should deploy UAVs in civilian airspace safely an at the lowest cost possible. In [15], it is predicted that the potential market value of business services that may take advantage of drone technology may reach several billion dollars (as shown in Fig. 2.4) in some industries.

2.2 Path Planning Algorithms

Path planning problems typically are solved exploiting a method included in one of the three categories: graph search based algorithms, sampling based algorithms, and combined algorithms (see Fig. 2.5). Currently, graph search based algorithms

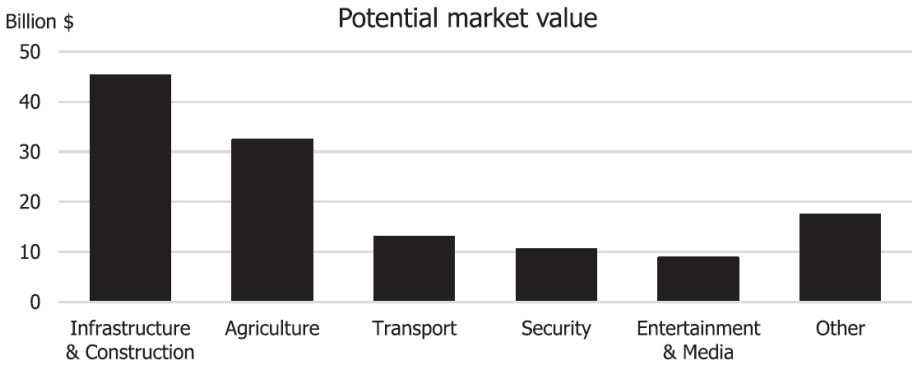

Fig. 2.4 Estimated potential market value of drone applications. Reproduced from [15].

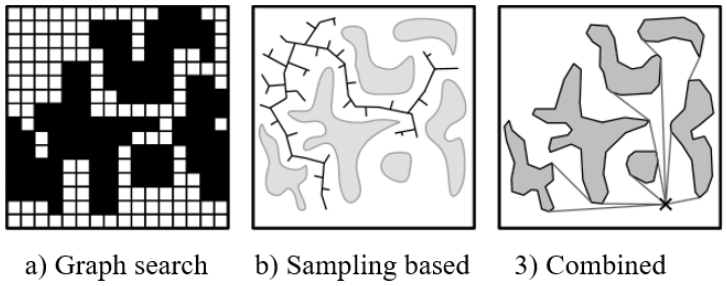

Fig. 2.5 Categories of different path planning methods. Reproduced from [16].

are dominating the other methods in path planning for robots [16]. The popularity of graph search based algorithms can be attributed primarily to simplicity of their implementation and secondly to the early establishment of them. In this section, we attempt to have a quick review on all the three categories and provide their advantages and limitations briefly.

2.2.1 Graph Search Based Algorithms

Graph search-based algorithms are based on the fact that, a grid graph is employed to represent the search space. In order to represent grids as graphs, each cell of the grid represents a node in the graph and edges of the graph connect the adjacent cells. Search based algorithms specify an order to search the graph nodes. In fact, the starting point of the robot and the destination is known and a search algorithm is run to find a point to point path on the graph.

Algorithms

Graph search based algorithms compute the path cost by exploring through the nodes to figure out the optimal path. They have experienced a series of technological advances during the last decades. Dijkstra algorithm is the first algorithm that finds the shortest path. One of the common variants of Dijkstra algorithm, fixes a node as *source* node and finds the shortest paths from the source to all the other nodes of the graph, generating a shortest path tree. A few years later, the A* algorithm by utilizing an admissible heuristic in order to narrow the search space, is introduced. A* finds the shortest path faster than any other methods. A* as a static algorithm, rerun the search from the scratch if the search area configuration changes (e.g. a shift in the configuration of obstacles). We explain Dijkstra and A* algorithms with more details in the following sections. In 1994, D* algorithm, the first dynamic search based algorithm is introduced [17]. D* algorithm, also called dynamic A*, is known for being the basis of planning in the DARPA UGV programs [18]. D* algorithm is a sensor based algorithm and deals with dynamic obstacles. To this end, D* changes the weights of edges in real time to establish a temporary map and then guides the robot from the source to the destination. D* algorithm, as well as A*, calculates the cost through a forward estimation. Afterwards, Quad-tree D* and Focused D* are the algorithms that were introduced to deal with the time complexity and space complexity limitations of the D* algorithm, respectively [19], [20]. Koenig and Likhachev proposed an algorithm based on LPA* which is similar to a simplified version of D* and is called D* Lite [21]. D* Lite is very popular as it is similar to the Focused D* from the behaviour point of view, but simpler to understand and to code. Actually, D* Lite is being used in most of the current implementations in the Stentz's lab (where D* was born).

Limitations

We should not undervalue the success of search-based methods. However, search based algorithms are subject to some acknowledged limitations. On one hand, searching on a grid removes the burden of maintaining the graph structure. On the other hand, since grid graphs have a fixed topology, graph based methods are subject to the resolution of the grid. Therefore, the resulting path is optimal only at the employed grid resolution. There is a trade-off between increasing the resolution of the

Table 2.1 Analysing search based algorithms. Reproduced from [22].

Algorithm	Shortcomings	Advantages
Dijkstra's	High computational time Static environment only	Easy implementation in different environments
A*	Heavy time burden Static environment only Post smoothing needed	On-line implementation Fast search ability
D*	Poor distance estimation	Fast search ability Adopted to dynamic environments

grid (and therefore resulting paths closer to real optimal path) and computation time. Table 2.1 provides a straightforward analysis of search based optimal algorithms.

2.2.2 Sampling Based Algorithms

Sampling based algorithms deal with the graph generation problem in the following way. First, generating a new vertex v. Then, checking if v is inside of an obstacle. If this condition is met, the cycle starts over or v needs to be projected to the boundary of the obstacle. Afterwards, edges are connected between v and the visible neighbours of v. As a result, a path exists between the source and destination vertices. As long as a proper path is investigated, A* search runs over the graph to find the resulting path. Since the basis of vertex generation is probabilistic, this approach was initially introduced as Probabilistic Road Map (PRM) [23], although it is allowed to be done deterministically by the framework as well.

RRT (Rapidly-exploring Random Tree) method is demonstrated similarly [24]. RRT searches the space rapidly to find a path between the source and destination. This method is proven to be working very effective on high dimensional spaces. Moreover, this method is recently adopted for dynamic configurations [25]. The characteristic of being an overall simple method made the RRT interesting. In order to implement this method, two challenges are in front. First, a kd-tree must be implemented. kd-tree is a data structure in order to sort k dimensional vertices, and it makes the range and nearest neighbor searching operations possible. Second, a $LineOfSight(v_1, v_2)$ method is needed to discover whether an edge between vertices v_1 and v_2 does exist or not.

The Voronoi diagram was introduced by Shamos and Hoey [26] in the field of computational geometry. After a series of form improvement, now is widely utilized in the field of path planning. Similar to PRM, a global or local graph is created by Voronoi, the shortest path can not be generated at the same time. Accordingly, Voronoi needs Dijkstra's, A*, or D* algorithm in order to find the shortest path. Three steps of Voronoi path generation is as following: first, sampling the environment or utilizing other environment construction methods. Second, 3D Voronoi graph generation. Third, employing a search algorithm to find the shortest path.

Artificial potential field methods was initially introduced by Khatib [27]. This method has been widely researched and implemented due to a low computational complexity. In this method, a potential function is assigned to the configuration space and the vehicle reacts to the potential field force like a particle. The potential function calculates the destination absorption and obstacle repulsion in the same time and leads the vehicle along the total force gradient. However, the artificial potential field methods are prone to be stuck in a local minima. A lot of research has been done, such as generating navigation functions, to overcome this shortage [28–30].

Sampling based algorithms depend upon the concept of either probabilistic completeness or resolution. It means that the resulting paths of these algorithms are optimal solution with a converging probability to 1 as the number of vertices grows accordingly.

The fundamental rule of appending vertices to a graph randomly and checking the connections, comparing with the search based methods, is not the most biologically inspired technique or in another word the most natural approach to path planning. However, it does not affect the apparent performance of sampling-based methods, but may affect one's decision to look for a fast yet method and more natural. Table 2.2 provides a straightforward analysis of sampling based algorithms.

2.2.3 Combinatorial Path Planning

In combinatorial path planning, the search space is essentially represented by a polyhedral that additional vertices and edges are added to it. As a result, a graph which is know as *road-map* is created. Similar to the sampling based path planning methods, this method looks for the shortest path from the source to the destination. The difference is that, where the sampling based method populates the free config-

Table 2.2 Analysing sampling based algorithms. Reproduced from [22].

Algorithm	Shortcomings	Advantages
RRT	Single path Static environment only Not optimal	Fast search ability Low computational time
PRM	Costly collision check Static environment only Not optimal	Powerful in complex environments Appropriate for re-planning
Voronoi	Static environment only Not convergent	Appropriate for on line planning
Artificial potential	Local minima	Converges very fast

uration space with new vertices, the combinatorial methods employ the polygonal obstacle boundaries directly. The are three types for the road-maps: a) shortest path, b) maximum clearance, and c) cell decomposition. In this thesis, we just consider the shortest path road map which is the *visibility graph* [31]. In visibility graphs, an edge is added between each two mutual in line-of-sight pair of vertices. The visibility graph has a variety of applications such as VLSI design, art gallery problems, computer graphics, and in pursuit evasion problems for multiple robots.

In visibility graphs, the graph is formed by adding the edges to the nodes in the graph (node in the graph represents a point location) if there is a visible connection between them (in other words there is a line-of-sight between them). Therefore, if the connecting line between two locations does not pass through an obstacle, an edge is drawn. As a result, the euclidean shortest path problem is decomposed into two sub problems: a) visibility graph construction and b) employing a shortest path algorithm such as A* to the graph. Figure 2.6 shows a visibility graph example.

The graph shown in Figure 2.6 is ridiculously dense from the path planning point of view and most of the edges are not essential. The *reduced visibility graph* (RVG) is introduced in [31]. Therefore, the essential edges based on necessary local conditions are added.

There exist some limitations for the visibility graphs. For example, extending the visibility graph as a road-map for more than 2D spaces is impractical. In [32] is shown that point to point planning in 3D spaces with polyhedral obstacles is a NP-hard problem. However, some approximate solutions have been presented in [33–35].

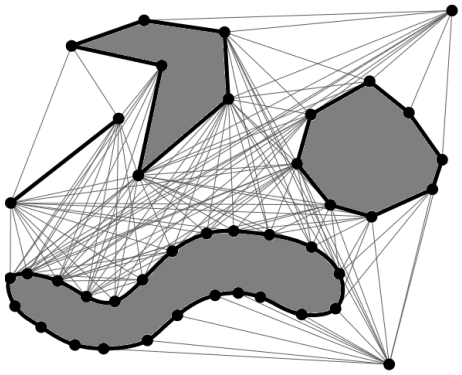

Fig. 2.6 Visibility graph example.

2.2.4 Shortest Path Algorithms

Search based methods utilize graph search algorithms to explore a path or trajectory over a discrete representation of the problem. Graph search algorithms explore paths between nodes, initializing from one node and traversing through the edges as far as the destination is reached. In the following sections, we investigate two of the most popular and fundamental graph search based algorithms in robotics.

Dijkstra's Algorithm

Dijkstra's algorithm is a graph based algorithm that finds the shortest path between the nodes. Considering a graph $G = (V,E)$ which is consist of a set of vertices V and edges E, and a non-negative cost $c(u,v)$ is associated to each (u,v), Dijkstra's algorithm is able to fix a node as *source* ($u_{source} \in V$) node and find the shortest path to all the graph nodes. This algorithm can be employed in many path planning applications. Network routing protocols is one of the applications of Dijkstra's algorithm which is extensively used. For instance, if the graph nodes represent the cities and edges of the graph represent the roads, Dijkstra's algorithm can be utilized to find the shortest paths from one city as source to all of the other cities. The associated cost to each edge is usually the distance between the two nodes, but other costs like safety and traffic can be considered in the cost function.

Dijkstra's algorithm operates by assigning initial cost values to all of the nodes (except the source), and improves them progressively. First, it marks all the nodes as unvisited nodes and collects them in a set which is called *unvisited set*. Next, the algorithm assigns a preliminary cost value which is not finalized: sets the source

node as current node and a cost value of zero and sets infinity to all the other nodes. Considering the current node, the tentative cost to reach all of its neighbors which are in unvisited set are calculated. It compares the most recent calculated cost with the previous assigned cost and updates the value with the lower one. As long as all the unvisited neighbors of the current node are visited, it marks the current node as visited and removes the current node from the unvisited nodes set. The algorithm never visits a visited node for the second time. To this end, in the event that the planned destination node is marked as visited (destination is reached), or the lowest tentative cost to the planned destination is infinity (among the other neighbors), the algorithm terminates. In the latter case the destination is unreachable. Differently, the algorithm selects the neighbor with the lowest tentative cost as current new node and repeats the previous steps. The pseudo-code of the Dijkstra's algorithm is shown in the Algorithm 1. In the following algorithm, cost $[u, v]$ returns the cost of moving on the connecting edge of u and v. The *parent* array is filled with the previous nodes from the destination to reach the source with the shortest path. S represents the source node and $\mathscr{N}$ represents the graph.

The running time of Dijkstra's algorithm on a graph can be expressed as a number of edges ($|E|$) and number of vertices ($|V|$) function. The bound tightness depends on the implementation of set $\mathscr{Q}$. In the simplest implementation of Dijkstra's algorithm, that the *minimum finder* is a linear search through $\mathscr{Q}$ set, the running time is $O(|E| + |V|^2) = O(|V|^2)$. The optimal running time can be achieved by implementing $\mathscr{Q}$ with Fibonacci heap [36], and is equal to $O(|E| + |V|log|V|)$. Practically, the first implementation is usual, as we used in some of our experiments in this thesis.

A* Algorithm

Dijkstra's algorithm finds the shortest paths to all the nodes included in the graph. In some cases, it is not only required to find the shortest path from a source to a specific single destination, but also required to optimize the performance of finding the shortest path (e.g. very large map or a time-critical application). In this case, A* algorithm, which is based on Dijkstra's algorithm, is preferable. A* focuses on searching the graph towards the destination and it performs the search based on the path cost up to the current node and an estimation of the cost required to reach the destination. However, in Dijkstra's algorithm, the search is broadwise. Furthermore,

Algorithm 1 Dijkstra's Algorithm

```
1:  function DIJKSTRA(𝒩, S)
2:      Create vertex set 𝒬
3:      for each vertex v ∈ 𝒬 do
4:          cost [v] = ∞
5:          parent [v] = undefined
6:          add v to 𝒬
7:      cost [S] = 0
8:      while 𝒬 is not empty do
9:          u = vertex in 𝒬 with min cost to u
10:         remove u from 𝒬
11:         for each neighbor v of u do
12:             if cost [u] + cost [u, v] < cost [v] then
13:                 cost [v] = cost [u] + cost [u, v]
14:                 parent [v] = u
15:     return cost[ ], parent[ ]
```

it is still guaranteed that A* find the optimal path. A* picks the path in which the f function is minimized:

$$f(n) = g(n) + h(n) \tag{2.1}$$

Where function g is the path cost from the source to the current node and the heuristic function h represents the estimation of the cost from the current node to the destination.

Path planning performance of A* depends on the quality of the heuristic. The value of heuristic should be a lower bound estimation of the cost (cost is assumed to be only distance) from the current node to the destination. Particularly, if the heuristic function gives exact distance to the destination (the estimated distance is equal to the distance on the grid), the algorithm scans only nodes on shortest path from source to destination. If the heuristic function overestimates the actual distance to the destination, it is not guaranteed that the discovered path is the optimal path. Therefore, choosing a proper heuristic function is a significant task. In this thesis, our path planning algorithm is used with the straight-line distances (Euclidean distance). The reason behind is that post-smoothing shortens the resulting path much more than other heuristic functions (e.g. Octile distance), at the cost of higher computation. In this case, the cost function g does not match the heuristic function h, that is, the

Algorithm 2 A* Algorithm

```
1:  function A*(𝒩, S, D)
2:      𝒰 = 𝒩 \ {S}                                  ▷ Unvisited nodes
3:      ℱ = {S}                                      ▷ Frontier nodes
4:      𝒱 = ∅                                        ▷ Visited nodes
5:      for each vertex v ∈ 𝒩 do
6:          g(v) = ∞
7:          h(v) = estimated cost from v to D
8:          parent [v] = undefined
9:      g(S) = 0
10:     parent [S] = S
11:     while ℱ is not empty do
12:         u = vertex in ℱ with min cost to u
13:         remove u from ℱ to 𝒱
14:         if u = D then                            ▷ Check if arrived to destination
15:             return
16:         for each neighbor v ∉ 𝒱 of u do
17:             if cost [u] + cost [u, v] < cost [v] then
18:                 cost [v] = cost [u] + cost [u, v]
19:                 parent [v] = u
20:                 if v ∈ 𝒰 then
21:                     move v from 𝒰 to ℱ
22:     return unreachable destination
```

increase in the g function is not exactly equal to the decrease in the h function, and consequently, A* takes longer to run. Therefor, for the 8-degree grids (we employ in this work), we use the Euclidean distance from the source to the destination as a heuristic function. The pseudo-code of the Dijkstra's algorithm is shown in the Algorithm 2.

A* algorithm creates three sets of nodes. The frontier nodes ($\mathscr{F}$) are stored in a priority queue and the visit procedure expands from them. The visited nodes ($\mathscr{V}$) are the nodes that have been already visited and will not be visited anymore. The unvisited nodes ($\mathscr{U}$) are the remaining nodes, i.e. not frontier nodes and not yet visited. In Algorithm 2, S represents the source node, D represents the destination node and $\mathscr{N}$ represents the graph. Our novel algorithms that we will present in this work are variants of A* algorithm.

2.3 Exploring the Wind Effect in Path Planning

When UAVs are operating in real-world environments, dealing with the wind effect that leads the aircraft to be drifted in a certain direction, is indispensable. In this situation, it becomes much more difficult for UAVs to follow certain trajectories. In order to successfully and precisely execute a planned trajectory, the trajectory must be designed by considering the physical capabilities of the vehicle and the effect of wind.

There exist many different techniques to compensate for the effect of wind. We note some of the popular techniques in the following.

2.3.1 Techniques to Compensate the Wind Effect

One technique to compensate for wind is employing the Monte-Carlo method [37]. In order to represent the state of the vehicle by maintaining a set of the vehicle's probable states, the Monte-Carlo method is utilized. Employing this sampling based representation, a prevision of the flying trajectory is reached. Hence, a prediction of the trajectory can be generated which includes multiple uncertainty sources and perturbations with arbitrary distributions in the model. In order to apply this method, models of the vehicle and atmosphere is required. A simple vehicle model which takes into account the heading references and the uncertainty of the wind, can be utilised to decrease the computations in real-time implementation:

$$x = v_i cos(\psi_i) + \omega_p cos(\omega_\varphi) \tag{2.2}$$

$$y = v_i sin(\psi_i) + \omega_p sin(\omega_\varphi) \tag{2.3}$$

$$\psi_i = \alpha_\psi(\psi_i^c - \psi_i) \tag{2.4}$$

Where (x_i, y_i) represents the two dimensional coordinates, ψ_i represents the heading of the vehicle, α_ψ is a known parameter and depends on the characteristics of the vehicle, and ψ_i^c is the heading reference to the control system. The atmospheric model is composed of the wind speed and direction. ω_p and ω_φ represent the wind vector speed modulus and direction respectively in the atmospheric model. Due to the nature of wind, wind direction strongly depends on the local terrain, mesoscale

and large scale considerations. Usually the wind direction is forecast utilizing complicated numerical models. It is modeled here utilizing a Normal distribution of mean $\overline{\omega}_\phi$ and standard deviation ω_ϕ^σ, without loss of generality. Moreover, wind speed modeled perfectly with Weibull distribution [38] at low altitudes. In the final model that considers the wind and vehicle's heading, the model parameters can be estimated by real flight data.

Another approach in dealing with the effect of wind on path planning is proposed in [39] by solving a no-wind case problem with a moving virtual target. In this method, the velocity of wind and the velocity of virtual target are equal but in the opposite direction. Therefore, the defined path for the vehicle in this problem is considered with respect to the moving air frame and is known as "air path". Additionally, the path of vehicle in the presence of wind with respect to the ground is known as "ground path". The equations of motion are presented in [39] with all details. The ground path produced by the algorithm is tracked by the control algorithm. The control algorithm breaks the designed path into smaller paths leading each one to be approximated by a polynomial. Then, a spatial sliding surface controller in order to track each polynomial in presence of wind, is utilized. It is proved that the whole system is robust to the wind disturbances.

One of the common strategies to compensate for the wind in path following is considering the inertial ground speed of the UAV, which includes the wind effects inherently [40], [41]. In [40], a guidance algorithm for a UAV is used to fly along a path which is consist of several way-points. An observer based wind estimator is included in the guidance algorithm. The perfect wind information are fed into the look-up table directly. Thus, considering the estimated speed of wind and direction, the air speed and desiring course change, a proximity distance (that is obtained from a look-up table) for each way-point is defined. This proximity distance leads to a smooth converge to any new course by the aircraft without over shoot or under shoot in strong wind situations. In fact, this look-up table as an efficient method, chooses the proper look-ahead distance, responding to a course change, wind heading and speed, and air speed. In [41], a method based on vector fields to generate course inputs to inner-loop control laws for accurate path following is developed. Vector fields can be used to track straight-line and circular paths asymptotically, in presence of strong wind. However, it is noteworthy to mention that tuning of vector fields is known to be complicated. In this method, the motion equations are represented in terms of ground speed and course, which are independent of the wind velocity.

Therefore, the wind disturbance rejection by utilizing course and ground speed (instead of heading and air speed) together with the vector field for path planning, is improved significantly. Moreover, the path following errors in the presence of wind is decreased significantly by using Lyapunov stability arguments.

One explicit approach is considering the wind by directly measuring the wind or employing a wind observer [42, 43]. In [42], it is aimed to observe a ground target from a UAV which is affected by the wind. The path follower algorithm is robust to the wind and does not rely on heading or wind knowledge. However, local wind information is required for a constant line-of-sight geometry. The estimate of the wind leads to a approximate solution for the camera field of view, and the error in the wind estimation leads to the target moving in the camera field of view. Therefore, the wind direction and speed must be continuously estimated. Air data instruments measure the air speed, GPS measures the ground speed and course, and wind speed and course relates to the heading of aircraft. The direction and speed of wind can be considered as unknown initial conditions, in case of constant wind, and can be measured by means of an observer. A nonlinear observer can be constructed to estimate the wind direction and speed in case of knowing the aircraft heading. It leads that the vehicle to improve the ability to stabilize the sensors and camera at the target. In [43], an alternative method to design a guidance controller for a UAV is proposed to perform the path following in windy situations. In order to eliminate the the wind effects, an observer-based disturbance control approach is adopted. In a two-step strategy, first a nonlinear disturbance observer estimates the unknown wind, then in order to improve the control performance, the estimates are embedded to the control system design. As a result, the aircraft flies into wind with a trimming angle to compensate for the perpendicular wind component to the path.

One possibility to compensate for the wind is described in [44], estimating the direction of the wind through an adaptive back stepping procedure. In this method, the focus of control strategy is on reducing the path deviation of the aircraft from the designed path in the lateral dynamics in presence of wind. By considering the adaptation laws to estimate the wind parameters, the control scheme is derived. The robustness of the control system is proved by considering unknown wind gusts in simulations.

Another popular approach is founded on a nonlinear guidance logic to track the trajectories [45]. This method, in the computation of commanded lateral acceleration,

utilizes inertial speed and in case of external disturbance, such as wind, adds adaptive capability to the speed change of vehicle due to the wind. The performance of this controller is compared with linear (PD and PID) controllers. It is revealed that the vehicle is moving around the desired path with a cross-track error in a range about 20 to 60 meters, using linear controllers. The linear feedback controller with a fixed gain, has limitations to immediately remove the errors, due to the inertial speed changes by wind. On the other hand, nonlinear guidance logic follows the path with an error less than 7 meters. The reason behind is that the nonlinear logic method considers the inertial velocity changes due to the wind and corrects the situation accordingly. Great privileges of this method is that it is simply and intuitive to be tuned, the guidance commands' magnitude is always upper bounded, and it is flexible in setting the initial conditions.

Considering the effect of wind not only helps to follow the generated paths more accurately, but also leads to be able to calculate the energy consumption of the aircraft more precisely. We will show in the next section how considering the effect of wind can help us to calculate the energy consumption during the flight.

2.4 Energy Aware Path Planning and the Wind Effects on Energy Consumption

Although UAVs have a lot of different applications, they also face many challenges. In particular, the performance and endurance of UAVs are fundamentally restricted by the energy on-board. The on-board energy is particularly finite as a consequence of vehicle's size and weight restrictions. Therefore, energy efficient path planning for maximizing the performance of the UAV and enabling it to accomplish the mission with a limited amount of energy is of paramount importance.

2.4.1 Flying Status

It is important to mention that the flying status of UAVs (including acceleration, deceleration, velocity and turning angle) is deterministic in the energy consumption of UAV's propulsion. Thus, it requires to be taken into account in energy efficient designs for UAV path planning.

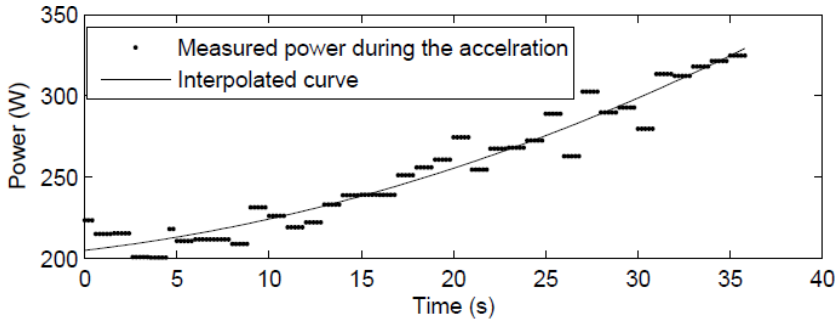

Fig. 2.7 Power consumption during maximum acceleration. Reproduced from [46].

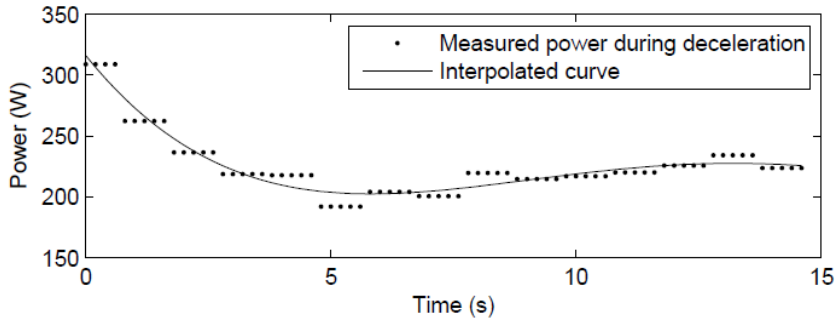

Fig. 2.8 Power consumption during maximum acceleration. Reproduced from [46].

Turning Angle

There are a large variety of drones and each possesses its own unique physical characteristics. Therefore, it is a difficult task to derive a parametric energy model to predict the consumption of energy in different flying conditions. We utilized the model which is proposed in [46], that analyzes the energy consumption for a specific drone type (IRIS quadrotor). A set of experiments are performed to realize the effect of different operating conditions on the energy consumption of the drone. The IRIS quad-copter's weight is about 1.3 kg, equipped with 850 kv motors, and powered with a battery of 3S lipo 11.1 V 5.5 Ah, and a PX4 autopilot. In one of the experiments, the drone accelerated and decelerated at full throttle, and the speed and absorbed current are monitored from the on-board GPS and control board, respectively. The power consumption is calculated from the Ohm's law. Fig. 2.7 and Fig. 2.8, show the power consumed during the maximum acceleration and deceleration, respectively.

In another experiment, the power consumption is calculated in different flight conditions, such as climbing, hovering ($v = 0$), horizontal flight, and landing. The

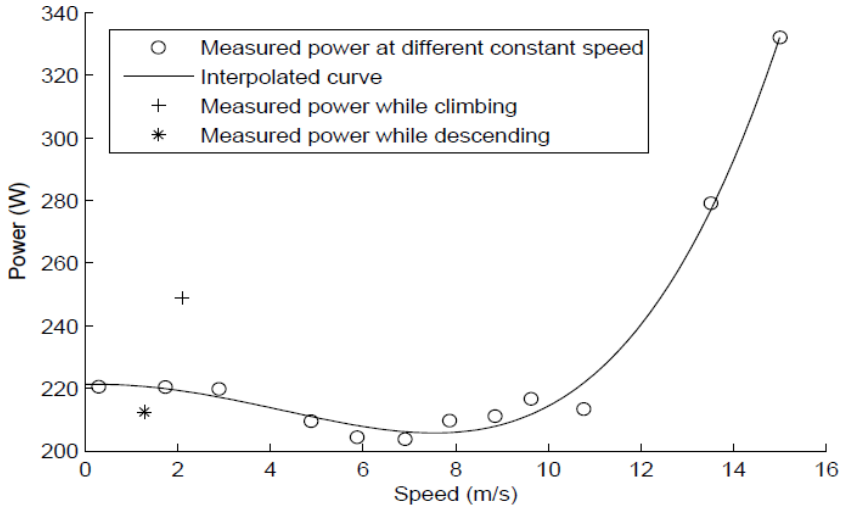

Fig. 2.9 Power consumption as a function of constant speed. Reproduced from [46].

results are shown in Fig. 2.9. Therefore, the energy consumption in a flight with constant speed v is as following:

$$E_v = P(v)\frac{d}{v} \tag{2.5}$$

The consumed energy for climbing and landing from a height of Δh is computed as:

$$E_{climb} = P_{climb}\frac{\Delta h}{v_{climb}} \tag{2.6}$$

$$E_{land} = P_{land}\frac{\Delta h}{v_{land}} \tag{2.7}$$

And, the consumed energy during hovering in a time interval of $[t_1, t_2]$ is computed as:

$$E_{hover} = P_{hover}(t_2 - t_1) \tag{2.8}$$

In the final experiment, the power consumption of drone during the turns is calculated. Considering the angular rotation speed as $\omega_{turn} = 2.1\ rad/s$ and the

power consumption in a turn as $P_{turn} = 225\ W/s$, the energy to cover a turn with $\Delta\theta$ angle can be calculated as:

$$E_{turn} = P_{turn}\frac{\Delta\theta}{\omega_{turn}} \tag{2.9}$$

2.4.2 Wind Disturbances

Wind is the most significant environmental factor that can affect the UAVs, including the speed and direction of the wind. Although the wind can incur resistance to the movement of drones, it can benefit the energy consumption of the drone in some cases as well. Hence, considering the wind in the energy consumption of the drone is crucially significant.

There are many path planning methods that consider the energy consumption of the drone in the path planning [47–50], however they do not consider the effect of wind in their energy calculation.

In most of the work like in [51–53], the authors usually compensate the wind in the control loop design of the UAVs to keep the position error of flight path as minimum as possible with respect to the desired path. However, we are compensating for the wind disturbances during the path computation, before the flight. This method helps us to be able to explore the effect of the wind in energy consumption of the drone. Accordingly, we are able to precisely calculate the energy consumption of the drone and plan the most appropriate path for it. We calculate the near-optimal path offline, considering the expected wind. The drone internal flight controller compensates in real time for the actual wind conditions experienced while following the path computed according to the expected wind.

In the following section, we describe the way we compensated the wind effect offline, to consider its effect in the energy consumption of the drone.

Offline Wind Compensation

As we explained before, a lot of different methods have been presented to compensate for the wind recently, and wind is generally compensated online and in the control loop design of the drones to keep the minimum deviation from the desired path. Despite that, we compensate for the wind in our algorithms offline and during the

path generation. As a result, the wind effect is considered in energy consumption of the drone. Our path planner considers the wind speed, direction and energy budget of the UAV and then plans the most suitable path for that specific situation.

Algorithm 3 Offline wind compensation

function GROUNDDISTANCE(u, v):
    return $\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$

function FLIGHTDISTANCE(u, v):
    input: v_d drone speed
    input: $\vec{v}_w$ wind speed vector
    if $x_v > x_u$ **then** ▷ θ declares the angle of the drone direction w.r.t x axis
        $\theta = \arctan(y_v - y_u)/(x_v - x_u)$
    else if $x_v < x_u$ **then**
        $\theta = 180 + \arctan(y_v - y_u)/(x_v - x_u)$
    else
        if $y_v > y_u$ **then**
            $\theta = \arctan(y_v - y_u)/(x_v - x_u)$
        else
            $\theta = -90$
    if $v_w > 0$ **then**
        γ =wind_direction $-\theta$ ▷ γ represents the angle between drone direction and wind direction
    else
        $\gamma = 180 +$ wind_direction $-\theta$
    // solve the system of equations (3.2)-(3.6)
    $v_{wx} = |v_w| \times \cos\gamma$
    $v_{wy} = |v_w| \times \sin\gamma$
    $v_{dy} = -v_{wy}$
    $v_{dx} = \sqrt{v_d^2 - v_{dy}^2}$
    $v_{gx} = v_{dx} + v_{wx}$
    $v_{gy} = v_{dy} + v_{wy}$
    $t_{uv} = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}/\sqrt{v_{gx}^2 + v_{gy}^2}$
    FLIGHTDISTANCE(u, v) = $t_{uv} \times v_d$
    return $t_{uv} \times v_d$

Algorithm 3 represents the function that we will utilize in our offline path planner algorithms to compensate for the wind.

Considering the wind disturbances and the turning angle in our offline path planning algorithms enables us to calculate the energy consumption of the drone very precisely. Consequently, the estimated energy consumption for a mission is

much closer to the real consumption and leads us to implement our algorithms in real-world path planners effortlessly (as we will show our implementation in the following chapters).

2.5 LTE Network for Unmanned Aerial Vehicles

The users of smart phones and Internet of Things (IoT) make the most of using cellular network coverage which is providing high capacity and extensive area wireless data services. It is highly interesting to employ cellular networks to provide support for UAVs enhancing safety in operations, control, and communication quality of service. Furthermore, cellular networks enable the UAVs to operate beyond visual range securely. Consequently, utilizing cellular networks attracted the attention of industry and companies [1], to provide cellular connectivity for UAVs. For instance, telecommunications leading companies like Qualcomm and AT&T have set up to develop the connectivity technologies, including optimization of LTE networks and advancement of 5G technology for drones [2]. The 3rd Generation Partnership Project (3GPP) is started the study on enhanced LTE support for UAVs in 2017 [3]. Terra Drone global UAV company and KDDI telecommunications operator jointly released a "4G LTE control system" that enables the users to control UAVs via 4G LTE network [54].

2.5.1 Cellular Networks Assisting UAVs

There are several reasons to prove that much more advanced communication is required for modern UAVs. We announce some of them as following:

- High speed digital data transmission in order to provide a high quality video streaming and supporting modern autopilots. Data rate requirement for video streaming defined by 3GPP [1] is up to 50 Mb/s.
- Beyond visual range operations and multi-UAV operations that make the point to point communication obsolete.
- Providing the required safety and security for the UAVs in operation that depends on a reliable communication.

- 3GPP requirements to ensure a proper command and control are data rates up to 100 kb/s and packet error rate less than 1% [1].
- Robust UAV navigation which makes the traditional UAV navigation (like Global Positioning System that is vulnarable to satellite signal disturbances as a result of bad weather conditions or tall buildings blockage) ineffective.

As a result, cellular network technology has the features that are noticed above in order to provide a proper communication for the low altitude (< 120 m) UAVs. Some of the strong points of cellular technology are as following:

- The infrastructures are already installed and are ready-to-use. Thus UAVs can reuse all the cellular antennas which are already deployed all around the world.
- Seamless handover across the whole network coverage.
- High capacity to support a numerous number of UAVs.
- high potential in developing and setting up new technologies to improve the performance of the system.
- Very reliable and providing a good quality of service.
- Highly secure communication (for instance, estimating to confront tampering with communications and eavesdropping).

In the next sections, we investigate the challenges that UAVs are facing in using the cellular network. Furthermore, we discuss the requirements for UAV communications and the way cellular networks may provide these requirements.

2.5.2 Challenges in Serving UAVs via LTE Network

Here we notice two of the most important challenges that we are facing in utilizing the LTE networks to serve the low altitude UAVs [55]. Network coverage is one of the main challenges. Originally, Cellular networks are designed and optimized for terrestrial wide-band communication. Therefore, the cellular network antennas are down tilted in order to prevent energy from propagating into other cells [56]. Since

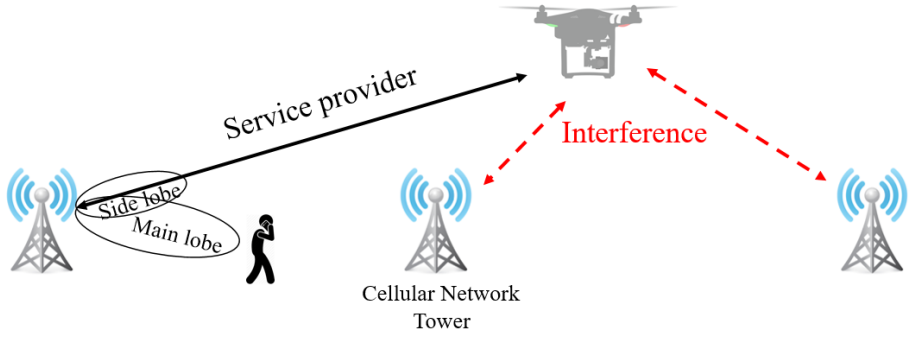

Fig. 2.10 Low altitude UAVs connectivity with terrestrial mobile networks.

the antennas are down tilted, it is more probable that the low altitude UAVs be served by the side lobes of antennas which are farther. In Fig. 2.10, the scenario is shown in detail. In fact the UAVs are more interested in sky propagation of antennas than the terrestrial propagation.

Interference is the other main challenge. Since propagation in the sky is more favorable for the low altitude UAVs, they might produce up-link interference for the cellular cells in the vicinity and experience down-link interference from the same cells. Furthermore, by increasing the number of UAVs and consequently the UAV connections, the up-link interference (if not managed properly) may lead to performance degradation for the user equipments on the ground. Solving the interference issues is one of the main tasks of the 3GPP study [57] to improve the LTE support for the UAVs.

The features of interference and coverage for the cellular connected UAVs are completely different from terrestrial one. In the next sections we explore the feasibility and potential enhancements in order to improve the efficiency of the connection for the low altitude UAVs without effecting the performance of user equipments on the ground.

2.5.3 Communication Requirements

Cellular network enabled UAVs have considerably different communication and spectrum requirements comparing with terrestrial cellular communication [58]. The communication between the UAVs and ground can be classified into two aspects: *command and control* (C&C) and *payload* communication. Command and control

refers to a safe control and reliable operation of the UAV from the ground control station side. Some of the C&C messages are as following:

- Remote control
- Status monitoring (velocity, altitude)
- Formation flying
- Regulatory compliance
  - Identification
  - Path planning and collision avoidance
  - Location service

Fulfilling the C&C requirements is very vital for a successful UAV operation. Thus, stringent requirements are essential and the cellular network must address these requirements.

C&C Cellular Support

Obviously, the ability to control the vehicle position and orientation is vital for the UAVs. For non-autonomous UAVs, a pilot controls the vehicle from a remote base station. For autonomous UAVs, that have a predefined flight plan, the possibility of interrupting the predefined plan by a pilot and take the control of vehicle in real time need to be considered for safety.

A reliable communication interface between the pilot and vehicle is needed to perform these operations. Such interface must fulfill the following requirements:

- Pilot commands to the vehicle and telemetry data to the pilot must be transmitted reliably.
- Real time control of the vehicle requires low latency.
- Serving many UAVs in the same area requires enough capacity.
- Resistance to natural or artificial source interference is necessary.

- Sufficient coverage for UAV communication.
- Operating safely in case of link failure.

Mobile network and Wi-Fi are the best candidates to fulfill the requirements of this control interface. We compare both technologies based on the above requirements in Table 2.3 [9].

Losing the C&C link may leads to catastrophic consequences. Therefore, C&C link of UAVs must operate over protected spectrum based on the International Civil Aviation Organization (ICAO) announcement. Cellular network has such potential and can provide a reliable C&C for UAVs.

Payload communication such as image, and real-time video streaming requires much higher data rate comparing to C&C. Full HD video transmission for instance requires several Mbps. Cellular network has also the potential to recover HD video as well.

2.5.4 Enhancing the Cellular Network Performance

Originally, cellular network is designed for terrestrial communication. Although they are already qualified to support the low-altitude UAVs, and since the number of cellular connected UAVs are increasing, beside focusing on improving the safety and reliability of UAV missions, the cellular industry has initiated to enhance the cellular network performance by doing technical improvements in order to provide better services to UAVs.

In order to enable a productive coexistence between the traditional terrestrial user equipments and the new aerial vehicles, the following factors must be considered [58]:

- *3-D coverage*: Since UAVs have higher altitude comparing with the conventional users on the ground, the cellular antenna height required to be exceeded. However, preliminary measurements by Qualcomm have shown that the current aerial coverage by side lobes of antennas for low altitude UAVs are satisfactory. For scenarios that are along fixed aerial corridors (e.g. pipe checkup), “UAV highway” concept (coverage only along certain fixed aerial corridors) can be utilized.

Table 2.3 Comparing mobile network and Wi-Fi technologies, considering control requirements fulfillment.

Requirement	Mobile Network	Wi-Fi
Transmitting data reliably	Cellular network operates in licensed spectrum and resource allocation is managed. Consequently, it is free from congestion and for special applications is able to give priority to resources to improve reliability.	Wi-Fi provides reliable transmission in uncongested spectrum, while in congested and unlicensed spectrum interrupts in data delivery.
Low latency	Cellular network can manage latency by providing quality of service tools using resource management algorithms.	Wi-Fi manages latency by providing quality of service management tools, but the achieved latency relies on the congestion level in the unlicensed spectrum.
Enough capacity	There is a defined capacity for cellular network in the covered area.	Wi-Fi capacity is usually limited due to the interference from other unlicensed spectrum users in the area.
Immune from interference	Mobile network works in licensed spectrum and is free of interference due to the unmanaged users. However, if communication is degraded due to the interference, a handover to another base station solves the problem.	Multiple users can operate independently and with minimum interference in identical spectrum.
Sufficient coverage	Cellular network is extendable to everywhere (within the covered area of network) by doing a handover.	The Wi-Fi range is limited.
Safe operation in failure	If communication to one base station is lost, a handover to another base station is possible.	If the Wi-Fi link is lost, there is no solution except re-establishing the connection.

Table 2.4 3GPP required improvements and solutions to fulfill UAV requirements.

3GPP Release	Required improvements and solutions
Release 14	1- Introduction of requirements. 2- 5G requirements defined in high level.
Release 15	1- Agreement to study on UAVs . 2- LTE bearers meet the C&C requirements (latency, throughput, reliability, etc). 3- LTE bearers meet the payload (e.g. HD video) requirements (latency, throughput, reliability). 4- Making sure that UAVs do not interrupt ground LTE users. 5- Meeting regulations for UAV communications.
Release 16	1- Making sure that UAVs can use the LTE positioning technologies. 2- Making sure that for vehicle-to-vehicle communications, the proximity services of LTE can be used. 3- Making sure that UAVs identification requirements are match with the LTE supported identities.

- *Unique channel*: High altitude of UAVs usually leads to a strong line-of-sight links between the UAV and base station. This unique channel on one hand creates strong communication channel between the UAV and base station, and on the other hand the line-of-sigh links domination cause the inter cell interference more critical for cellular systems.

- *Aerial-ground interference*: In down-link communication (base station to UAV), UAV may receive interference from other base stations due to the line-of-sight channels. In up-link communication (UAV to base station), UAV may cause interference to the adjacent non-related base stations.

- *Up-link traffic*: The original cellular network is designed for dominant down-link communication. However, the UAV communications with cellular network requires a higher data rate in up-link (like for video streaming) than the down-link, which must be considered in new generations.

In Table 2.4, the solutions and developments in order to address the use of LTE for UAVs in 3GPP releases are shown [59].

Chapter 3

Communication-Aware UAV Path Planning

In this chapter, we present our novel communication-aware path planning algorithms which are applicable for UAVs. First, we compare our work with the-state-of-the-art works, explaining the lacks and shortages of other works, and the advantages of our work. Then we address the problem of generating the near-optimal path from one point to another for a UAV and the challenges that we are dealing with. We describe the model of the flight area that our algorithms are working on it. Moreover, we describe the drone mobility model, the energy constraints, and the mobility planning of drone between each two nodes. After that, we introduce our path planning algorithms which are the variants of classical A* algorithm. The AT-PP algorithm that maximizes the average throughput along the path and the MT-PP algorithm that maximizes the minimum throughput along the path subject to the budget of energy. We explain the algorithms and their performance in detail providing the pseudo-code of the algorithms. Finally, we investigate the methods to smooth the resulting paths and introduce our novel path smoothing method called A*IPS.

This chapter has previously been published as: Afshin Mardani, Marcello Chiaberge, and Paolo Giaccone. Communication- aware uav path planning. In *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)* [60].

3.1 Related Work

Up to now, various path-planning methods and algorithms for UAVs have been developed [61], but they are entirely unaware of the cellular network coverage. The algorithm in [62] is a heuristic search based on A*, which iteratively optimize the path during the available search time. Lazy Theta*, proposed in [63], is an any-angle path finding algorithm in which the resulting paths are not constrained to the form of graph edges and is faster than Theta* algorithm [64]. In [65], a heuristic-based re-planning algorithm (AD*) is presented that continuously improves its solution while time allows, and corrects its solution when updated information is received. In [66], inspiring by A* algorithm and Dubins path, a path planning method to discover the shortest, flyable and safe path for fixed wing UAVs with obstacle escape is presented. In [67], the authors discussed any-angle path planning algorithms that are variants of classical A* algorithm. In their method, through propagating information along grid edges, short paths are found without constraining the resulting paths to grid edges. In [68], a path planning method to create trajectories for aerial vehicles in a two dimensional space is proposed in order to avoid risky areas, conflict, and no-fly zones. In sampling based algorithms like Rapidly-exploring Random Trees (RRT) that is introduced in [24], the algorithm is able to find a sub-optimal path in a high dimensional space while some pre-defined information about the robot operating space is required. The RRT has no optimization process and re-planning procedure. Therefore, an improved version RRT* [69] is proposed to bridge this gap.

Recently, a lot of research has been done to optimize the path for UAV communication systems for different setups. In [70], the authors used an aerial wireless coverage 3-D modeling in mission planning of aerial vehicles for monitoring and surveillance of vast areas like long linear utility infrastructures. In [71], the flying trajectory of UAV is optimized for up-link communications. In [72], the authors used a UAV-based mobile relay in order to send data to different group of users. They optimized the data volume and relay trajectory based on the visiting sequence to the different group of users. In [73] and [74], the UAVs' movement is optimized to improve the network connection of a UAV assisted ad-hoc network. In [75], a multi-antenna UAV flying over a collection of single-antenna mobile ground nodes for providing relay services for mobile *ad hoc* networks is considered. They optimized the heading of UAV in order to optimize the up-link communication performance. In [76], they investigated a communication system with multi-antenna UAVs as

relays between ground-based terminals and a network base station. They developed a closed form expression to optimize the UAV heading based on knowledge of the user terminal's future position, in order to maximize the uplink data rate. They attempted to keep the rate of each individual link above a certain threshold. In [77], the authors employed the UAVs as aerial base stations as a promising solution to enhance the performance of existing cellular systems. They exploit the UAV's high mobility to serve a group of users on the ground in order to maximize the minimum average rate among all the users, by jointly optimizing the trajectory of the UAVs, controlling the transmit power, and scheduling the user association and communication. In [78], a UAV-enabled orthogonal frequency division multiple access (OFDMA) system is studied, where the UAV is dispatched as a mobile base station to serve a group of users on the ground. The minimum average throughput of all ground users is maximized while meeting a given set of constraints, by jointly optimizing the UAV trajectory and OFDMA resource allocation. In [79], a comprehensive framework for hyper-dense small-cell networks assisted by caching UAVs is proposed, its feasibility is analyzed, and the secure transmission is discussed. UAVs are utilized to provide data traffic to mobile users cooperatively with small-cell base stations, due to their lower cost and higher mobility. The work in [80] studies a hybrid cellular network with UAV-aided offloading at the edges of multiple cells, by accounting for the interference between ground base stations and UAV. The UAV trajectory is selected to maximize the sum rate of edge users by avoiding the interference, while the rate requirements of all the users are guaranteed.

However, in none of the above works the energy consumption of the UAVs and the effect of wind on the energy consumption has been considered.

In general, UAV communication systems may encounter various new challenges [81]. In fact, the performance of UAV communication networks are highly confined by on-board energy. Thus, maximizing the information bits while minimizing the energy consumption in the same time is of paramount importance. Furthermore, minimizing the energy consumption of UAV systems regarding the energy expenditure in propulsion power consumption to support the movements and maintaining the UAV in the sky, is more important than reducing the energy consumption in communication circuits and signal transmission.

Moreover, in many researches trajectory optimization has been studied, but not particularly for communication purposes. Many algorithms have been developed on

energy-aware UAV path planning. For instance in [46], path optimization problem is studied considering the energy consumption of UAV, but the communication performance is not covered. In [82], the task of seeking out the goal while considering the total energy consumption of UAV to be minimized is studied. This optimization depends on unknown disturbances, like wind. In this study, the communication is neglected as well.

In [83], the authors represented a multi-layer architecture for intelligent navigation of Remotely Piloted Aircraft Systems (RPAS) in urban environments. In the path planning layer, the planner computes an optimal way point-based path, then the on-line planner continuously updates the offline path. Their algorithm searches for an optimal path that minimizes the cost function. It minimizes the risk-cost, the estimated energy consumption, and the length of the path. Another example is [84] which proposes a shortest trajectory planning for UAVs on an embedded GPU. A fast, energy-efficient global planner for multi-rotor UAVs has been developed supporting human operator during rescue mission. The planner is suitable for real-time path re-computation in dynamically varying environments but of small area (no more than 200 m^2). The work in [85] proposes a multi-objective path finder that can discover Pareto-optimal solutions concerning energy consumption and length of the path. Their solution is on the basis of NAMOA* search algorithm that exploits a monotone heuristic cost function. Unfortunately, in all of these works the effect of network coverage is completely neglected.

Several recent works have targeted the path planning for UAVs considering the energy efficiency and network yield, simultaneously. But the absence of wind as a crucial factor on the UAV's energy consumption and trajectory planning is sensible. In [47], the energy-efficient designs for drone communication is studied, where a drone is exploited to communicate with the base station. In this work, the energy efficiency maximization via path optimization is performed. The work in [48] proposed a computationally efficient suboptimal algorithm that can saves energy by 50 percent, increases network throughput by 15 percent, and extend network lifetime by 33 percent compared to the-state-of-the-art. In [49], the offline path finding problem for UAVs is addressed. The authors in this work attempted to discover paths that meet mission objectives, are safe considering collision and grounding, are fuel efficient, and satisfy criteria for communication. In [50], they proposed a joint UAV trajectory and power control scheme that significantly enhances the achievable rate of UAV communication system comparing to benchmark schemes. Particularly,

they maximized the average achievable rate from the UAV to the ground receiver over a finite communication period by jointly optimizing the UAV trajectory and transmit power allocation. These are subject to maximum speed of UAV, initial/final locations, and average transmit power constraints. Nevertheless, in all of the above cited works, the wind effects on the energy consumption is neglected.

A lot of research has been done on following certain trajectories by UAVs under some circumstances. They mostly investigate the effect of wind on following a generated path accurately in order to accomplish the assigned tasks. This effect can be considered in the control loop design of the UAVs to keep the position error of flight path under a determined error with respect to the desired path. Therefore, the wind disturbances are not considered in the path generation. The work in [86] considered the effect of wind that causes the aircraft to drift in a certain direction. A method is designed based on an accelerated A* algorithm that follows the trajectory planner to take into account the wind effects. In [87], UAV path following in cluttered environments under windy conditions in a two dimensional configuration space is investigated. They designed a novel guidance law with low computational complexity which allows the UAV to follow the path with minimum deviation. In [88], the authors presented an algorithm based on the idea of following a vector field that converges smoothly to the desired path. Their work includes the technique of dealing with wind disturbances when following a generic sufficiently smooth 2-D path.

There is not a unique approach to compensate for wind effects in trajectory planning for UAVs. For instance, in [39], the authors dealt with the wind disturbances on trajectory planning by iteratively solving a no wind case problem with a moving virtual target. In most of the work like in [51–53], the authors usually compensate the wind in the control loop design of the UAVs to keep the position error of flight path as minimum as possible with respect to the desired path. However, we are compensating for the wind disturbances during the path computation, before the flight. This method helps us to be able to explore the effect of the wind in energy consumption of the drone. Accordingly, we are able to precisely calculate the energy consumption of the drone and plan the most appropriate path for it. We calculate the near-optimal path offline, considering the *expected* wind. The drone internal flight controller compensates in real time for the actual wind conditions experienced while following the path computed according to the expected wind.

3.2 The Path Planning Problem

In this work, the problem of generating the near-optimal path from an initial waypoint to another one in a 2-D space for an autonomous UAV is addressed. We maximize the quality of the video streaming application during the flight in our innovative communication-aware approach by considering the accessible energy on-board and the expected wind conditions. Noteworthily, we compute the trajectory offline and upload it to the drone flight controller. Then, it follows the offline path and compensates for the actual disturbances during the flight.

3.2.1 Flight area

We modelled the flight area, as shown in Fig. 3.1, based on a grid graph. Indeed, we subdivided the cellular coverage area into parts with a regular tessellation and placed each node at the center of each square. We associated each node with a throughput value which is an average throughput value experienced within the corresponding square. An edge connects the nodes corresponding to adjacent squares. In our study, 8-degree grid graph is considered, i.e. eight neighbors for every node. It is notable that our approach is extendable to any other type of grid graph.

Formally, we define the flight area with an undirected grid graph $G = (V, E)$. We associate node $i \in V$ with a physical position (x_i, y_i) and the throughput b_i. b_i is the drone experienced throughput in the area in the proximity of node i when uploading the streaming data to the cellular network. The physical distance d_{ij} is associated with the edge $(i, j) \in E$ connecting nodes i to j.

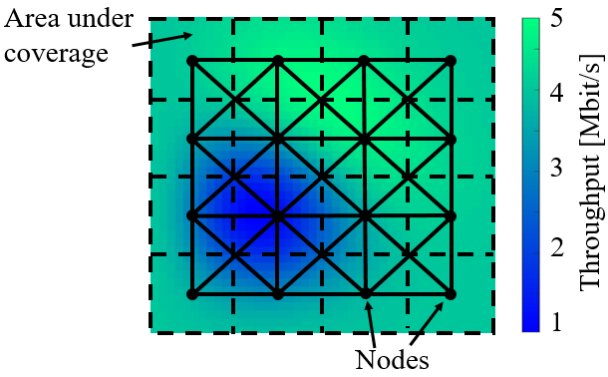

Fig. 3.1 8-degree grid graph and the corresponding coverage map.

This work is focusing on the path planning for the UAVs. We are concentrating on how to design a path for a mission that the drone experience the maximum throughput while the throughput map is given. Therefore, in this work, the average throughput experienced in each square is assumed to be known in advance. This information can be obtained through some coverage maps obtained by means of on-field measurements or on some channel model applied into the square area.

For the sake of simplicity, obstacles are not considered in the flight area, even though we can easily extend the methodology to this scenario by removing the corresponding nodes and edges.

3.2.2 Drone Mobility Model

The drone is assumed to fly at constant *air speed* v_d, and its power consumption is equal to P. The drone's total available on-board energy is E_0. The required energy due to traveling from node i to node j is E_{ij}. The *source node* i_s is the node that the drone departs from as the starting way-point, and the *destination node* i_d is the node that the drone terminates the mission as the destination way-point. A path comprised of multiple way-points decomposes into several segments, and the algorithm runs for each segment.

Let $\mathscr{P}$ be the set of all possible loop-less paths connecting i_s to i_d and let $p \in \mathscr{P}$ be a generic path. The path planning aim is finding a path $p \in \mathscr{P}$ that connects i_s to i_d such that maximizing the communication performance, contingent on the following amount of energy:

$$\sum_{(i,j)\in p} E_{ij} \leq E_0 \tag{3.1}$$

3.2.3 Mobility Planning Between Two Nodes

Considering the problem in which the drone needs to move from way-point i to j. Let v_{dx} be the drone *air speed*, along the x-axis and v_{dy} be the drone *air speed*, along the y-axis. Let v_{gx} and v_{gy} be the drone *ground speed*, along the x-axis and y-axis, respectively. Let t_{ij} be the flight time from i to j. The *wind speed* is assumed to be constant and its speed along the two axes are v_{wx} and v_{wy}.

Table 3.1 Variable definitions.

Parameter	Description	Unit
b_i	Cellular network throughput at node i	bit/s
d_{ij}	Physical distance	m
E_{ij}	Energy consumed between two nodes	J
E_0	Total energy of UAV	J
t_{ij}	Time of flight	s
P	Power consumption	W
v_d	Air speed of UAV	m/s
v_g	Ground speed of UAV	m/s
v_w	Wind speed	m/s

Now the following system of equations can be written:

$$v_{gx} = v_{dx} + v_{wx} \tag{3.2}$$

$$v_{gy} = v_{dy} + v_{wy} \tag{3.3}$$

$$x_j - x_i = v_{gx} \cdot t_{ij} \tag{3.4}$$

$$y_j - y_i = v_{gy} \cdot t_{ij} \tag{3.5}$$

$$v_{dx}^2 + v_{dy}^2 = v_d^2 \tag{3.6}$$

where the effect of wind and the actual ground speed are related by (3.2) and (3.3); the traveled physical distance is related to the ground speed by (3.4) and (3.5); eventually, the air speed of drone is related to its two components by (3.6). The drone speed and its flight time are calculable by solving the above equations. Therefore, the flight distance travelled by the drone is $v_d \cdot t_{ij}$, and the energy consumption between node i and j is:

$$E_{ij} = P \cdot t_{ij} \tag{3.7}$$

The ground distance from node i to j is equal to the Euclidean distance between them. The total transferred data along the path from node i to j is $b_i \cdot t_{ij}$. Table 3.1 contains the introduced parameters and their descriptions altogether.

Considering the energy model employed in [46], the energy consumption of the drone depends on different operating conditions, such as speed, horizontal and vertical acceleration, and turning angle. We do not consider the consumed energy in climbing and descending, since it is common in all path plannings and does not affect the path. Moreover, we do not have any hovering period in our

applications. Therefore, we consider the energy consumption during the rotations based on the work in [46]. The energy required to cover all the n turns (with angles $\theta_k, k \in \{1,2,...,n\}$) included in the path is computed as:

$$E_{\text{turns}} = \sum_{k=1}^{n} P_{\text{turn}} \frac{\theta_k}{\omega_{\text{turn}}} \tag{3.8}$$

where ω_{turn} represents the angular rotation speed, P_{turn} represents the power consumed during the rotations. Therefore the total consumed energy is computed considering also all the energy spent for all the turns along the path:

$$E_{tot} = E_{i_s i_d} + E_{\text{turns}} \tag{3.9}$$

3.3 Near-Optimal Communication-aware Path Planning

Our propounded approaches approximate the optimal path founded on the classical A* search algorithm for path planning, adopted in many contexts as robotics and video games. A* was designed for finding the shortest (or approximately shortest) path in a much shorter computation time than canonical Dijkstra's algorithm, by considering a smaller search subspace. Actually, A* employs a cost function which is the summation of two functions. Function g (in common with Dijkstra's algorithm) represents the cost of the path from the source to the current node, and function h is a user provided heuristic function that estimates the cost of the path from the current node to the goal. Computation time and optimality is conditional on the choice of heuristic function. Particularly, if the estimated cost by the heuristic function is equal to the real cost, the algorithm only expands the nodes on the least cost path from the start to the goal.

3.3.1 Path Planning Algorithms

In this section, we present our path planning algorithms as variants of the A* algorithm.

Algorithm 4 Average-Throughput Path Planning (AT-PP)

1: **function** AT-PP($\mathcal{N}, \{b_v\}_{v\in\mathcal{N}}, S, D, P, v_d, E_0, \beta$)
2: **for each** vertex $v \in \mathcal{N}$ **do** ▷ Initialization, for each vertex
3: path_bw[v] = -1 ▷ Throughput from S to v
4: acc_path_bw[v] = -1 ▷ Cumulative throughput from S to v
5: parent[v] = -1 ▷ Parent of node v
6: ground_path_distance[v] = ∞ ▷ Ground distance from S to v
7: flight_path_distance[v] = ∞ ▷ Flight distance from S to v
8: path_hops[v] = -1 ▷ Number of nodes from S to v
9: path_time[v] = -1 ▷ Flight time from S to v
10: path_energy[v] = ∞ ▷ Energy from S to v
11: path_cost[v] = ∞ ▷ Cost based on A*
12: path_bw[S] = acc_path_bw[S] = b_S ▷ Setting the values for S
13: parent[S] = S
14: ground_path_distance[S] = flight_path_distance[S] = path_hops[S] = 0
15: path_time[S] = path_energy[S] = 0
16: path_cost[S] = GROUNDDISTANCE(S, D)
17: $\mathscr{U} = \mathcal{N} \setminus \{S\}$ ▷ Unvisited nodes
18: $\mathscr{F} = \{S\}$ ▷ Frontier nodes
19: $\mathscr{V} = \emptyset$ ▷ Visited nodes
20: **while** $\mathscr{F}$ is not empty **do** ▷ Visit all the frontier nodes
21: $u = \arg\min_{v\in\mathscr{F}} \{\text{path\_cost}[v]\}$ ▷ Find the min cost node in $\mathscr{F}$
22: move u from $\mathscr{F}$ to $\mathscr{V}$
23: **if** path_energy[u]> E_0 **then** ▷ Check the required energy to reach u
24: **continue** ▷ If greater than E_0, consider a new node in $\mathscr{F}$
25: **if** $u = D$ **then** ▷ Check if arrived to destination
26: **return** ▷ End. Return the whole state, from line 4 to 11
27: **for each** neighbor $v \notin \mathscr{V}$ of u **do**▷ Check all the neighbors of u that are in $\mathscr{U}$ or $\mathscr{F}$
28: **if** $v \in \mathscr{U}$ **then**
29: move v from $\mathscr{U}$ to $\mathscr{F}$ ▷ Move v to the frontier, if is not there
30: bw = (acc_path_bw[u]+b_v) / (path_hops[u]+1) ▷ Average throughput along the path
31: path_bw[v] = bw
32: acc_path_bw[v] = acc_path_bw[u] + b_v
33: parent[v] = u
34: ground_path_distance[v] = ground_path_distance[u] + GROUNDDISTANCE(u, v)
35: flight_path_distance[v] = flight_path_distance[u] + FLIGHTDISTANCE(u, v, v_w, w_{dir})
36: path_hops[v] = path_hops[u] + 1
37: path_time[v]=path_time[u]+FLIGHTDISTANCE(u, v, v_w, w_{dir})/v_d
38: path_energy[v] = path_energy[u] + $P \times t_{uv}$
39: path_cost[v]=(flight_path_distance[u]+FLIGHTDISTANCE (u, v, v_w, w_{dir})) + FLIGHTDISTANCE(v, D, v_w, w_{dir}) + β/path_bw[v] ▷ Main cost function
40: **return** error - unreachable destination

AT-PP Algorithm

This algorithm is a modified version of A* on grids. We provided the pseudo-code of the algorithm in the Algorithm 4. For the sake of readability, degenerate cases are neglected. In this algorithm the average throughput along the path is maximized, as:

$$\max_{p \in \mathscr{P}} \sum_{(i,j) \in p} \frac{b_i}{|p|} \tag{3.10}$$

contingent on the budget of energy. Accordingly, we modified the cost function of A* as is shown in ln. 39. Our cost function combines the estimated distance to the final destination D and the inverse of the average throughput ("path_bw") up to some node. Therefore, by increasing the average throughput, an average level of communication QoS is guaranteed.

AT-PP algorithm maintains several values for every vertex v (ln. 3-11):

- $path\_bw(v)$ is the throughput from the start node to node v found so far. It is used in the cost estimation of node v in the cost function.
- $acc\_path\_bw(v)$ is the cumulative throughput from the start node to node v, and it is utilized for average throughput calculation.
- $parent(v)$ allows to retrieve the resulting path when the algorithm terminates.
- $ground\_path\_distance(v)$ contains the ground distance from the start node to node v.
- $flight\_path\_distance(v)$ contains the flight distance from the start node to node v considering the effect of wind. This value is used in the estimate of the cost of node v in the cost function.
- $path\_hops(v)$ contains the number of nodes from start node to node v.
- $path\_time(v)$ is the flight time from the start node to node v, considering the wind.
- $path\_energy(v)$ is the flight energy consumption from the start node to node v, considering the wind.
- $path\_cost(v)$ is the cost of the path from start node to node v.

As in usual Dijkstra and A*, AT-PP algorithm creates three sets of nodes. The frontier nodes ($\mathscr{F}$) are stored in a priority queue and the visit procedure expands from them. The visited nodes ($\mathscr{V}$) are the nodes that have been already visited and will not be visited anymore. The unvisited nodes ($\mathscr{U}$) are the remaining nodes, i.e. not frontier nodes and not yet visited. As long as the frontier nodes list is not empty (ln. 20) (if it is empty, it announces that the destination is unreachable (ln. 40)), the algorithm selects the node with minimum path cost (ln. 21) from the frontier nodes as the current visiting node. First, it moves the current visiting node from frontier nodes to visited nodes (ln. 22), then it checks the required energy to reach the current visiting node (ln. 23). In case of exceeding the energy budget, the algorithm fetches a new node from the frontier nodes (ln. 22). Otherwise, in the next step, it checks if the current visiting node is the destination or not (ln. 25). If yes, the algorithm extracts the obtained path and terminates. Otherwise, it updates all the neighbors of the current visiting node which are not already visited (ln. 27). It checks each neighbor if it is in frontier nodes or not (ln. 28). If that neighbor is not included in frontier nodes, it will be moved there (ln. 29), and all the values will be updated (ln. 31-38). In this algorithm, throughput is the average along the path (ln. 30). Finally, the path cost will be updated in (ln. 39). This procedure continues until the algorithm reaches to the destination.

The fundamental difference in our algorithm from A* is that our algorithm does not examine whether the value of the g function of visiting node plus the length of the straight line from the visiting node to the next neighbor node is smaller than the value of the g function of the next neighbor node. In our algorithm, distance is considered in the cost function plus the inverse of the bandwidth of the next neighbor. Afterwards, the neighbor of the visiting node with minimum cost will be sent to the frontier nodes list of the algorithm. Therefore the node with minimum cost from the frontier nodes list will be the next visiting node. Then, the algorithm searches for the path with minimum cost. Consequently, both the throughput and the distance are optimized in the resulting path.

In the cost function, the bandwidth is weighted by a factor β. This coefficient is tuned to be a comparable value with the other part of the cost function. It will be shown in Chapter 5, Sec. 5.3, β is numerically tuned to maximize the throughput in different scenarios for a specific coverage map.

The available on-board energy E_0 is taken into account by pruning the visit as long as the energy consumption overshoots E_0 (see ln. 23). Notably, it is probable not to find any path due to the incompatibility of the energy consumption and E_0 (see ln. 40).

MT-PP Algorithm

In this algorithm the minimum throughput along the path is maximized, contingent on the budget of energy. We modified the cost function of A* and defined the worst-case throughput cost function for the algorithm as following:

$$\max_{p \in \mathscr{P}} \min_{(i,j) \in p} b_i \tag{3.11}$$

Therefore, by keeping the throughput above a minimum level, a minimum level of communication QoS is guaranteed.

We provided the pseudo-code of the algorithm in the Algorithm 5. This is an iterative algorithm and functions as follows, mimicking a dichotomic search. At the beginning, the algorithm finds the shortest path while the minimum throughput is maximized. In this situation, if the calculated energy consumption is less than the available on-board energy, the resulting path is the best and final path. On the contrary, if the calculated energy consumption for the resulting path is greater than the energy budget of UAV, an initial threshold value t_h is assumed by the algorithm. t_h could be set equal to the average throughput achievable in the area. In this part, the algorithm maintains three variables: *throughput level* (b_{level}), which contains the highest throughput value that the algorithm could obtain in the first stage for the resulting path, *throughput step* (b_{step}), which contains the decrement value in each iteration, and *throughput resolution* (b_{res}), which determines the final resolution for the solution. At the beginning, b_{step} is set equal to half of the maximum achievable throughput in the coverage map. On the basis of a variant of AT-PP, the algorithm selects the path such that the throughput at some visited node is computed as the minimum along the path (unalike to ln. 30 of AT-PP).

On the termination of each iteration, t_h increases in order to maximize the minimum throughput. The algorithm may not find a path with a minimum throughput $\geq t_h$ (possibly due to the energy restriction), then it decreases t_h in the following

Algorithm 5 Minimum-Throughput Path Planning (MT-PP)

1: **function** DRONEPATHPLANMT-PP()
2: $(Graph, S, D, P, b_n, v_d, (x_n, y_n), E_0, \beta, \mathcal{N}, b_{step}, b_{res}, b_{level})$:
3: PATHMT-PP() ▷ Compute the path
4: **while** $b_{step} >= b_{res}$ **do** ▷ As steps are greater than resolution
5: **if** path_energy $> E_0$ **then**
6: $b_{step} = b_{step}/2$ ▷ Each step will be half of previous step
7: $b_{level} = b_{level} - b_{step}$ ▷ Throughput decreases by one step
8: **if** $b_{level} < b_{res}$ **then**
9: $b_{level} = \min\{b_n\}$ ▷ Throughput is minimum
10: PATHMT-PP()
11: **else if** path_energy $< E_0$ and $b_{level} < \min\{b_S, b_D\}$ **then**
12: $b_{step} = b_{step}/2$
13: $b_{level} = b_{level} + b_{step}$ ▷ Throughput increases by one step
14: PATHMT-PP()
15: **else**
16: **break**
17: **function** PATHMT-PP()
18: **for each** vertex $v \in \mathcal{N}$ **do** ▷ Initialization, for each vertex
19: path_bw[v] = -1 ▷ Throughput from S to v
20: acc_path_bw[v] = -1 ▷ Cumulative throughput from S to v
21: parent[v] = -1 ▷ Parent of node v
22: ground_path_distance[v] = ∞ ▷ Ground distance from S to v
23: flight_path_distance[v] = ∞ ▷ Flight distance from S to v
24: path_hops[v] = -1 ▷ Number of nodes from S to v
25: path_time[v] = -1 ▷ Flight time from S to v
26: path_energy[v] = ∞ ▷ Energy from S to v
27: path_cost[v] = ∞ ▷ Cost based on A*
28: path_bw[S] = acc_path_bw[S] = b_S ▷ Setting the values for S
29: parent[S] = S
30: ground_path_distance[S] = flight_path_distance[S] = path_hops[S] = 0
31: path_time[S] = path_energy[S] = 0
32: path_cost[S] = GROUNDDISTANCE(S, D)
33: $\mathcal{U} = \mathcal{N} \setminus \{S\}$ ▷ Unvisited nodes
34: $\mathcal{F} = \{S\}$ ▷ Frontier nodes
35: $\mathcal{V} = \emptyset$ ▷ Visited nodes
36: // Visit all the frontier nodes
37: **while** $\mathcal{F}$ is not empty **do**
38: $u = \arg\min_{v \in \mathcal{F}} \{\text{path\_cost}[v]\}$ ▷ find the min cost node in $\mathcal{F}$
39: move u from $\mathcal{F}$ to $\mathcal{V}$
40: **if** $u = D$ **then** ▷ Check if arrived to D
41: **return** All parameters from line 4 to 11
42: // Check all neighbors
43: **for each** neighbor $v \notin \mathcal{V}$ of u **do** ▷ Check all the neighbors of u that belongs to $\mathcal{U}$ or $\mathcal{F}$
44: **if** $v \in \mathcal{U}$ **then**
45: move v from $\mathcal{U}$ to $\mathcal{F}$ ▷ Enter v to the frontier, if its not
46: **if** (flight_path_distance[u] + FLIGHTDISTANCE(u, v) < flight_path_distance[v]) **then** ▷ Update, in case of better flight distance
47: bw = min{path_bw[u],b_v} ▷ Throughput is min along the path
48: path_bw[v] = bw
49: parent[v] = u
50: ground_path_distance[v] = ground_path_distance[u] + GROUNDDISTANCE(u, v)
51: flight_path_distance[v] = flight_path_distance[u] + FLIGHTDISTANCE(u, v)
52: path_time[v] = path_time[u] + FLIGHTDISTANCE(u, v)/v_d
53: path_energy[v] = path_energy[u] + $P \times t_{uv}$
54: path_cost[v] = (flight_path_distance[u] + FLIGHTDISTANCE (u, v)) + FLIGHTDISTANCE(v, D) + β/path_bw[v] ▷ Cost function
55: **return** error - unreachable destination

iteration. A dichotomic search is adopted in order to optimize the selection procedure of t_h values. The process terminates as long as the algorithm achieves a given precision on the minimum throughput along the path.

The obtained path approximates the one with the maximum throughput possible compatible with the available energy. However, when the resulting path corresponds to the minimum distance path, and the calculated energy consumption is still not compatible with the available energy, then there does not exist any path for this mission that meets the energy budget.

3.3.2 Path Smoothing

The MT-PP and AT-PP resulting paths are constrained to grid edges since their headings are actually restricted to be along the edges of the grid graph. Consequently, the resulting paths are longer than the shortest path on the free 2-D space. This shortcoming led to smooth the resulting paths, by employing a post-smoothing process leading to an increase in the run-time.

Post-Smoothing method

We take into account the A* post-smoothing algorithm (A*PS) presented in [89]. The smoothing procedure starts from the first node of the obtained path by A*. It checks if the current node has line of sight to the successor of its successor in the path or not. We say node u and u' have line-of-sight (LOS) if and only if the straight line from u to u' neither passes through the low throughput (less than a specified minimum network throughput) nodes nor passes between low throughput nodes that share an edge. If so, the algorithm removes the intermediate node and connects the current node to the successor of its successor. A*PS continues this procedure until the current node does not have a LOS to the successor of its successor. The resulting paths of A*PS are usually shorter than A* on grids. In this method, implemented in our approach, the well-know line-drawing algorithm of Bresenham in computer graphics [90] is used to check LOS between two nodes [91]. Notably, we tailored specifically the *Grid* function to consider only nodes with a throughput larger than a minimum specified value.

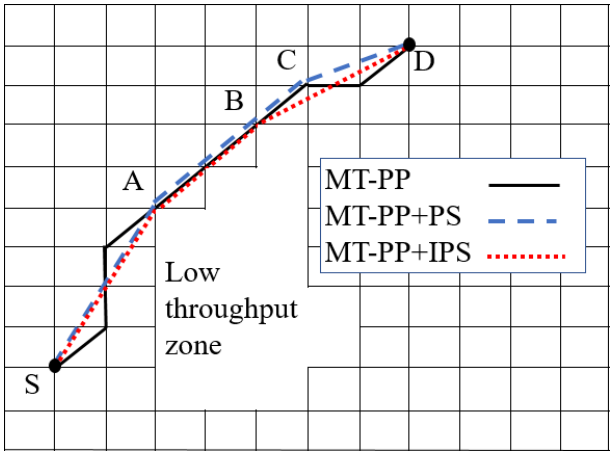

Fig. 3.2 The difference between A*IPS vs A*PS shown in an example.

Improved Post-Smoothing method

In this section, we introduce an innovative improved post-smoothing process (A*IPS). The resulting paths of our method are shorter than A*PS in some cases. A*IPS functions similar to A*PS, but the difference arises when the A*PS, during the graph visit, is checking the LOS from the visiting node to the successor of its successor. At this stage, A*IPS in parallel, is additionally checking the LOS from the successor of its successor to the goal node. A*IPS terminates the whole process as long as the LOS is discovered to the goal and the shortest path is achieved. In Fig. 3.2, we show an example to be more clear about the process. If we consider the point A as the starting point, the A*PS checks the LOS from point A, node by node, towards the goal. Since A*PS has no LOS (observing from point A) to any node located between point C and D, the process ends in point C and then starts over from point C to the goal. However, A*IPS is checking the LOS from both the starting point A and its following node to the goal simultaneously. As a result, A*IPS detects the LOS from point B to the goal and ends the process immediately. Notice that in this work, LOS is defined in terms of throughput values; we consider two points in LOS if all the closest nodes in the direct path among them possess throughput values of greater or equal to the current value.

In our algorithms, the post-smoothing process executes after finding the path in each iteration, enabling us to calculate the real energy consumption after smoothing (making the path shorter) and also considering the final turning angles of the path in calculating the energy consumption.

It will be shown in Chapter 5, Sec. 5.3, that the A*IPS resulting paths are shorter or equal to the A*PS, with a possible reduction in computation time.

Chapter 4

Algorithm Integration into the QGround Control (QGC) Path Planner

In this chapter we show how our approach can be utilized in practice to plan an optimal path for a drone in the real-world situations. First we introduce the MAVLink which is a telecommunication protocol designed for uncrewed vehicles. Some of the key features like efficiency, the ability to support numerous programming languages, and reliability that have made the MAVLink protocol very popular are noticed. Furthermore, we analyze the MAVLink messages in more details. The message structure and fields are investigated precisely and the application of each field is described. Then, we look into one of the most powerful open-source control stations, called QGroundControl, and its communication with the auto pilot. We show the high level data flow between the control station and the UAV. We investigate some specific and important mission commands that are exploited by the control station to communicate with the vehicles and were useful in our work. Moreover, we demonstrate the communication sequence of uploading a mission to a vehicle in a diagram with great detail. After that, we introduce our designed proxy in order to monitor the data which are passing between the control station and the vehicle and we study our proposed architecture. We explain in what way and when the proxy acts transparently or modifies the messages. The message flow among the control station, the proxy, and the vehicle is analyzed particularly. Finally, we validate our

approach by showing the resulting path which is different from the original path and passes through the original way-points and high throughput regions.

This chapter has previously been published as: Afshin Mardani, Marcello Chiaberge, and Paolo Giaccone. Communication-aware uav path planning. In *2019 IEEE Access journal*.

4.1 MAVLink Protocol

MAVLink (Micro Air Vehicle Link) is a binary telemetry communication protocol which is designed for systems with limited resources, and links with constrained bandwidth, in order to communicate with uncrewed vehicles [92]. MAVLink is developed in two versions: version 1.0 and version 2.0, while version 2.0 implementations can parse and send version 1.0 packets.

4.1.1 MAVLink features

Some of the key features that have made the MAVLink protocol very popular to serve as inter-operable interface between various manufacturers components, and to be deployed in many products, are as following:

- MAVLink is highly efficient. Since it requires no redundant framing (MAVLink v1.0 has 8 bytes of overhead and MAVLink v2.0 has 14 overhead bytes), it is very suitable for the applications that lack a wide communication band-width.
- MAVLink is reliable enough. MAVLink is field-proven since it has been utilized from 2009 for communication between various vehicles and base stations, and over various challenging communicating channels with high latency and noise. Moreover, it has methods for packet loss and packet authentication.
- It supports many programming languages like C, C++, Python, Java, and operates on various operating systems such as Linux, Windows, and MacOS. Furthermore, 255 systems (e.g. vehicles, base stations) can work simultaneously on the network. It also allows on-board (e.g. Auto pilot and camera) and off-board (e.g. GCS and drone) communications.

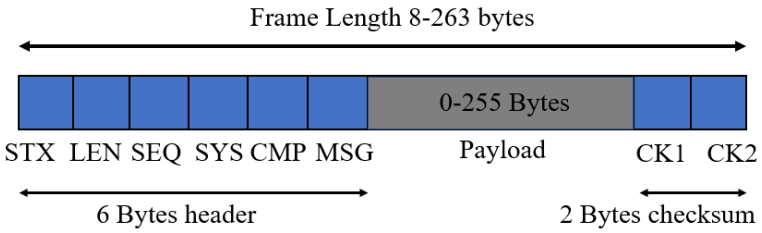

Fig. 4.1 Data structure of MAVLink messages.

4.1.2 MAVLink message

MUVLink messages are very lightweight and pursue a mixed publish-subscribe and point to point design pattern. Data streams like telemetry that usually there is not a unique recipient for them, are transmitted in a multi-cast way whereas the system configuration data that require to be delivered assuredly, are sent point to point with re-transmission. A large set of messages are defined by MAVLink protocol and can be find in XML files. The message set that can be implemented by most of the autopilots and ground control stations is defined in common.xml. All the standard definitions by the MAVLink project exist in this file.

Message Structure

Each MAVLink packet frame has a length of 8 to 263 bytes [93], with the frame structure shown in Fig. 4.1.

System ID (SYS) and component ID (CMP) fields are filled by the sender to inform the receiver about its identity. Each vehicle has a unique system ID. Vehicles usually use "1" system ID and base stations use "255". The sub-devices that are in the same system, use the same system ID and different component ID. Message (MSG) determines the type of the message (e.g. message ID of "HEARTBEAT" is "0"). Payload holds the actual data to be transmitted. Payload length (LEN) indicates the length of the payload. Packet sequence (SEQ) indicates the sequence number of each message that will be counted by each component allowing to detect the packet loss. Finally, packet start (STX) indicates the start of a new message.

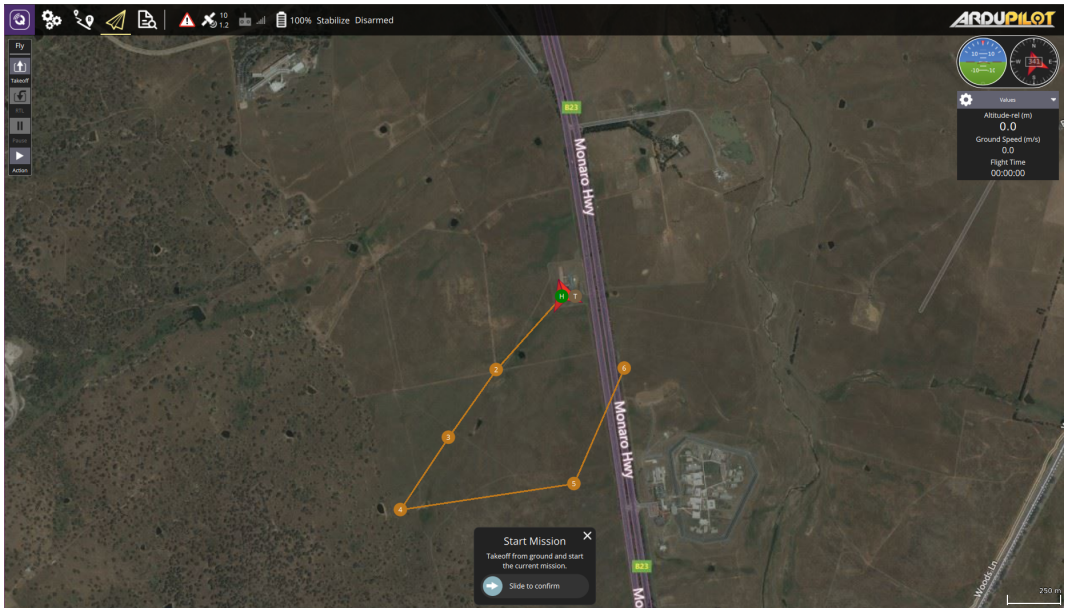

Fig. 4.2 QGC flight map display.

4.2 QGC Communication with Auto Pilot

4.2.1 QGroundControl Control Station

QGround Control (QGC) control station [4] is a very powerful open-source control station for UAVs that provides full flight control and mission planning for all vehicles which are MAVLink enabled, and vehicle setup for UAVs which are powered by PX4 and ArduPilot autopilots. It provides a straightforward and top quality feature support for the users. Fig. 4.2 shows the flight map display of the QGC that indicates the vehicle position, way-points, flight path, and vehicle instruments. Some of the QGC key features are as following:

- Providing a full configuration and setup for vehicles which are powered by ArduPilot and PX4 Pro autopilots.
- Providing flight support for all the vehicles that are powered by MAVLink protocol communicating autopilots, including PX4 and ArduPilot.
- Mission planning for autonomous flights.
- Flight map display.

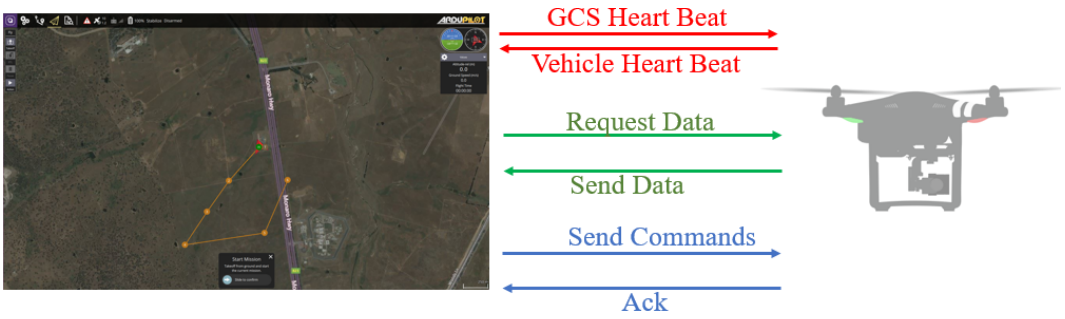

Fig. 4.3 High level data flow between QGC and UAV.

- Video streaming.
- Supporting multiple vehicle management.
- Supporting all major operating platforms, including Windows, OS X, Linux, iOS, and Android.

4.2.2 Mission Commands

The MAVLink protocol provides lots of command types that are supported on each of the vehicle types to communicate with the autopilot [93]. There are different types of commands that we categorize them in three main types as following:

- Navigation commands which are utilized to control the movement of the vehicle, such as takeoff and landing.
- Do commands that do not affect the movement of the vehicle and are utilized for auxiliary functions, such as camera settings.
- Condition commands that delay Do command until a condition is met, such as reaching a specific altitude.

During a mission, only one "Navigation" command can be run with one of the other two types of commands in the same time.

4.2.3 Message Flow

Fig. 4.3 is showing the high level data flow between the QGC and drone [93]. As long as the connection is established between the QGC and drone, they both send a "HEARTBEAT" message at 1 Hz to each other. In this way, they are informing each other that they are alive and respond. The QGC requests the data it needs by sending messages of type "REQUEST_DATA_STREAM" or "COMMAND_LONG" to the drone. The drone, depending on the type of request, answers to the requests of QGC by sending the requested data or an acknowledgement. We list a set of important messages that we used in our work below [92]:

- HEARTBEAT: This is the most important message between QGC and drone. They both keep sending this message to each other to make sure both are in sync and respond.
- REQUEST_DATA_STREAM: This message is sent when the target requests for data stream. This message includes the ID of the requested data stream and the requested message rate.
- COMMAND_LONG: This command is used by the QGC and it sends a command with up to seven parameters to the target vehicle. It includes the target system ID and the ID of the requested command that specifies the type of command such as arm/disarm.
- MISSION_REQUEST_LIST: This message is to request for the list of all mission items by the vehicle to initiate mission download.
- MISSION_COUNT: This message is to report the vehicle about the number of mission items. This is used to initiate mission upload, or in the response of the requested list when downloading a mission.
- MISSION_REQUEST: This message is to request for the mission items by the vehicle from the QGC. It asks for each item with the sequence number and the answer to this message is a MISSION_ITEM.
- MISSION_ITEM: This message encodes a mission item. It contains the positional information x,y, and z in float parameters and emitted in the response of mission request message.

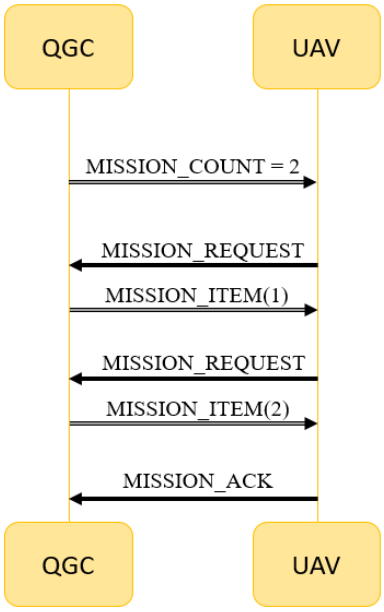

Fig. 4.4 The communication sequence of QGC and UAV when uploading a mission to the UAV.

Now in the following diagram, we can show the communication sequence to upload a mission to a vehicle (assuming that all the operations are successful). First of all, the QGC sends the MISSION_COUNT to the vehicle that includes the number of mission items. The vehicle receives the message and prepares to upload the items. Next, the vehicle responds to the QGC by sending the MISSION_REQUEST message, asking for the first mission item with $seq = 1$. After that, the QGC responds to the vehicle by sending the first mission item. This cycle repeats until all the mission items are uploaded. When the last mission item received by the vehicle, it responds to the QGC by sending a MISSION_ACK message and the operation is accomplished. It is notable that the QGC sets a *timeout* after each message and re-send the message if it receives no response from the vehicle. Moreover, the vehicle repeats its request if the sequences are mismatched.

4.3 Setting a Proxy Between QGC and Auto Pilot

In section 4.4, it will be shown that how we integrate our algorithm in QGC path planner. Earlier in this section, we explain the architecture and performance of the

Fig. 4.5 The architecture of the designed proxy.

designed proxy. Furthermore, we show the message flow of the new architecture and describe it in details.

4.3.1 Architecture

For the purpose of integrating our algorithms in the path planner, we need the permission to modify the mission way-points which are planned by the QGC and then upload the new way-points to the vehicle. Towards that end, firstly we require to monitor the data which are passing between the QGC and the vehicle. Therefore, we designed a transparent proxy that sits between the QGC and the vehicle, and redirects the MAVLink messages without any modification. Fig. 4.5 shows the architecture of the designed proxy. This proxy is transparent and acts as intermediary without modifying any data.

In order to implement the proxy, we used the DroneKit-Python [94]. The DroneKit API helps the developers to create powerful applications that communicate with vehicles over MAVLink protocol. The DroneKit-Python is the python language implementation of the DroneKit. It provides the users accessing to the vehicle's telemetry, parameter information, and vehicle's state. It enables mission management and direct control of vehicle operations and movements. DroneKit-Python can be used on an on-board companion computer and communicate with the autopilot using a low latency link, or can be used for ground station applications (like QGC) communicating with the vehicle using a higher latency RF-link.

We managed to forward the MAVLink messages coming from the UAV through the DroneKit-Python to QGC using TCP connections in both sides. We used the <MAVConnection.pipe()> method to execute the forwarding function. The code is already present in DroneKit [95] (look into mavlink.py file). The code starts a listener on TCP port 5762 to the vehicle and establish a connection on TCP port 5760 to the QGC. The link configuration of QGC in order to connect with the proxy running

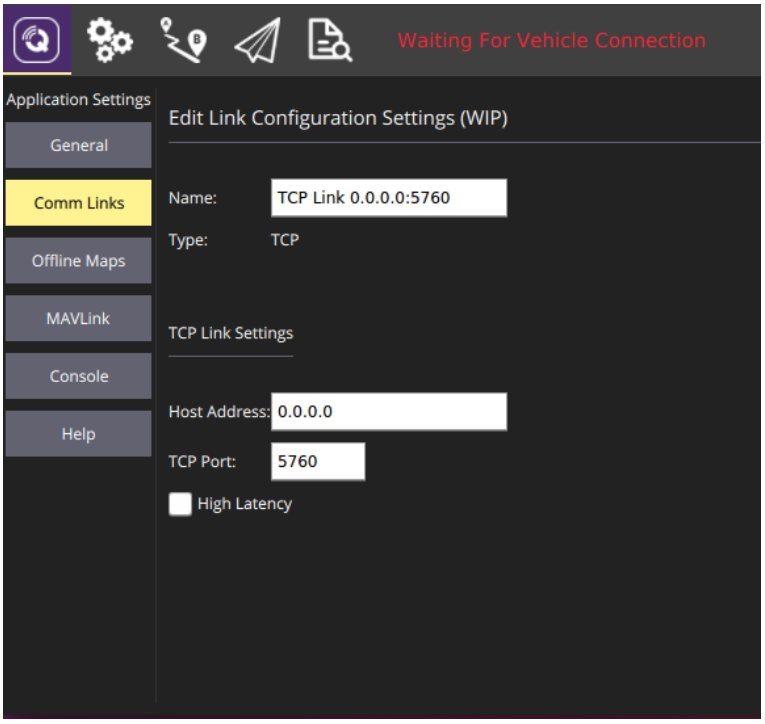

Fig. 4.6 The QGC connection configuration.

DroneKit is shown in Fig. 4.6. Comm Links allows us to create communication links manually and connect to them.

Message Modification

Up to now, the proxy simply forwards the messages coming from each side to the other side. As we described before, we need to modify the path between each two user-defined way-points in QGC by adding a number of new way-points. The new modified path which is planned by our algorithms and is different from the original straight line between each two way-points, experiences a higher average or minimum throughput with respect to the straight line that is planned by the QGC. Therefore, we modified the callback functions which are used to forward the messages from the vehicle to the target (QGC) and from the target to the vehicle. The callback functions are inside the pipe function (see mavlink.py file) [95]. After the modification, all the messages pass through the proxy unchanged except the "MISSION_ITEM"s. The proxy blocks all the mission items coming from the QGC and sends them to our algorithms for modification. Then, the proxy forwards the new modified way-points

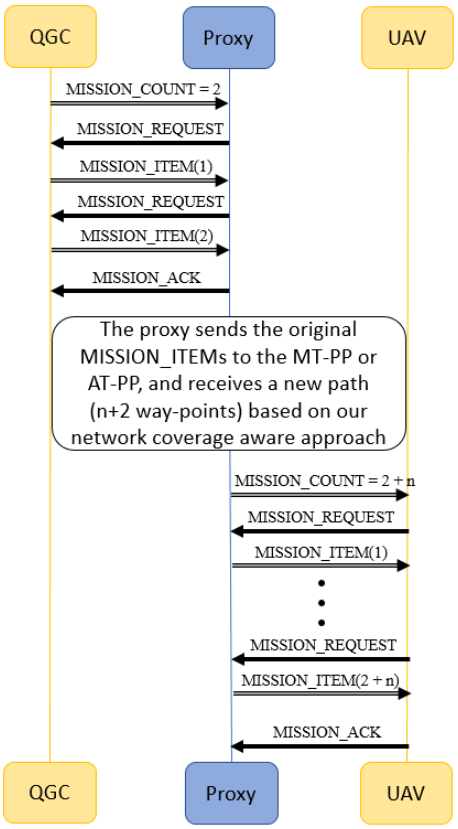

Fig. 4.7 The communication sequence of QGC and UAV in presence of the proxy when uploading a mission to the UAV.

(including the previous original way-points) which are planned by our algorithm, to the vehicle to carry out the mission. In case of mission upload, the proxy acts no more transparently.

4.3.2 Message Flow

In the diagram in Fig. 4.7, we show the detailed communication sequence to upload a mission to a vehicle while the proxy is set between the QGC and the vehicle (assuming that all the operations are successful). First of all, the QGC sends the MISSION_COUNT (includes the number of mission items) to the vehicle. Since the messages are related to the mission way-points, the proxy acts as a UAV and instead of forwarding the message to the UAV, it receives the message and responds to the QGC with a MISSION_REQUEST message, asking for the first mission item with $seq = 1$. After that, the QGC responds to the fake vehicle (the proxy) by sending

the first mission item. This cycle repeats until all the mission items are uploaded to the proxy. When the last mission item received by the proxy, it responds to the QGC by sending a MISSION_ACK message and the operation is accomplished. At this moment, the proxy sends the original MISSION_ITEMs, that are defined by the user from the QGC side, to our algorithms and waits for the new planned way-points from the algorithms. As long as the path computation is terminated, the new way-points are ready to be sent to the vehicle. Now, the proxy changes its role and sends a MISSION_COUNT message to the vehicle acting as QGC. The real vehicle responds to the fake QGC (the proxy) by sending a MESSAGE_REQUEST message, asking for the first mission item. This cycle repeats until all the mission items are uploaded to the vehicle and the vehicle terminates the process by sending a MISSION_ACK to the proxy. Accordingly, the new uploaded path experiences a better network coverage in terms of average or minimum throughput with respect to the shortest path.

4.4 Algorithm Integration into the QGroundControl Path Planner

Our approach can be utilized in practice to plan an optimal path for a drone in the real-world situations. As validation, we integrated our algorithms in a popular, open-source, offline path planner like QGroundControl (QGC) station [4]. To achieve a seamless integration, we designed a *proxy* between the QGC and the autopilot. All the data sending from the QGC towards the drone and sending from the drone to the QGC, are passing through the proxy. From the point of view of QGC, the proxy acts as a standard drone; from the point of view of the drone, the proxy acts as QGC. This allows to achieve a transparent behavior, which does not require any modification in QGC and the drone (except for proper configuration settings).

The proxy acts as a server, creating a TCP connection with the QGC, and as a client, creating another TCP connection with the drone. TCP connection is employed to provide reliable transport of drone configuration. Fig. 4.8 demonstrates the adopted architecture.

In a standard situation (when the proxy is not present), once the user is defining a mission on the QGC, the way-points are transferred to the drone by uploading the

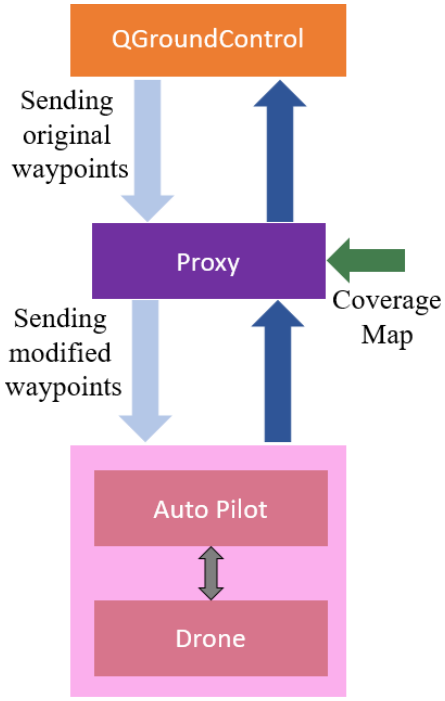

Fig. 4.8 Proof of concept for the path planner.

mission and the UAV starts the mission as long as a command is received from the QGC side by the user. In this case, the drone flies between each two way-points using the shortest path (a direct path). Now, by considering the proxy between the QGC and the autopilot, all the data sending and receiving from both sides are passing through the proxy. The proxy relays transparently all the data through, except if the data type is MISSION_ITEM. Each MISSION_ITEM contains the GPS coordinates of a way-point and all the other information (e.g., system ID) that are needed by the drone. These data are packed based on the MAVLINK telemetry protocol. Therefore, in the case that data are not way-points or not related to the start or end of way-point transmission, the proxy acts transparently and passes the data equivalent to a pipe. But, if the data are involved in defining way-points, the proxy acts double-faced. Whenever QGC is sending way-points, the proxy acts as the drone. In this case, it sends requests to the QGC for the way-points and receives them one by one and sends the acknowledgment to the QGC when all of the way-points are received. Then, the proxy feeds the original way-points to our algorithms and a new path is planned between each two original way-points. The new path is typically different from the default path (shortest path) and is designed according to our proposed network coverage aware approach.

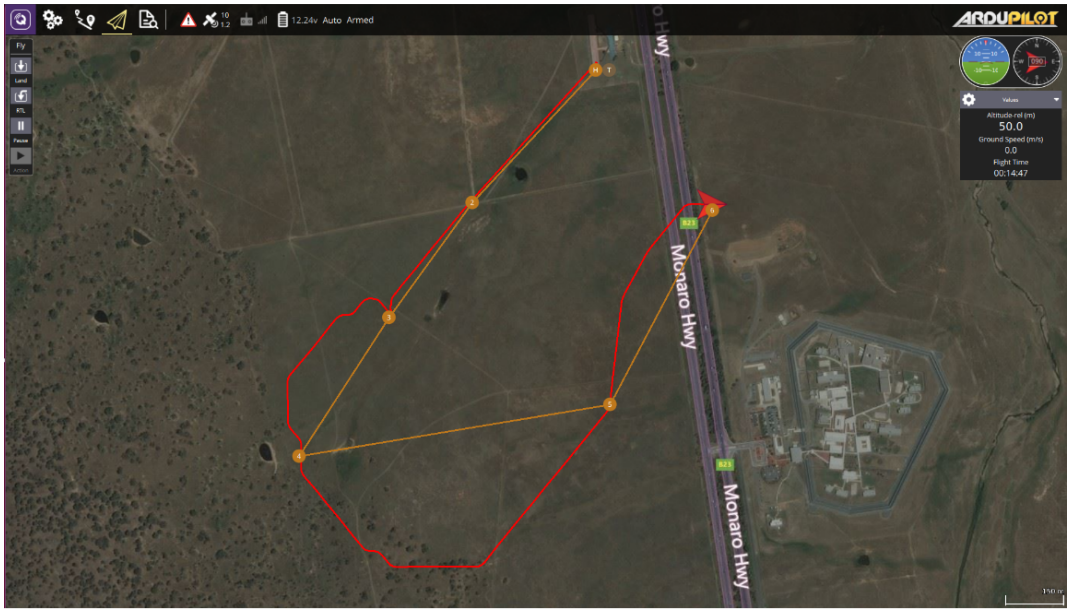

Fig. 4.9 The GUI of QGroundControl integrated with our approach. The orange path is the original path, whereas the red one is the one computed by MT-PP and uploaded to the drone.

As shown in Fig. 4.8, the coverage map is fed to the proxy. In the event that the proxy requires to send the new way-points to the drone, it acts as QGC. In this case, the proxy asks for sending the new way-points and answers to the requests of the drone for the way-points. When the transmission is finished, it receives the acknowledgement from the drone. Now, the drone receives a path which is passing by all the defined way-points and the throughput of the path is maximized considering the on-board energy budget and the effect of wind. Fig. 4.9 demonstrates the resulting path which is planned by MT-PP algorithm. The orange path is the default path that the drone passes through, in the usual direct connection (i.e., absence of the proxy) with the QGC. The red path is the one which designed by MT-PP and passes through the original way-points and high throughput regions. Depending on the algorithm (MT-PP or AT-PP), the red path may be different.

Chapter 5

Performance Evaluation

In this chapter, first we introduce the coverage maps, called cone map and valleys map, that we performed our experiments on them. Then we define the scenarios which are the selection of various source-destination pairs with the coverage map, and we considered them to evaluate our algorithms based on them. After that, the numerical result are presented. We present different comparisons of our results from our algorithms performing in different scenarios. We compare the average and minimum throughput obtained by our algorithms in all the defined scenarios in presence and absence of wind. Moreover, the path length and computation time of our algorithms are compared with the-state-of-the-art algorithms in all scenarios. Finally, a comparison in energy consumption of our algorithms with the-state-of-the-art algorithms is performed.

5.1 Scenarios

Fig. 5.1 reports the two coverage maps on a 4 km$\times$4 km area that we tested our algorithms on them. In the first map denoted as "cone" (left side of Fig. 5.1), maximum throughput shows in the center, then linearly it decreases to a minimum value, and finally it remains constant in the whole border with a higher value than the minimum. The performance of greedy approaches may be impaired by such discontinuity in the coverage (as the one considered in our work). Therefore, we can consider the cone map as a simple worst-case coverage scenario. In the second map

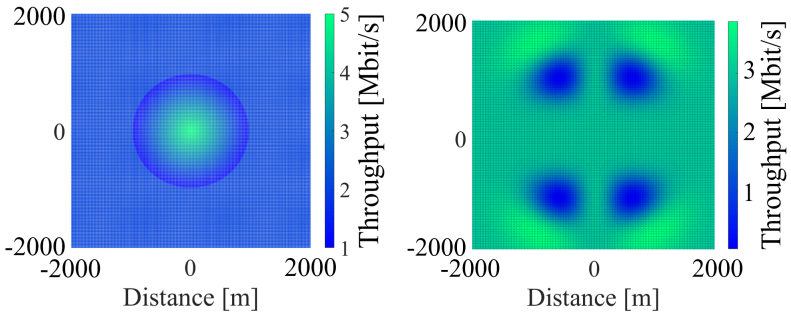

Fig. 5.1 The valleys map (right), and the cone map (left).

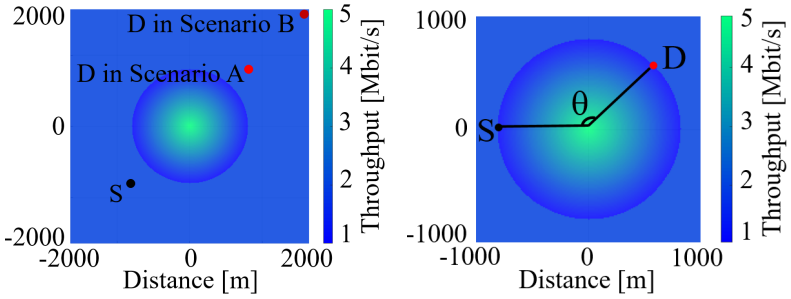

Fig. 5.2 Scenario A, B (left) and scenario C (right). "S" stands for source position. "D" stands for destination position.

denoted as "valleys" (right side of Fig. 5.1), the coverage is continuous with four low throughput valleys.

The UAV mission starts from a starting point to an ending point. We tried to combine the selection of various source-destination pairs with the coverage map, defining the following scenarios:

- Scenario A (Fig. 5.2): A symmetric case on the cone map. The source is positioned in (-1000,-1000) m and the destination in (1000,1000) m.
- Scenario B (Fig. 5.2): An asymmetric case on the cone map. The source is positioned in (-1000,-1000) m and the destination in (2000,2000) m.
- Scenario C (Fig. 5.2): We positioned the source and the destination at θ degrees from the peak of throughput on the cone map. It allows us to investigate how a path is deviated from its direction through the high throughput region.
- Scenario D (Fig. 5.3): The source is positioned in (-1000,-1500) m and the destination in (1000,1300) m, on the valleys map. Regarding the two low throughput regions between the source and destination on the direct path, this

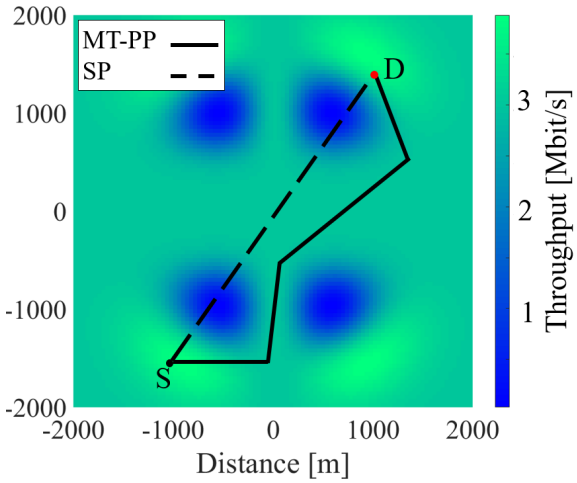

Fig. 5.3 Resulting paths obtained by MT-PP and SP under scenario D.

case enables us to demonstrate the deviation of the resulting path from the low throughput regions.

In this work, wind is called “head wind” when it is in the opposite direction of the direct path from source to destination, and is called “tail wind” when it is in the same direction.

5.2 Methodology

In this work, we compare the behavior of our algorithms with the Shortest Path (SP) algorithm, both in dealing with the wind presence and the on-board energy limit. SP is selected since it represents all the cited state-of-the-art algorithms in Chapter 3, Sec. 3.1, that compute the direct path from source to destination and are oblivious of the coverage map.

In the simulations of this work, as depicted in Fig. 3.1, 8-degree graph on a 101×101 grid with a distance of minimum 40 meters between two nodes is considered. Notably, the graph can simply be modified to represent the presence of obstacles by removing the corresponding nodes and edges. Thus, our algorithms can consider the presence of obstacles. However, we did not consider any obstacle in this work for the sake of simplicity.

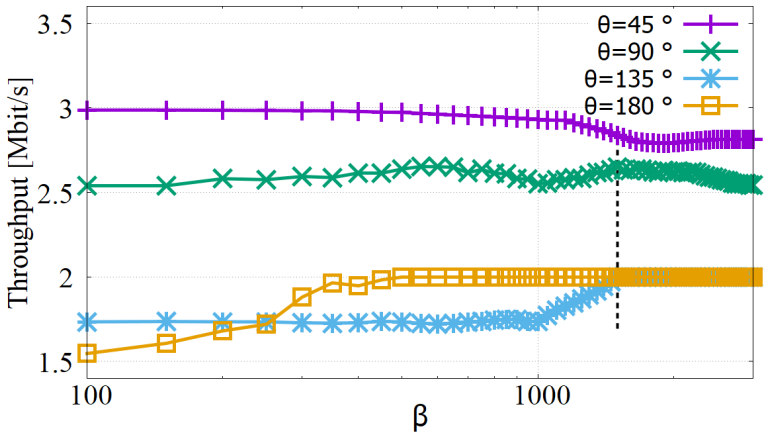

Fig. 5.4 β optimization under scenario C.

We implemented our algorithms in MATLAB and executed them on a 2.67 GHz Core i7 PC with 8 GByte of RAM running Windows 10. We used `tic` and `tac` MATLAB commands to evaluate the running times.

5.3 Numerical evaluation

Firstly the choice of coefficient β which is employed in the cost function of our algorithms is optimized. Towards this end, under scenario C, we selected four destinations in $\theta = 45, 90, 135$, and 180 degrees. In Fig. 5.4, throughput in function of β is plotted. In order to obtain a high throughput value in most of the cases, it is set to $\beta = 1500$ (shown with dashed line in Fig. 5.4).

Table 5.1 compares the obtained average throughput and Table 5.2 compares the obtained minimum throughput along the resulting paths of our algorithms with SP, for all the defined scenarios, in presence and absence of wind. We employed the IPS method to smooth the resulting paths of both the MT-PP and AT-PP algorithms.

Consider for now the performance without wind. Fig. 5.3 is verifying the expected performance of MT-PP under scenario D. In this figure, the obtained paths by MT-PP and SP are depicted. Considering the Table 5.1 and 5.2, MT-PP and AT-PP both achieved a greater average throughput comparing with SP (50% greater for MT-PP) whereas the achieved minimum throughput is much greater than SP (9 times greater for MT-PP.)

Table 5.1 Average throughput comparison.

Scenario	Algorithm	Average Throughput [Mbit/s]		
		no wind	head wind 2 m/s	tail wind 2 m/s
D	MT-PP + IPS	3.3	2.6	3.5
	AT-PP + IPS	2.7	2.3	2.8
	SP	2.2	2.2	2.2
C $\theta = 135^\circ$	AT-PP + IPS	2.8	1.5	2.8
	MT-PP + IPS	1.5	1.5	1.5
	SP	1.5	1.5	1.5
A	MT-PP + IPS	2.0	2.0	2.0
	AT-PP + IPS	2.5	2.5	2.5
	SP	2.5	2.5	2.5
B	MT-PP + IPS	2.0	2.0	2.0
	AT-PP + IPS	2.3	2.3	2.3
	SP	2.3	2.3	2.3

Table 5.2 Minimum throughput comparison.

Scenario	Algorithm	Minimum Throughput [Mbit/s]		
		no wind	head wind 2 m/s	tail wind 2 m/s
D	MT-PP + IPS	3.0	1.0	3.1
	AT-PP + IPS	1.8	1.0	2.2
	SP	0.3	0.3	0.3
C $\theta = 135^\circ$	AT-PP + IPS	0.1	0.1	0.1
	MT-PP + IPS	0.1	0.1	0.1
	SP	0.1	0.1	0.1
A,B	MT-PP + IPS	2.0	2.0	2.0
	AT-PP + IPS	0.1	0.1	0.1
	SP	0.1	0.1	0.1

Fig. 5.5 shows the path found by AT-PP under scenario C ($\theta = 135^\circ$), verifying the algorithm behavior. Specifically the resulting path tends to pass the close proximity of the cone vertex with maximum throughput. This path is optimal, considering that the average throughput is maximized. Thus, the average throughput of the path is ameliorated up to 1.3 Mbit/s comparing with SP.

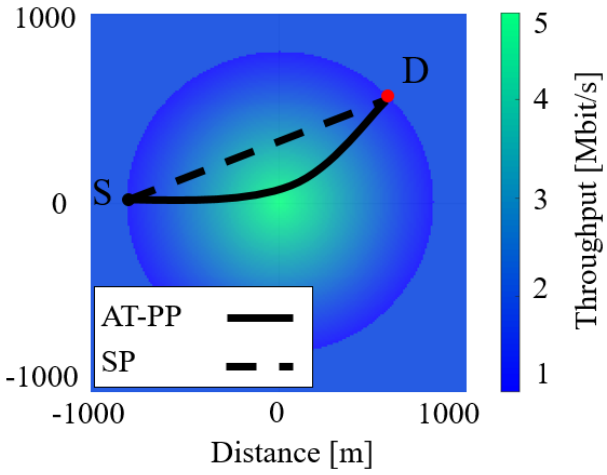

Fig. 5.5 Resulting paths obtained by AT-PP and SP under scenario C ($\theta = 135°$).

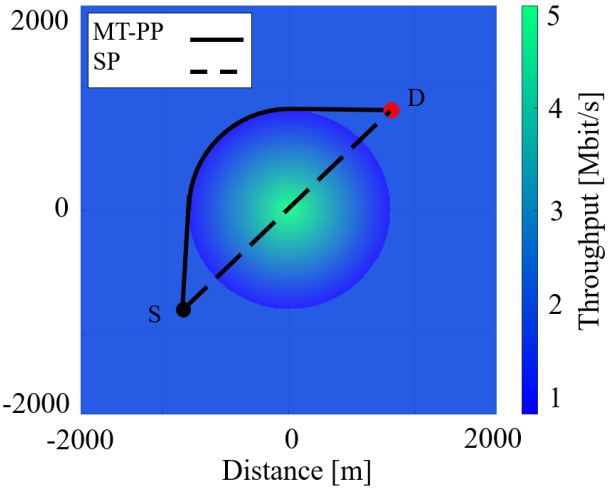

Fig. 5.6 Resulting paths obtained by MT-PP and SP under scenario A.

In addition, in scenarios A in Fig. 5.6, and in scenario B in Fig. 5.7, MT-PP achieves a minimum throughput that outperforms the SP by about 20 times.

Consider for now the performance without wind. Fig. 5.3 is verifying the expected performance of MT-PP under scenario D. In this figure, the obtained paths by MT-PP and SP are depicted. Considering the Table 5.1 and 5.2, MT-PP and AT-PP both achieved a greater average throughput comparing with SP (50% greater for MT-PP) whereas the achieved minimum throughput is much greater than SP (9 times greater for MT-PP.)

Fig. 5.5 shows the path found by AT-PP under scenario C ($\theta = 135°$), verifying the algorithm behavior. Specifically the resulting path tends to pass the close proxim-

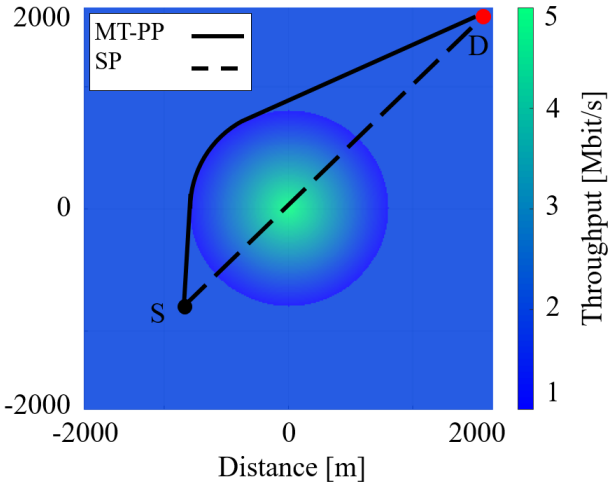

Fig. 5.7 Resulting paths obtained by MT-PP and SP under scenario B.

ity of the cone vertex with maximum throughput. This path is optimal, considering that the average throughput is maximized. Thus, the average throughput of the path is ameliorated up to 1.3 Mbit/s comparing with SP.

In addition, in scenarios A in Fig. 5.6, and in scenario B in Fig. 5.7, MT-PP achieves a minimum throughput that outperforms the SP by about 20 times.

Now, we consider the wind effect. MT-PP and AT-PP gain less in the presence of head wind, comparing with either the tail wind or the wind absence. The reason behind is the restricted amount of on-board energy that prevents the drone from flying into the high throughput regions and compensating for the wind concurrently. The throughput of the path in scenario A and B is not influenced by the wind in either case, since MT-PP has a tendency to keep away from the low throughput regions so far as energy is provided. The resulting paths of AT-PP and MT-PP in scenario C and D, in presence of head wind, are approaching to SP leaving more energy behind for the wind compensation. As a consequence, the throughput of resulting paths are lower than the tail wind or no wind cases.

In the considered offline path-planning scenario, the wind can be taken into account just based on some forecast weather model. For an area of few km squares (or at most tens of them) where the UAV will fly, we expect to know in advance only an average wind value for each tile.

Nevertheless, the algorithm is compatible, without any modification, with variable winds, i.e. wind intensity different for each point of the grid map. In Fig 5.8,

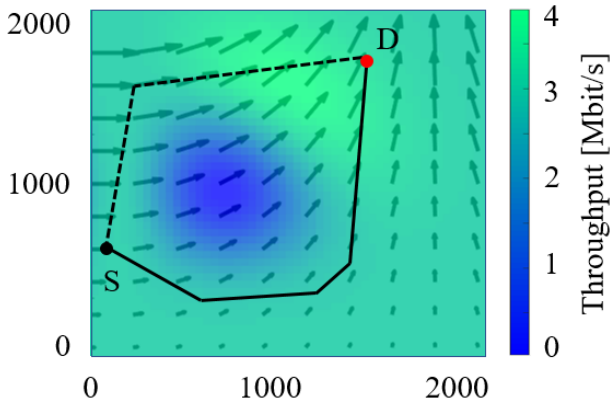

Fig. 5.8 Comparing the performance of MT-PP in presence of variable wind (solid line), and absence of wind (dashed line).

we considered variable wind in the field. In this scenario, the MT-PP algorithm planned a longer path comparing with the dashed line path which is the shorter path in the absence of wind. The resulting path of MT-PP in presence of variable wind is longer, but the energy consumption of the drone is lower, compared to the no wind case, because of the presence of tail wind during the path. Moreover, the energy consumption of the dashed line path in presence of variable wind is higher because of the presence of cross wind in half of the path.

In Table 5.3, the effect of path smoothing methods is investigated. In this table, the path length and the corresponding computation time for the resulting paths of MT-PP algorithm alone (i.e., without any post-smoothing), MT-PP with standard post-smoothing, and MT-PP with our improved post-smoothing method under all the defined scenarios are compared. By construction, the least computation time belongs to MT-PP with no post-smoothing applied, while applying standard PS increases the computation time by 3.4%, in scenario B (as the worst case). Alternatively, applying IPS to MT-PP in scenario B adds up to 1.0% additional time. Regarding the path length, PS and IPS equally shorten the path lengths by 5.1% in scenario A, whereas they shorten the length of paths by 5.4% and 6.3% respectively, in scenario B. Therefore, IPS has always priority over PS in terms of both path length and computation time.

The effect of energy budget is investigated in Table 5.4. We compared the energy consumption of the MT-PP resulting paths of each iteration in scenario D with the SP, in one part of table, and the output paths of AT-PP in case of setting an energy bound for scenario C ($\theta = 135^\circ$). In scenario D, MT-PP finds the path 1 in the first

Table 5.3 Path length and computation time comparison.

Algorithm	Scenario	Run time [s]	Path length [m]
MT-PP	A	1.18	3507
	B	2.91	4922
	C ($\theta = 135°$)	0.05	1535
	D	1.52	4715
MT-PP + PS	A	1.21	3325
	B	3.01	4655
	C ($\theta = 135°$)	0.07	1418
	D	1.55	4450
MT-PP + IPS	A	1.21	3325
	B	2.94	4610
	C ($\theta = 135°$)	0.05	1418
	D	1.57	4435

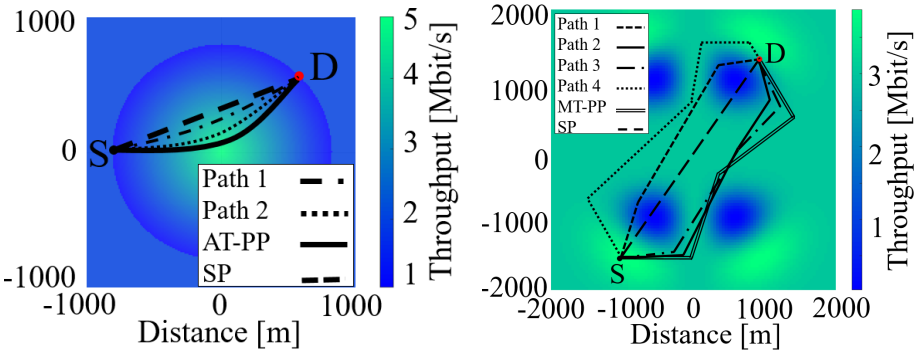

Fig. 5.9 The effect of energy budget on the AT-PP (left), and MT-PP (right).

iteration. The energy consumption limit is set to 91 kJ in this experiment. Since the energy consumption is not reached the limit, it continues to increase the threshold up to the path 4 which exceeds the energy limit. Then the threshold is decreased and the final MT-PP path is found. The path of each iteration is shown in Fig. 5.9 (right). In Scenario C, each resulting path is the output of AT-PP with a specified energy limit. It is obvious in Fig. 5.9 (left) that as long as we set a greater energy limit, the path will be longer and closer to the high throughput region. It is notable to mention from Table 5.4 that the energy consumption of MT-PP and AT-PP paths are clearly more than the SP, but they meet the energy budget constraint. In this experiment, the angular rotation speed (ω_{turn}) is considered $2.1 rad/s$, and the power consumed during the rotations (P_{turn}) is considered $225W$. Power consumption of the drone (P) in all the experiments in this work is considered $200W$ [46].

Table 5.4 Comparison of energy consumption.

Algorithm & Scenario	Path	Energy Consumption [kJ]
MT-PP + IPS Scenario D	1	74.5
	2	85.8
	3	88.5
	4	93.6
	MT-PP	90.5
	SP	64.6
AT-PP + IPS Scenario C ($\theta = 135^\circ$)	1	31.0
	2	34.2
	AT-PP	36.7
	SP	29.4

Chapter 6

Conclusions

This chapter gives an overview and summarizes the work accomplished in this thesis, covering the whole contributions.

Inspired by providing cellular connectivity for UAVs, in order to guarantee the requirements of QoS in the communication for video streaming, we investigated the path optimization problem for an autonomous drone. In the study, the cellular coverage map, the drone budget of energy, and the possible presence of wind is considered. Two algorithms are presented to maximize either the worst-case throughput (MT-PP) or the average throughput (AT-PP) along the path. In addition, a novel path smoothing method outperforming the classical one in both terms of path length and computation time is proposed. It is shown, through a numerical evaluation of various scenarios, that our communication-aware approach leads to ameliorate the throughput of the path comparing with the classical approaches that are absolutely oblivious of the cellular coverage maps, with a profitable impact on video streaming applications. We evaluated the performance of the MT-PP and AT-PP algorithms, in presence and absence of the wind. We showed that, in absence of the wind or in presence of the tail-wind, our algorithms in most of the cases outperform the shortest path significantly. Besides, in presence of the head-wind, MT-PP and AT-PP, due to the restricted energy budget, gain less than the tail-wind or when the wind is absent. We investigated the effect of the energy budget on the resulting paths of MT-PP and AT-PP. We showed that the energy budget may prevent our algorithms from achieving the throughput-optimal path. Finally, we validated experimentally

our approach by integrating our algorithms in a popular offline path planner (QGC). We proved that our algorithms can be practically utilized in real-world path planners.

References

[1] 3GPP TR 36.777, Enhanced LTE support for aerial vehicles. ftp://www.3gpp.org/specs/archive/36_series/36.777.

[2] Qualcomm and AT&T to Trial Drones on Cellular Network to Accelerate Wide-Scale Deployment. https://about.att.com/story/qualcomm_and_att_to_trial_drones_on_cellular_network.html.

[3] Rp-170779, Study on enhanced LTE support for aerial vehicles. http://www.3gpp.org/ftp/tsg_ran/tsg_ran/TSGR_75/Docs/RP-170779.zip. Accessed on December 14, 2017.

[4] QGroundControl drone control. http://qgroundcontrol.com/.

[5] Edsger Wybe Dijkstra. A note on two problems in connexion with graphs:(numerische mathematik, _1 (1959), p 269-271). 1959.

[6] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Transactions on Systems Science and Cybernetics*, 1968.

[7] Federal Aviation Administration. FAA aerospace forecast: Fiscal years 2016–2036. *Tech. report TC16-002*, 2016.

[8] Alena Otto, Niels Agatz, James Campbell, Bruce Golden, and Erwin Pesch. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, pages 411–458, 2018.

[9] Unmanned Aerial Vehicle (UAV) Utilization of Cellular Services. https://www.atis.org, 2017.

[10] D. Simmons. Rwanda begins Zipline commercial drone deliveries. 2016.

[11] Package delivery with a drone: DPDgroup starts the first regular drone service worldwide. https://www.dpd.com/group/wp-content/uploads/sites/77/2019/02/20190218_DPDgroup_Drone_datasheet.pdf, 2016.

[12] The Aerial Services Department of Lufthansa starts partnership with a leading drone producer DJI. *Tech. report 26th of January 2016*.

[13] N. Chamaria. Drone usage in agriculture could be a $32 billion market. *Motley Fool*, 2016.

[14] The FAA Finally Provides a UAV Exemption for Yamaha. https://www.droneuniversities.com/drones/the-faa-finally-provides-a-uav-exemption-for-yamaha/, 2015.

[15] Michał Mazur, A Wisniewski, and Jeffery McMillan. Clarity from above: Pwc global report on the commercial applications of drone technology. *Warsaw: Drone Powered Solutions, PriceWater house Coopers*, 2016.

[16] David T. Wooden. *Graph-based Path Planning for Mobile Robots*. PhD thesis, Georgia Institute of Technology, 2006.

[17] Anthony Stentz. The d* algorithm for real-time planning of optimal traverses. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1994.

[18] Larry D Jackel, Eric Krotkov, Michael Perschbacher, Jim Pippine, and Chad Sullivan. The darpa lagr program: Goals, challenges, methodology, and phase i results. *Journal of Field robotics*, 23(11-12):945–973, 2006.

[19] Danny Z Chen, Robert J Szczerba, and JJ Uhran. Planning conditional shortest paths through an unknown environment: A framed-quadtree approach. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 33–38. IEEE, 1995.

[20] Anthony Stentz et al. The focussed dˆ* algorithm for real-time replanning. In *IJCAI*, volume 95, pages 1652–1659, 1995.

[21] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[22] Liang Yang, Juntong Qi, Dalei Song, Jizhong Xiao, Jianda Han, and Yong Xia. Survey of robot 3d path planning algorithms. *Journal of Control Science and Engineering*, 2016:5, 2016.

[23] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 521–528. IEEE, 2000.

[24] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[25] Anna Yershova, Léonard Jaillet, Thierry Siméon, and Steven M LaValle. Dynamic-domain rrts: Efficient exploration by controlling the sampling domain. 2005.

[26] Michael Ian Shamos and Dan Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 151–162. IEEE, 1975.

[27] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[28] Christopher I Connolly, J Brian Burns, and R Weiss. Path planning using laplace's equation. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 2102–2106. IEEE, 1990.

[29] Elon Rimon and Daniel E Koditschek. Exact robot navigation using artificial potential functions. *Departmental Papers (ESE)*, page 323, 1992.

[30] Satish Sundar and Zvi Shiller. Optimal obstacle avoidance based on the hamilton-jacobi-bellman equation. *IEEE transactions on robotics and automation*, 13(2):305–310, 1997.

[31] NJ Nilson. A mobile automation: An application of artificial intelligence techniques. In *Proceedings of the Fith International Joint Conference on Artificial Intelligence*, page 509, 1969.

[32] J. Canny. The complexity of robot motion planning. Technical report, Cambridge, MA: MIT Press, 1987.

[33] Tomas Lozano-Perez. A simple motion-planning algorithm for general robot manipulators. *IEEE Journal on Robotics and Automation*, 3(3):224–238, 1987.

[34] Christos H Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information processing letters*, 20(5):259–263, 1985.

[35] John H Reif and James A Storer. A single-exponential upper bound for finding shortest paths in three dimensions. *Journal of the ACM (JACM)*, 41(5):1013–1019, 1994.

[36] Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

[37] Jose A Cobano, Roberto Conde, David Alejo, and Aníbal Ollero. Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties. In *2011 IEEE International Conference on Robotics and Automation*, pages 4429–4434. IEEE, 2011.

[38] Isaac YF Lun and Joseph C Lam. A study of weibull parameters using long-term wind observations. *Renewable energy*, 20(2):145–153, 2000.

[39] Timothy McGee, Stephen Spry, and Karl Hedrick. Optimal path planning in a constant wind with a bounded turning rate. In *AIAA*, 2005.

[40] John Osborne and Rolf Rysdyk. Waypoint guidance for small uavs in wind. In *Infotech@ Aerospace*, page 6951. 2005.

[41] Derek R Nelson, D Blake Barber, Timothy W McLain, and Randal W Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007.

[42] Rolf Rysdyk. Unmanned aerial vehicle path following for target observation in wind. *Journal of guidance, control, and dynamics*, 29(5):1092–1100, 2006.

[43] Cunjia Liu, Owen McAree, and Wen-Hua Chen. Path-following control for small fixed-wing unmanned aerial vehicles under wind disturbances. *International Journal of Robust and Nonlinear Control*, 23(15):1682–1698, 2013.

[44] Alexandru Brezoescu, Tadeo Espinoza, Pedro Castillo, and Rogelio Lozano. Adaptive trajectory following for a fixed-wing uav in presence of crosswind. *Journal of Intelligent & Robotic Systems*, 69(1-4):257–271, 2013.

[45] Sanghyuk Park, John Deyst, and Jonathan How. A new nonlinear guidance logic for trajectory tracking. In *AIAA guidance, navigation, and control conference and exhibit*, page 4900, 2004.

[46] Carmelo Di Franco and Giorgio Buttazzo. Energy-aware coverage path planning of UAVs. In *ICARSC*. IEEE, 2015.

[47] Yong Zeng and Rui Zhang. Energy-efficient UAV communication with trajectory optimization. *Transactions on Wireless Communications*, 2017.

[48] Kai Li, Wei Ni, Xin Wang, Ren Ping Liu, Salil S Kanhere, and Sanjay Jha. Energy-efficient cooperative relaying for unmanned aerial vehicles. *Transactions on Mobile Computing*, 2016.

[49] Esten Ingar Grøtli and Tor Arne Johansen. Path planning for UAVs under communication constraints using SPLAT! and MILP. *Journal of Intelligent & Robotic Systems*, 2012.

[50] Yuwei Huang, Jie Xu, Ling Qiu, and Rui Zhang. Cognitive UAV communication via joint trajectory and power control. *arXiv preprint arXiv:1802.05090*, 2018.

[51] Kejian Wu, Baiqing Fan, and Xiao Zhang. Trajectory following control of UAVs with wind disturbance. In *CCC*. IEEE, 2017.

[52] Alessandro Rucco, A Pedro Aguiar, Fernando Lobo Pereira, and João Borges de Sousa. A predictive path-following approach for fixed-wing unmanned aerial vehicles in presence of wind disturbances. In *Robot*. Springer, 2016.

[53] Syed Ussama Ali, M Zamurad Shah, Raza Samar, and Athar Waseem. Wind estimation for lateral path following of UAVs using higher order sliding mode. In *ICISE*. IEEE, 2016.

[54] Unmanned Systems Technology, "KDDI and Terra Drone Announce New 4G LTE Drone Control Network". https://www.terra-drone.net/global/2017/04/10/kddi-and-terra-drone-have-announced-completion-of-inventing-4g-lte-control-system/.

[55] Xingqin Lin, Vijaya Yajnanarayana, Siva D Muruganathan, Shiwei Gao, Henrik Asplund, Helka-Liina Maattanen, Mattias Bergstrom, Sebastian Euler, and Y-P Eric Wang. The sky is not the limit: Lte for unmanned aerial vehicles. *IEEE Communications Magazine*, 56(4):204–210, 2018.

[56] MVSN Prasad, MM Gupta, SK Sarkar, and Iqbal Ahmad. Antenna beam tilting effects in fixed and mobile communication links. *Current science*, 88(7):1142–1147, 2005.

[57] "Study on Enhanced LTE Support for Aerial Vehicles". http://www.3gpp.org/ftp/tsg_ran/tsg_ran/TSGR_75/Docs/RP-170779.zip. accessed Dec. 14, 2017.

[58] Yong Zeng, Jiangbin Lyu, and Rui Zhang. Cellular-connected uav: Potential, challenges, and promising technologies. *IEEE Wireless Communications*, 26(1):120–127, 2019.

[59] Support for UAV Communication in 3GPP Cellular Standards. https://www.atis.org, 2018.

[60] Afshin Mardani, Marcello Chiaberge, and Paolo Giaccone. Communication-aware uav path planning. In *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*.

[61] Liang Yang, Juntong Qi, Jizhong Xiao, and Xia Yong. A literature review of UAV 3D path planning. In *WCICA*. IEEE, 2014.

[62] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in neural information processing systems*, 2004.

[63] Alex Nash, Sven Koenig, and Craig Tovey. Lazy theta*: Any-angle path planning and path length analysis in 3D. In *Third Annual Symposium on Combinatorial Search*, 2010.

[64] Alex Nash, Kenny Daniel, Sven Koenig, and Ariel Felner. Thetaˆ*: any-angle path planning on grids. In *AAAI*, 2007.

[65] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic A*: An anytime, replanning algorithm. In *ICAPS*, 2005.

[66] Xueqian Song and Shiqiang Hu. 2D path planning with Dubins-path-based A* algorithm for a fixed-wing UAV. In *ICCSSE*. IEEE, 2017.

[67] Alex Nash and Sven Koenig. Any-angle path planning. *AI Magazine*, 2013.

[68] Jie Zeng, Xuejun Zhang, and Xiangmin Guan. Path planning for general aircrafts under complex scenarios using an improved NSGA-II algorithm. *Journal of Computational Information Systems*, 2013.

[69] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 2010.

[70] Hrishikesh Sharma and P Balamuralidhar. A communication approach for aerial surveillance of long linear infrastructures in non-urban terrain. In *NCC*. IEEE, 2015.

[71] Feng Jiang and A Lee Swindlehurst. Optimization of UAV heading for the ground-to-air uplink. *Journal on Selected Areas in Communications*, 2012.

[72] Kazuya Anazawa, Peng Li, Toshiaki Miyazaki, and Song Guo. Trajectory and data planning for mobile relay to enable efficient internet access after disasters. In *GLOBECOM*. IEEE, 2015.

[73] Zhu Han, A Lee Swindlehurst, and KJ Ray Liu. Optimization of MANET connectivity via smart deployment/movement of unmanned air vehicles. *Transactions on Vehicular Technology*, 2009.

[74] Seungkeun Kim, Hyondong Oh, Jinyoung Suk, and Antonios Tsourdos. Coordinated trajectory planning for efficient communication relay using multiple UAVs. *Control Engineering Practice*, 2014.

[75] Feng Jiang and A Lee Swindlehurst. Dynamic UAV relay positioning for the ground-to-air uplink. In *GC Wkshps*. IEEE, 2010.

[76] Pengcheng Zhan, Kai Yu, and A Lee Swindlehurst. Wireless relay communications with unmanned aerial vehicles: Performance and optimization. *Transactions on Aerospace and Electronic Systems*, 2011.

[77] Qingqing Wu, Yong Zeng, and Rui Zhang. Joint trajectory and communication design for multi-UAV enabled wireless networks. *IEEE Transactions on Wireless Communications*, 2018.

[78] Qingqing Wu and Rui Zhang. Common throughput maximization in UAV-enabled ofdma systems with delay consideration. *available online: arxiv. org/abs/1801.00444*, 2018.

[79] Nan Zhao, Fen Cheng, F Richard Yu, Jie Tang, Yunfei Chen, Guan Gui, and Hikmet Sari. Caching UAV assisted secure transmission in hyper-dense networks based on interference alignment. *IEEE Transactions on Communications*, 2018.

[80] Fen Cheng, Shun Zhang, Zan Li, Yunfei Chen, Nan Zhao, Richard Yu, and Victor CM Leung. UAV trajectory optimization for data offloading at the edge of multiple cells. *IEEE Transactions on Vehicular Technology*, 2018.

[81] Yong Zeng, Rui Zhang, and Teng Joon Lim. Wireless communications with unmanned aerial vehicles: opportunities and challenges. *Communications Magazine*, 2016.

[82] Nicola Bezzo, Kartik Mohta, Cameron Nowzari, Insup Lee, Vijay Kumar, and George Pappas. Online planning for energy-efficient and disturbance-aware UAV operations. In *IROS*. IEEE, 2016.

[83] Stefano Primatesta, Elisa Capello, Roberto Antonini, Marco Gaspardone, Giorgio Guglieri, and Alessandro Rizzo. A cloud-based framework for risk-aware intelligent navigation in urban environments. In *ICUAS*. IEEE, 2017.

[84] Daniele Palossi, Michele Furci, Roberto Naldi, Andrea Marongiu, Lorenzo Marconi, and Luca Benini. An energy-efficient parallel algorithm for real-time near-optimal UAV path planning. In *Proceedings of the ACM International Conference on Computing Frontiers*, 2016.

[85] Nuwan Ganganath, Chi-Tsun Cheng, and K Tse Chi. Multiobjective path planning on uneven terrains based on NAMOA. In *ISCAS*. IEEE, 2016.

[86] Martin Seleckỳ, Petr Váňa, Milan Rollo, and Tomáš Meiser. Wind corrections in flight path planning. *International Journal of Advanced Robotic Systems*, 2013.

[87] Mangal Kothari, Ian Postlethwaite, and Da-Wei Gu. UAV path following in windy urban environments. *Journal of Intelligent & Robotic Systems*, 2014.

[88] Hector Garcia de Marina, Yuri A Kapitanyuk, Murat Bronz, Gautier Hattenberger, and Ming Cao. Guidance algorithm for smooth trajectory tracking of a fixed wing UAV flying in wind flows. In *ICRA*. IEEE, 2017.

[89] C Thorpe and L Matthies. Path relaxation: Path planning for a mobile robot. In *OCEANS*. IEEE, 1984.

[90] Jack E Bresenham. Algorithm for computer control of a digital plotter. *Systems journal*, 1965.

[91] Kenny Daniel, Alex Nash, Sven Koenig, and Ariel Felner. Theta*: Any-angle path planning on grids. *Journal of AI*, 2010.

[92] MAVLink developer guide. https://mavlink.io/en/.

[93] MAVLink basics. https://ardupilot.org/.

[94] DroneKit-Python documentation. http://python.dronekit.io/.

[95] DroneKit-Python library. https://github.com/dronekit/dronekit-python.