TÉCNICO LISBOA

DroneOpenStreetMaps

A High-Level Tool for Drone Mission Planning

André Guilherme Pires Carvalho

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Manuel Fernando Cabido Lopes

Examination Committee

Chairperson: Prof. Nuno João Neves Mamede
Supervisor: Prof. Manuel Fernando Cabido Lopes
Member of the Committee: Prof. Alberto Manuel Martinho Vale

July 2021

Acknowledgments

First, I would like to thank my advisor – Prof. Manuel Lopes – for the theme suggestion and guidance provided, always with a wise and confident mindset throughout all these months.

Secondly, I would like to thank my parents for their passive support throughout my entire life. Their delicate pressure towards education and accomplishment made me achieve my proudest objectives.

Right after, a special thanks to my grandmother, my girlfriend, and my brother. Their consistent presence and questions such as: ”How’s your thesis ?”; ”When do you finish ?”; and ”Once you’re done, can we do ’this or that’!?”; motivated me to work through the most challenging weeks and defeat my frequent procrastination.

Finally, I would like to thank my closest friends and relatives. Their positive words and ambitions surpassed my own, insistently believing in my capabilities to exceed my obstacles and fears, keeping me optimistic through these past months.

To each and every person who contributed directly or indirectly to my academic experience -

Thank you.

Abstract

With the manufacturing cost reduction and the improvements in robustness, control, and autonomy, Drones are now widely available for professional and personal use. With all this technology various applications such as delivery, surveillance, aerial inspections, and many more where drones can be employed, navigation is one aspect in common in all of them and usually requires specialized handling. However, automatic navigation for drones is still an underdeveloped topic with many challenges.

To address this problem and help non-specialized people with their drone tasks, this work focuses on an navigation sub-problem, the automatic trajectory planning. We created a high-level tool to automatically plan low-altitude trajectories, from a start to an end location, verifying several geographical and legal constraints, called DroneOpenTool. This tool automates the geographical information gathering from the flight location surroundings, plans a near-optimal solution given start and goal locations, flying restrictions, and resolution, and outputs a flyable trajectory to be uploaded into a drone control system. To evaluate our tool, we tested the performance of each part of our system, closely compared both our planning approaches in various scenarios, and related its outputs' efficiency with terrestrial solutions.

Although we did not perform practical experiments, our tool's outputs are strictly shaped to apply to real-world drone missions and achieve the expected approximate safe and optimized navigation outcomes. Base on the demonstrated results, we can state that the taken approach is viable within the initial expectations, yet far from a stable performance required by a reliable drone mission control system – at least under the followed design.

Keywords

Drone; Trajectory Planning; A* Algorithm; Flight Tool; UAV; Open Geographic Information;

Resumo

Com a redução dos custos de fabrico e as melhorias na sua robustez, controlo, e autonomia, Drones estão agora amplamente disponíveis para uso profissional e pessoal. Com toda esta tecnologia várias aplicações tais como entrega de produtos, vigilância, inspecções aéreas, e outros cenários onde drones podem ser utilizados, a navegação é um aspecto comum em todas e que requer manuseamento especializado. No entanto, a navegação automática para drones é ainda um tópico subdesenvolvido e com muitos desafios.

Para abordar este problema e ajudar pessoal não especializado, este trabalho centra-se num subproblema da navegação, o planeamento de trajectórias. Criámos uma ferramenta para planear trajectórias de baixa altitude, verificando restrições geográficas e legais, chamada DroneOpenTool. A ferramenta automatiza a recolha de informação geográfica, planeia uma solução quase óptima dada uma localização de início e objectivo assim como as desejadas restrições de voo e resolução, e produz uma trajectória de voo pronta a utilizar num sistema de controlo de drones. Para avaliar a ferramenta, testámos o desempenho de cada parte do sistema, comparando ambas as nossas abordagens de planeamento em vários cenários, e relacionámos a eficiência dos resultados com soluções terrestres.

Embora não tenhamos realizado experiências práticas, os resultados da nossa ferramenta estão moldados para serem aplicados em missões com drones do mundo real e alcançar resultados seguros e otimizados, como esperado. Com base nos resultados demonstrados, podemos afirmar que a abordagem adotada é viável dentro das expectativas iniciais, mas longe de um desempenho estável exigido por um sistema de controlo de missões, pelo menos sob a conceção seguida.

Palavras Chave

Drone; Planeamento de trajectorias; Algoritmo A*; Ferramenta de Voo; UAV; Informação geográfica;

Contents

List of Figures

List of Tables

Listings

Acronyms

ANAC	Portuguese National Authority of Civil Aviation
ANN	Artificial Neural Network
API	Application Program Interface
ATC	Air Traffic Control
ATZ	Aerodrome Traffic Zone
BVLOS	Beyond Visual Line Of Sight
CLI	Command-Line Interface
CPP	Coverage Path Planning
CRS	Coordinate Reference System
CTT	Portuguese National Post Office Services Provider
GIS	Geographic Information System
GPS	Global Positioning System
GNC	Guidance, Navigation and Control
GNSS	Global Navigation Satellite System
HTML	HyperText Markup Language
IFR	Instrument Flight Rules
IST	Instituto Superior Técnico
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
LIAN	Grid-based Angle-constrained Path Finding Algorithm
ONR	Obstructed Nodes Ratio
OSM	OpenStreetMaps
RPA	Remote Piloted Aerial vehicle

RTT	Rapidly-Exploring Random Trees
UAV	Unmanned Aerial Vehicle
US	United States
UTM	Unmanned Traffic Management
VFR	Visual Flight Rules
WGS	World Geodesic System

1

Introduction

Contents

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) commonly known as Drones have been around for more than two decades. Throughout the years, the drone market registered a substantial reduction in manufacturing costs as well as an enhancement of drone control systems, which ultimately expanded the applicability of drones to numerous scenarios [1].

From entertainment to professional employment, drones provide additional optical perspectives and visual reach as well as fast and effortless motion, supporting the accomplishment of specific tasks and reducing their execution costs. Some examples, mentioning only optical sensors, are aerial photography and video recording, information gathering for disaster management, inaccessible terrain geographic mapping, building and structures safety inspections, border control and surveillance, law enforcement, among several others.

According to recent market's research [2], there were 110 000+ commercial drones registered in the United States (US) in 2018, more than double compared to 2016. Perspectives point to this numbers increment in the following years, reaching 600 000 drones by 2022.

Recently, drones have been employed to express shipment and last-mile delivery, which often resembles a compelling use for logistics companies due to its encouraging estimated value. Consequently, the utilization of drones in the logistical industry has been widely studied to raise the technology to a standard [1].

Targeting last-mile delivery with drone technology might seem a far off logistical problem, but considering the growing number of low volume packages to deliver, enlarged by e-commerce, could reveal itself as this industry's next step. Accordingly, drones can improve the quality of service provided by logistical companies since these are fast, energy-efficient, and, once fully automated, can be easily deployed. Consequently, drones allow for different efficient strategies of the organization's delivery service, resulting in overall funds saving opportunities.

Drone delivery is one application that is gathering great interest from large organizations, such as Amazon [3], Post DHL [4], and Google [5]. Understanding the business advantages of this technology, in 2016, Amazon introduced a new last-mile delivery service employing drones, Amazon Prime Air, which revolutionized the delivery process panorama. The use of UAVs for delivering goods to costumers' doorsteps is becoming a new reality [1].

1.2 Problem Description

The use of drones for delivery services introduces several competitive advantages. The most notorious one is speed since drones have no constraints dealing with roads or traffic, and hence, they can deliver packages faster than a wheeled vehicle from a close-by storage location. Furthermore, drones can

traverse troublesome terrain (e.g., valleys, mountains, forests) with relative ease, in many cases, taking a much shorter route. Additionally, drones can commonly fly over terrestrial obstacles, such as water, farming crops, or rural areas with poor infrastructure, to deliver a package.

Delivery drones can also contribute to a better environmental impact when considering delivery traveling emissions since they often have electric motors. Moreover, the use of delivery drones over delivery trucks can considerably impact the atmosphere pollution levels and reduce traffic in urban areas.

Complementarily, if made autonomous, the non-existing need for a drone's intense remote control or supervision during the aerial delivery of a package results in lower demand for human labor, and if unveiled to its full capability, make available efficient 24-hour schedule delivery services.

Despite the benefits of using delivery drones in the logistical problem, such technology also has adversities. Specifically, one of the main drawbacks arises from the drones' battery capacity. The low battery capacity limits the drones' flight range and, when added a payload, can shorten the battery lifespan even further. Moreover, flight restricted zones exhibit another constraint to delivery service drones, as it can be an invisible obstacle to the drone delivery process. Another important aspect is safety and security, both of the drone and its transporting goods. Considering the variety of drone operation scenarios, the well-fare of the drone is a priority. Imprudent trajectories can infer damage to both, but also to third party entities. Hence, avoiding collisions at all costs is key to a successful planning mission. Finally, considering that the UAVs' flight law is still in development, this project will preemptively consider possible restrictions and challenges regarding regulation that may hinder drone flight planning.

Taking into account these difficulties is essential to optimize the route that the drones should follow in order to maximize the drones' range of operation (i.e., a greater area of delivery) as well as reducing the delivery times to the user.

Modeling the problem as a general search problem, the developed work will apply existent and established search algorithms to efficiently identify the most optimal flight path from one point to another, drawing an aerial trajectory for a drone in its mission, relying on geographical knowledge to warrant operational safety layer and boost delivery performance.

This thesis aims to enable the use of drones by a larger group of people to perform tasks where autonomous flight away from the line of sight is needed. Applications such as medicine delivery can be made more accessible to smaller distribution processes, as well as last-mile delivery and many other options for different individual purposes. It introduces a new high-level planning system relying on the integration of established search algorithms to identify and evaluate efficient drone flight trajectories, making it easy to use delivery drones in the real-world setting. The system requires the user to select the start and goal position, a set of constraints, and the level of precision for the mission to automatically plan a drone trajectory path, satisfying such requirements. The primary focus of the system regarding safety is avoiding geographical restraints such as buildings, lakes, and airport restricted flight zones,

and also, but not so easily defined, topological restrictions that arise from legal constraints such as high-population density avoidance, making the most out of all information provided by an open Geographic Information System (GIS).

Our work's ambition is to create a tool that assembles all the necessary parts for a hands-on drone mission planner, concerning safety and legal emerging issues, improving interactability with the users, and push the drone navigation technology towards fully autonomous task.

1.3 Objectives

This thesis aims to develop a software tool with regard to the following objectives:

- Automatic geographic information gathering from open and reliable sources;
- Compute the most efficient path for a drone to navigate, given two points (i.e., initial and final location) and a set of arbitrary rules;
- Out of the shelf multi-purposed system for drone mission planning;
- Creation of a demonstrative and interactive tool's interface;

The developed work directly applies to medical deliveries since medical supplies are essential to social well-being, particularly during pandemic crisis (2020-2021 , COVID-19).

1.4 Contributions

This work aims to provide a simple and intuitive assemble of tools for a drone-interested universe of users that desire to explore the capabilities of this technology by handling all the trajectory planning details down to a point-to-point traversable plan. Additionally, we focused on a growing application, drone-based last-mile delivery services, to motivate and lead our research.

With this purpose in mind, we compiled a literature review on topics related to existing drone solutions, geographical information usage in planning, and path planning approaches for drone navigation to assist our work's development.

We developed a hands-on UAVs missions planner system, using search-based planning methods, that can estimate optimal trajectories given a set of geographical restrictions relying on an open GIS and study the overall suitability of such approach.

Apace with this work's analysis and results, we provide all the research, code, and an online interactive demonstration with the expectation to contribute to this technology accessibility and utilization towards an automated future.

1.5 Document Outline

The organization of the document's remaining chapters is as follows. Under chapter 2, we will provide the background information required to understand the concepts and reasoning. In chapter 3, we analyze the related literature to get a better overview of the current state of the art for path planning, geolocated information systems, and data structures & efficiency areas. The step by step description of the system, as a full report of its functioning, is presented in chapter 4. In chapter 5, we present the gathered results and respective analyses. Lastly, the document discussion and critical design options in chapter 6, followed by the conclusion.

2

Background

Contents

2.1 Last-Mile delivery

Last-mile delivery corresponds to the last phase of the delivery service to the client or entity who ordered the package. This delivery service's phase is defined by the goods pickup and drop-off to the end customer. In other words, the definition of last-mile delivery is the transportation of the goods from a distribution hub to the final destination. Additionally, this delivery phase is characterized by the low volumes of goods and at high frequencies, revealing an expensive part of the supply chain due to the number of trips ends [6]. Thus, it's one of the most critical periods of the logistical problem. Any last-mile delivery solution's main objective is to deliver the items to the end-user, minimizing the costs for the retailer while maximizing the costumers' commodity.

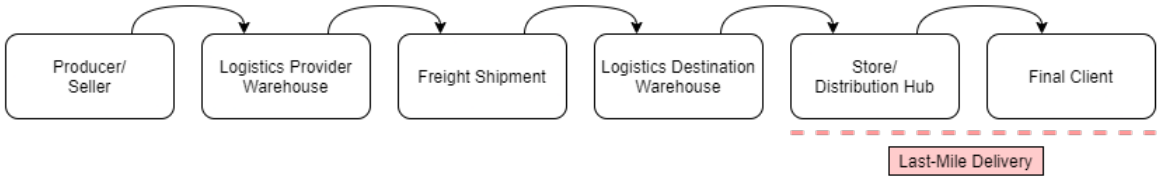


Figure 2.1: Last-Mile Delivery phase in a general Logistics Model.

Last-mile delivery characteristics merge delicately with the limitations of nowadays UAV technology. Relatively short trips with low volume packages are ideal conditions for drone applications. Considering such concerns for the context of our work, we explored a solution suited for planning last-mile delivery missions.

2.2 Drone Regulations

Automated drone regulations are yet to be established throughout the world. Nonetheless, there are already several remote-controlled drone regulations anticipating these agents' operation principal risks. The risk of uncontrolled flights in restricted airspace and therefore an areal collision's risk, and the drone falling to the ground possibility crashing into property or people are the two foremost concerns for this technology's legal regulation.

While initiatives like U-space [7] are emerging and eventually will take care of legal and operational restrictions of low-altitude flights for drones in a standardized way, as it aims to be an enabling framework for the industry, we yet still need to consider them for our tool's development.

The following table presents a summary of the legal framework implemented in Portugal territory by the competent entity, Portuguese National Authority of Civil Aviation (ANAC) [8]. The table 2.1 presents a current regulation's overview regarding UAV flights in Portugal, containing the necessary information

to explain the set of rules idealized as restrictions on path planning for our project. We describe these with detail further in chapter 4 where we apply zone constraints and try to address people concentrations regulation.

Table 2.1: Legal Framework Summary Table.

Zone Constraints:		
Military, Diplomatic, and Security Forces Facilities	Prohibited	Exception: Authorization of the entities representing the respective bodies without prejudice to compliance of the national regulation
Accident areas with in-course protection and rescue operations	Prohibited	Exception: Authorization of operations' commander under the conditions established in the regulations.
Restricted Area Class A: Instrument Flight Rules (IFR) flight operations only	Air Traffic Control (ATC) authorization required	Exception: Specifically authorized missions.
Restricted Area Class C: IFR and Visual Flight Rules (VFR) flights (separated communication)	ATC authorization required	Exception: Specifically authorized missions.
Restricted Area Class D: IFR and VFR flights	ATC authorization required	Exception: Specifically authorized missions.
Unrestricted Area Class G (General Rules)	No ATC authorization required	Max-Height: 120 meters (30 meters for toy aircraft)
Aerodrome vicinity	Authorization of the aerodrome responsible required (max-height: 120 meters)	Exception: No authorization required if the flight does not exceed the max-height of the highest natural or artificial obstacle within a radius of 75 meters centered on the aircraft.
Other Situations:		
Night flights	Require ANAC authorization	
BVLOS operations	Require ANAC authorization	
Flying above 120 meters of the surface	Require ANAC authorization	Except for flights within an Aerodrome Traffic Zone (ATZ), where the max-height is the corresponding maximum vertical limit of the respective aerodrome.
Flights above the specified heights of the national airports' areas	Require ANAC authorization	
Overflight people concentrations	Require ANAC authorization	By People concentrations, in this case, are groups with more than 12 people.
Operation of remotely piloted aircraft system with a maximum operational mass higher than 25 kg	Require ANAC authorization	
Flights within a 1 km radius of heliports with the following uses: a) Areal civil protection mission; b) Under management, command or responsibility of public entities; c) Medical emergency missions.	Require ANAC authorization	

2.3 Drone Energy Equation

Electric drones have their flight time-limited by their battery capacity. The energy consumption of a drone and it's attached weight generally defines the range of operability and can even be a decisive factor for the planned route feasibility.

Within the scope of our work, we used the following energy consumption model to define a heuristic for the informed planning algorithm, described further in the chapter, and derive some of the assumptions throughout the work's development.

Dorling et al. [9] derived an equation for the power consumed by a multirotor drone in hover as a function of its weight. Additionally, the authors suggest that the average power during hover is an upper

bound for the average power used during the whole flight, including the average energy consumed during takeoff and landing.

From the equation applied to single rotor helicopters in hover [10], that calculates the power P^* in Watts as a function of the thrust T in Newtons, fluid density of air ρ in kg/m^3, and the area ζ of the spinning blade disc in m^2, like

$$P^* = \frac{T^{\frac{3}{2}}}{\sqrt{2\rho\zeta}} \tag{2.1}$$

where the thrust $T = (W+m)g$, given the frame W in kg, the battery and payload weight m in kg, and gravity g in Newtons, Dorling et al. derived the equation for the power consumed by an n-rotor copter, assuming that all rotors share equally the same total mass $W + m$ of the copter as

$$P' = (W' + m')^{\frac{3}{2}} \sqrt{\frac{g^3}{2\rho\zeta}} \tag{2.2}$$

where $W' = W/n$ represents the frame's weight each rotor carries, and $m' = m/n$, for batteries and payload.

This derivation allows for the representation of the power consumed by all n rotors, powering the whole aircraft, as:

$$P = nP' = (W + m)^{\frac{3}{2}} \sqrt{\frac{g^3}{2\rho\zeta n}} \tag{2.3}$$

In the context of our work, we assume W as the total drone's weight alone, including battery, and m as the payload weight only. We used such assumption due to the presumption that the general use case for the developed work is using a non-customizable drone for delivery missions but rather the commercial market's existing options.

2.4 Geolocation Systems & World Geodesic System

Our work relies on geographic and geolocated data. Thus, in this section, we introduce some base concepts about geolocation systems and challenges on geolocated information handling.

A common standard of geolocation is the Global Positioning System (GPS) or nowadays Global Navigation Satellite System (GNSS). GNSS is a satellite-based radio navigation system that provides geo-localization (i.e., latitude, longitude, and height) to a receiver where there is an unobstructed line of sight to four or more GPS satellites [11].

The GPS coordinate itself is quite useful on its own. However, working with other systems to fill and the environment around the UAV can achieve significant objectives, although it may introduce fur-

ther challenges since the correspondence of coordinates between specific systems is not straightforward. With this issue in mind, we have to know the reference frame in order to match both coordinates. The Defense Mapping Agency, now National Geospatial-Intelligence Agency, was involved in the World Geodesic System (WGS)'s development since 1960. After several iterations, the WGS84 was born in 1984, contributing to a universal standard, still in use in most current geolocation systems [11].

By knowing the reference frame, we can use the coordinates across several systems or project the geographic points onto different ones. Consequently, calibrating the several tools to a reference frame is essential when applying transformations and calculating additional location points for tasks such as obstacle detection, described in section 4.3.1.

2.5 Planning Methods

To produce an output trajectory, we have to solve a path planning problem. This problem can be defined as finding a traversable path from a location to another. Approaches to path planning problems frequently use a method from one of these three categories: search-based, sampling-based, or combinatorial.

Our approach categorizes as search-based, the most established of the three in the field of robotics. The main idea behind such a method comes from the regularity of the structure used to represent the configuration space, commonly perceived as a grid. The start and goal locations within the configuration space are known, and the search runs on the grid to find a point-to-point path [12].

Dijkstra introduced the first algorithm to solve such problems with his breadth-first search [13] , in 1959. A short after, in 1968, Hart et al. introduced an algorithm that uses admissible heuristics to narrow the search and called it A* [14] . A* soon became popular in many computer science fields due to its completeness, optimality, and efficiency. Regarding pathfinding, the A* algorithm will find the solution as fast or faster than any other method, as long as its using the best-suited heuristic for the problem.

Although search-based planning is widely used, it carries certain limitations. The search grid fixed topology graph structure is simple to generate and maintain but is subject to resolution completeness, meaning that the resulting paths from the applied algorithms are only optimal at the resolution employed. Therefore, the trade-off between high-resolution grids, resulting in closer to optimal trajectories, and the required computational resources must be considered.

Additionally, we took inspiration from the second category, sampling-based, to formalize an alternative path planning solution, further described in the document, section 4.5.2. Instead of using an explicit representation of the environment, sampling-based algorithms rely on a collision checking module for providing information about the candidate trajectories' feasibility and connect a set of points sampled from the obstacle-free space to build a graph (roadmap) of traversable paths. The roadmap is then used

to construct the solution to the original motion-planning problem [15].

The last category, combinatorial path planning, tries to compute an exact representation of free space. The most common procedure is to build a visibility graph within the configuration space and use it as a roadmap to find an optimal solution. Although combinatorial algorithms may provide an elegant and practical solution to cases that have convenient properties (e.g., low dimensionality, convex models), they struggle if the set of obstacles is too broad, then a requirement of both completeness and practical solutions may be unreasonable [16]. For this reason, we discarded any possible implementation of it on our work.

2.6 Drone Delivery in Portugal

The following examples contribute to our cause in many different aspects. The first aspect, is the close relation to the developed work. Additionally, it represents a motivation in the development of this project as we can relate to the business opportunities in the market for this type of technology. Lastly, an objective, as we aim to make the examples part of what our system can generally do, specifically the mission trajectory planning component.

The examples are Remote Areas Medicine Delivery and Urban Area Autonomous Flight Demonstration, both Beyond Visual Line Of Sight (BVLOS). These examples are from the Portuguese company Connect-Robotics[1] success stories.

The first one, Remote Areas Medicine Delivery, corresponds to the transport and delivery of medicine in remote areas through the use of autonomous drones. The project aims to improve the delivery service of the pharmaceutic sector, specifically in remote regions, where the elder population has difficulties in accessing essential medicine. Moreover, we would like to mention that this project won the João Cordeiro - Pharmacy Innovation 2018 prize[2].

To make this service possible, the team customized a standard drone. In particular, the drone has a small container for transporting goods attached to its frame. Additionally, the drone carries both a phone, which corresponds to the processing element providing the GPS coordinates and sensors that measure the temperature and humidity levels.

The drone's estimated range of operation is up to 15km (including the returning journey), on an electric motored drone, with the cost of the transportation roughly around 2 cents, claims the head of the project in an interview [3] for the 13th Pharmacy National Congress.

The example addresses the delivery problem in remote areas, where people have no means to get essential products unless they travel to the closest village or city. Such issues do not correlate with

[1] Drone Services Portuguese Company: https://connect-robotics.com/

[2] Portuguese Pharmaceutical Innovation prize. 2018 edition winner: https://www.premiojoaocordeiro.pt/edicoes-anteriores/173-premio-joao-cordeiro-inovacao-em-farmacia-2018.html

[3] Interview video: https://youtu.be/GqSFBrjH31c

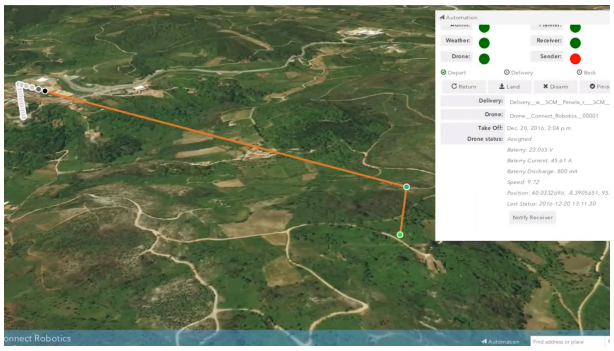


Figure 2.2: Connect Robotics UI for Remote Areas Medicine Delivery, taken from a demonstration video: youtu.be/lrwnAZTidhc .

the drone's flight trajectory, since there are no evident restraints, but more with the result of having such options delivering the goods to proximate location, preferably in a short period. An illustration presented in fig. 2.2, where we can witness the an interface used for the introduced scenario. The example conveys the importance and reliability of applications this technology can have, even in the less convoluting scenarios.

The second, Urban Area Autonomous Flight Demonstration, has similar mission objectives to the previous one, but a rather intricate problem scenario and environment. In this solution, drones were tested to transport and deliver small packages of mail for the Portuguese National Post Office Services Provider (CTT). The main focus was to assess the effectiveness of the mail delivery between an outpost and the main building of the post office with a single path approved by the competent authority, to be repeatedly traveled by a package delivery drone.

In this example, trajectory planning is a crucial factor due to the inferred circumstances of drone navigation through a dense urban environment, respectively, Lisbon city-enter. Recalling the law restrictions to the flight of Remote Piloted Aerial vehicle (RPA), the drone is not allowed to overfly more than 30 meters tall structures near a commercial airport. Thus, giving emphases to the need to avoid such obstacles while performing the mission trajectory planning in this kind of environment. Hence, our geographical information assisted drone trajectory planner should easily be able to recognize all the buildings in such high-density area of operation, and accordingly, draw a safe path around them, attaining the most optimal and efficient option.

3

Related Work

Contents

In this chapter, we will review relevant solutions for drone delivery and discuss both their practicality as well as aspects to improve. In the same way, we will use the presented solutions to inform the development of our approach.

In the following sections, we will review several state-of-the-art methods and tools that allow us to gather essential knowledge for the project and the corresponding implementation of an efficient drone path planning module.

3.1 Existent Planning and Drone Control Solutions

In a brief search through the internet[1], one can find numerous mobile applications, software tools, and business services for a large variety of different purposes for the use of drones. Imagery, mapping, and surveillance are the most common applications among the popular ones. Although these applications do not focus on path planning as their primary objective, most disclose the autonomy to handle drone navigation.

QGroundControl [17] and APM Planner [18] are open-source tools that offer a diverse range of utilities for full flight mission control for drones. Apart from all the information provided during the operation of the UAV, the user can define a set of way-points for the aerial vehicle to follow, allowing the application to take control of the drone navigation. Also, the ability to upload flight plans to be performed by a drone enables the software to accomplish fully autonomous drone flights. Due to its applications with drone technologies, these aforementioned tools have allowed several recent developments and studies around drone's subject [19, 20]. Despite being general drone control tools, they mostly require heavy user input to define autonomous flight plans, demanding the location of each individual way-point to be selected by the user, leading to simple and, in some cases, reckless paths.

The platform Airmap[2], offers to its users a path planning tool for UAV, in which the users can plan their remote flights. The main difference from the previously presented software is the compelling feature of real-world restrictions, namely the restricted flight areas globally, providing users the knowledge of the law restraints for each country, in the form of visual bounding boxes through the tool's interface. Additionally, a great advantage considered in this planning tool is the ability to check the flight restraints and facilitating digital flight authorizations with the competent authority. Although Airmaps' planning tool services immensely upgrade trajectory planning for drones, through the display, checking, and authorization mechanisms, they still don't provide an automatic trajectory mission planning for autonomous flights. Thus, possible users that cannot pilot a drone, or have sufficient knowledge to set-up detailed mission paths, are not able to reap the benefits of such systems.

[1] Popular link from Google search on Drone Applications: https://www.heliguy.com/blog/2019/03/12/15-drone-apps-to-help-you/
[2] Airmap drone solutions: https://www.airmap.com/

The system we propose to develop will similarly define the drone trajectory through a well-set collection of way-points, howbeit, it will automatically determine the drone path between start and goal objective points. Therefore, reducing the user input to a minimum and handling the rest, allowing for an autonomous definition of the drone's mission trajectory. Likewise, addressing legal authorization for flights, we aim to develop a system that recognizes such geofences[3] in UAV navigation, as well as any other arbitrary restriction options for user convenience.

3.2 Geographical Information Usage

Pursuing the idea of having detailed restrictions on the drone's flight trajectory, we must introduce a geographical data processing component that allows the system to recognize and handle such information.

V.T. Nguyen et al. [21] used real-world geographical data to reconstruct, in a virtual reality simulation, any specific environment for a drone flight simulator, with the purpose of training operators to pilot drones in realistic scenarios. Reportedly, the authors used OpenStreetMaps' GIS [22] to gather buildings data such as latitude and longitude coordinates, but also the buildings structure properties such as height and shape, to realistically map and model any form of building composed environments.

Yakovlev et al. in [23] describe the use of the same GIS to obtain a digital terrain chart for their experiments where the creation of such maps consisted of a squared binary grid plotted from a 1347 x 1347 meters map fragment of Moscow. The grid, with the values 0 and 1 and a static resolution of around 2.7 meters per grid cell, represented the permeable and impermeable positions, respectively, serving as their data structure for the planning algorithms search.

Similarly, we intend to represent an operational area for each drone mission where obstacles' positions are inferred by collecting their geographical information from suchlike external GISs. However, our system's design will make this task non-static, including the obstacle detection area definition in the mission planning process. Such an alternative will make the overall system more flexible, without the need for pre-computed areas input.

The extension potential of the previous techniques to a diverse spectrum of other obstacles and non-physical restraints that one can likewise gather in a GIS enables a large variety of 2-dimensional and 3-dimensional geographic constraints that our system could recognize. Thus, we aimed to accomplish the usage of such detailed information to aid our trajectory planner in achieving several levels of resolution of path planning and being able to assist a large number of different use cases.

[3] Geographic restricted areas that UAVs shouldn't fly, it's outer bound is considered a geographic fence

3.3 Path Planning Approaches for Drone Navigation

In this section, we will present different path planning alternatives for enhanced drone navigation. Navigation is the term used to guide a vehicle from a specific location to the desired goal, along a planned path in an environment characterized by terrain and a set of distinct elements, such as obstacles, milestones, way-points, and landmarks [24].

The problem of path planning can be split into two separate segments, planning in a static environment and planning in a dynamic environment. The characterization of the environment, as static or dynamic, is subject to the movement of elements that compose it. For example, if there's no change of position of the environment's objects during the action, it's characterized as static, otherwise as dynamic.

Notwithstanding, this project aims to solve the static and accessible path planning problem. Additionally, and using the concepts of Guidance, Navigation and Control (GNC) [25], we focus on solving the Guidance problem's part for low-altitude missions, and hence, we will review several congruent approaches from relevant literature.

3.3.1 Drone Routing Problems

Intelligent solutions for drone routing problems, typically addressed in surveillance and monitoring applications, can provide useful knowledge on what the constraints are, as well as detail some of the autonomous drone path planning issues.

Examples include single-vehicle routing problems for covering an area with a set of target locations described in [26] and the multiple-vehicles system that can self-organize and cooperate to ensure spatial and temporal coverage of specific targets over time in [27]. In both, the main objective is to maximize the coverage time of defined targets, leaving aside an essential aspect for our domain, the definition of the drones' path characteristics itself.

Although also addressed in [28], the authors call it Coverage Path Planning (CPP) problems, sharing the same constraint of limited battery capacity, where the aim is to minimize the consumption of the drone's energy while ensuring the maximum coverage of a given area. Additionally, this work considers a crucial feature for our work development, the obstacle awareness and avoidance of the environment structures.

The previous works provide a great notion of drones' capabilities and limitations during their autonomous missions. However, these works do not address the intricacies of the path definition for point-to-point navigation but rather roaming paths around the different objective locations or areas.

3.3.2 Path Planning Techniques

Further, turning our heads to the specific facet of point-to-point path planning, we will preview some path planning options for our system.

Behnke's [29] studied the use of A* algorithm for a local multiresolution path planning grid-based search. Grid-Based path planning methods use a decomposition of the configuration space into an array of cells where neighboring cells are connected by edges. The cost of each edge can be derived from the cost of the two adjacent cells. The representation of the obstacles is achieved by setting the value of the corresponding cell to a higher cost. Therefore, a minimal cost path can be found by searching such graph configuration. Furthermore, since each grid cell has a base cost, and consequently, the remaining path from any grid point to the target cannot be less than the Euclidean distance[4] to the target weighted by the base cost. Hence, the accumulated sum of the navigated the best path plus the value given by the heuristic can be applied to determine the expansion order[5].

Additionally, the author explored different grid resolutions, using a higher resolution for the space around the vehicle, denoting more details as the closer the agent is, and becoming lower as further away from it.

The author concluded that the A* algorithm outperforms other local multi-resolution path planning strategies by using fewer grid cells, and consequently, decreasing substantially the time necessary to obtain a solution.

Several studies including [30, 31] proposed the use of alternative algorithms, such as Genetic algorithms, which can perform path planning and optimize the trajectory of mobile systems. Such algorithms encode several parameters into computational chromosomes. These algorithms are particularly relevant for path planning since location points, trajectory segments, or complete routes can be transposed into the aforementioned parameters.

Genetic algorithms offer pseudo-random search techniques inspired by evolution theory, generating an initial population, where each individual represents a possible solution for the problem. Through the use of a set of genetic operations, the initial population evolves by retaining the best-fitted individuals and regenerating a refined population where such individuals are going to be the parents of the following generation.

Although genetic algorithms are robust in complex spaces and usually less computationally expensive than other techniques, these path planning algorithms may converge slowly to an optimal solution, which ultimately can affect the speed of navigation when applied to real-time problems.

Kala et al. [32] path plan the navigation of a mobile robot. In particular, the author uses an Artificial Neural Network (ANN) to find the path between an initial position and the goal position. To achieve this,

[4] As known as, "Areal Distance", or "Straight-line Distance"
[5] Order in which the algorithm chooses the next node to follow in search of the goal

the ANN learns from historical data.

This method consists of generating test cases and solving them with other path planning algorithms, in this specific case A* algorithm, to collect inputs and corresponding outputs, and hence, train the neural network. Thus, with enough training data, we can establish a neural network to predict an optimal path to the goal given any position.

Although the training of the neural network is a time-consuming process, this solution offers swift responses after the training, and consequently, provides promising results.

In a compiled comparison of the previous options, the A* algorithm outperforms the remaining algorithms in terms of optimality and efficiency when the search space is relatively small. Besides, genetic strategies would have difficulties in maze-like situations due to their random behavior. Notwithstanding, the use of ANNs can be beneficial when the problem has a large input size [32].

Xia et al. in [33] demonstrated, through simulation, the applicability of the A* Algorithm in UAV flights. The paper presents a modern warfare theme and aims to improve low altitude penetration effectiveness of UAVs used in combat. As reported, the authors built a model of the terrain complex surface, using a smoothing surface algorithm, and applied the A* algorithm to plan and avoid threat factors of flying above a detectable altitude in a martial context, such as mountain peaks and other terrain deformations. The achieved results are similar to the ones we expect to deliver with the proposed path planner module. Alternatively, we will use geographic data to infer the mission's environment in search of restrictive obstacles instead, not considering the altitude or the terrain elevation.

Chatterjee and Reza in [34], explored the option of using Rapidly-Exploring Random Trees (RTT) [35] to solve the dynamism of continuous changes to pre-planned mission trajectories in congested airspace locations, such as city centers. Their solution has many similarities to our work, and although it lacked some performance results, it might be considered for future optimizations to our planning strategy.

Yakovlev et al. in [23] presented a technique that involved decomposing a squared area in a grid of binary values to perform automatic path planning, focusing on the constrained flight dynamics of the drone. Utilizing available terrain maps, the authors classified the grid-like environment as obstructed and non-obstructed and applied a set of path planning algorithms, including A*, WA* (weighted-A*), R* and their developed Grid-based Angle-constrained Path Finding Algorithm (LIAN), to compute paths for drones. This work's main conclusion was that the designed algorithm works and performs within the expected result range, with the advantage of incorporation some of the drones' flight dynamics constraints. All this research work's content is a structured baseline for our work's development, as we aim to create a system that solves the same problem. Nonetheless, we considered a slightly different approach, where we account for a part of the flight dynamics' constraints in the environment setting to allow for more general and modular solutions.

In [36], the authors analyze how a drone can navigate through high-density urban areas with the

support of several sensors extra GPS, due to degraded signal or unavailable sensor data in such areas. The paper presents a detailed explanation of how each data sensor contributes to the navigation in this type of environment. Unlike the above paragraphs, this work does not supply direct implementation options for our project but adds useful knowledge for further developments in this technology.

4

Implementation

Contents

The developed work aims to establish a system to plan drone missions and aid users in drawing the best possible paths for their tasks in any environment using open-access geographical data. Therefore, this chapter describes the whole system implementation, starting with the architecture diagram and modular explanation, followed by its mathematical formulation. Then, we show how it gathers and processes the required geographical information and how it does the data conversion into a suitable data structure for the planning task. Further, it's introduced both the online and offline versions of the planning system, presenting detailed information on their differences and how they work. Additionally, it's bestowed an overview of output options for several system's use cases and web-application interface.

4.1 System's Architecture

In the present section, we describe our system's procedures and structure (fig. 4.1), detailing each module's purpose and contribution to the system's principal tasks.

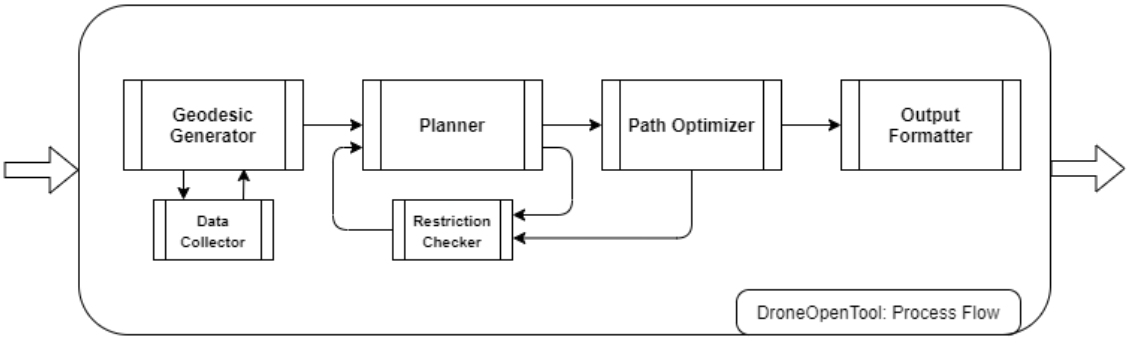


Figure 4.1: System's architecture modular diagram.

Geodesic Generator & Data Collector. The Geodesic Generator is the first process to engage upon the user input parametrization. It manages and formats the system's input and draws the rectangular points' mesh area that the system uses to perform the *environmental information assembling* task. This module works closely with an essential module called Data Collector. The Data Collector module is a fundamental part of the project. It's responsible for gathering all the necessary information from the Application Program Interfaces (APIs) and formatting the data to the path planning module needs. More specifically, through the input GPS coordinates of both the initial and goal location, the geodesic generator transforms them into their projection to calculate the adjacent points, for example, a grid-like mesh of point locations around the area at hand. The data collector then evaluates the best fit query to gather all the geographic objects set as restrictions.

Planner. The Planner is the second big module of the path planning process. The module functions simultaneously with the Restriction Checker module since all the paths generated need to be examined

to guarantee that the trajectories do not break any restrictions. The Planner takes the treated data from the Geodesic Generator and finishes the last part of the *environmental information assembling* task, requesting the restriction checker module for all the points obstructed by restrictions. Hereafter, it designates one of the developed methods for path planning, *structuring the data* accordingly to a graph and performing the respective *search for the optimal path*.

Restriction Checker. The restriction checker aids the whole path planning process, ensuring the results' integrity and validity. This module employs a collection of methods to validate the missions' restrictions compliance. From point inclusion tests to restricted points filtration, this module is indispensable to the system's functioning. Moreover, since the base restrictions are either area-related, this module can be replaced in the future by a communication module to a remote component such as an Unmanned Traffic Management (UTM) system, as in [37].

Optimizer. The efficiency filter module, called optimizer, takes the planner's resulting path and applies diverse methods to enhance the system's final output. For example, excessive point filtration, decreasing the straight line segments with more than two co-linear waypoints, or path smoothing techniques to eliminate sharp trajectory deviations.

Output Formatter. The Output Formatter is the last step of the system. Its main objective is to output either a generalized path plan to be interpreted by the drones' system or other customize detailed path outputs, such as a point per point overview of the mission, for particular hardware and visualization tools.

4.2 Mathematical Formulation

The system's principal task, mission trajectory planning, is conducted by the following mathematical formulation:

Given a start and goal points, s and g, generate a rectangular area M in a 2D Euclidean space of dimensions $d(s_{ij}, g_{ij})$ x $d(s_{ij}, g_{ij})/2$, expressible by non-intersecting union two sets M_{free}(free location points) and M_{restr}(restricted location points) with $s \in M_{free}$ and $g \in M_{free}$. Then, the formulation of the sets of restrictions represented by $M_{restr} = \{M_{res1}, M_{res2}, ..., M_{resN}\}$, $M_{res.i} = \{obs_1, obs_2, ..., obs_N\}$ and $obs_i = \{p_{i1}, p_{i2}, ..., p_{iMi}\}$ where the *i*th obstacle has a form of a polygon determined by the set of nodes $p_{ij} = (x_{ij}, y_{ij}) \in M$. The drawn trajectory tr should connect s with g so that: (1) all points considered for tr will be permeable and lie in M; (2) each node of tr contains only the points in the M_{free}; (3) the final path should account for any additional constraints within M_{free}.

4.3 Environmental Information Assembling

Environmental Information Assembling task is one of the most resource-consuming processes of the system, yet one of the most important. It is performed mainly by the data collector and restriction checker modules in combination to provide the planner the working structure to apply the planning algorithms.

4.3.1 Obstacle Detection Area

Given a start and goal positions, representing the take-off point and the delivery point for the drone mission, the data collector module starts to build a collection of points in between the two locations and relative to the orientation of its direct path, in a straight line. This collection of pinpoint coordinates can acquire any suitable shape for the problem at hand.

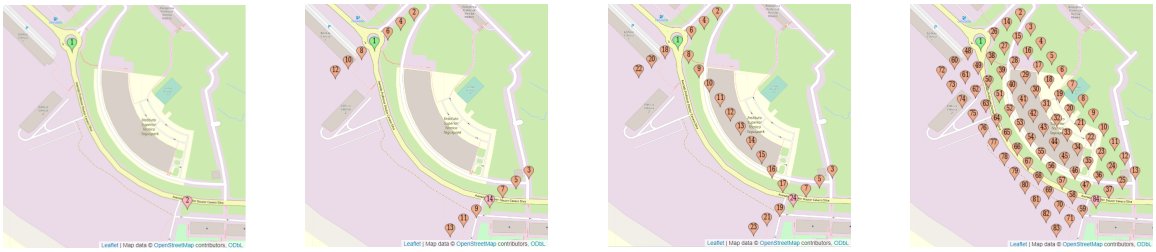
Within the developed work, the system applies a rectangular shape grid with the two points (Start & Goal) comprehended in the smaller edges of the rectangular area formed by equidistant coordinate points was suitable for the type of trajectories tested and analyzed further in the document.

Figure 4.2: Example of a pinpoint collection for environmental information assembling.

The points' collection is independent of the geography of the environment, serving solely as a template for the graph structure needed for the planning process, focusing on an equidistant filling of the area for the forward obstacle detection.

The process uses geodesic transformations of the GPS coordinates to be able to calculate orientated distances. The system uses a library named Pyproj [38], a python interface to PROJ, a geospatial coordinate transformation software to handle Coordinate Reference System (CRS) cartographic projections and geodesic transformations [39].

The principal use of this library in the system is to calculate the distance between two GPS coordinates. For example, one cannot tell the distance between Torre de Belém and Padrão dos Descobrimentos, in meters without recurring to a geodesic tool to aid that task[1]. Although it seems simple, the system uses this type of measurements to be able to construct the whole grid-like pinpoint collection, and further, be able to survey the surrounding environment.



(a) 1st - Start & Goal locations setting.

(b) 2nd - Start/Goal parallel bounds formation.

(c) 3rd - Linear points filling example.

(d) 4th - Finalized pinpoint collection.

Figure 4.3: The different stages of the pinpoint collection construction for environmental information assembling.

1. First, the system measures the straight line distance between Start and Goal points; 2. Secondly, the system starts to define equidistant points in the lines perpendicular to the straight line between Start and Goal Points, both with half of the length of the straight line between Start and Goal, with both positions in the intersections points; 3. Ultimately forming an capital 'I' with Start and Goal on each end of the lengthier line and in the middle of the two perpendicular lines; 4. Lastly, the correspondent points in the perpendicular line of the 'Start' position connect with a pinpoint straight line to the coordinate points of the perpendicular 'Goal' line, including nonetheless the pinpoint straight line between Start and Goal defined coordinates;

Although this pinpoint collection contains all the coordinates the system needs to plan the trajectory, this collection has no structure. Thus, the system stores all the pinpoints in an organized way to be able to translate all the coordinate points into a structured grid-graph with no misplaced information.

Moreover, the whole process of building this collection of pinpoints accounts and scales relatively to one important input variable called granularity. Granularity is the number of points that one wants to fit in the straight line that connects start to goal, as well as all the consequent parallel lines. Thus, this single variable influences the point-to-point distance, meaning that increasing the granularity value will reduce the distance between pinpoint coordinates.

Although it seems desirable to have a great granularity to have precise paths, further in the analysis chapter chapter 6, it's followed with considerations on this point.

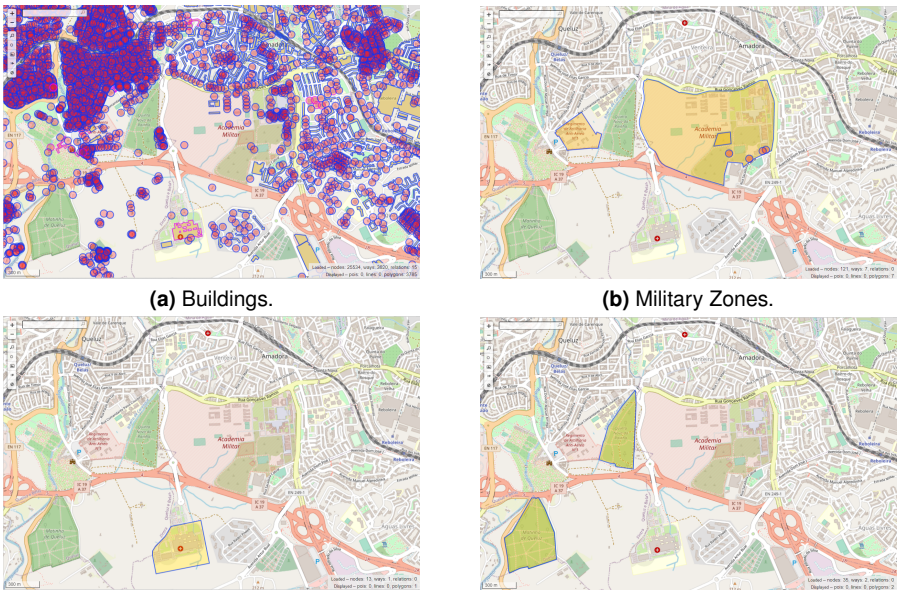
[1] Torre de Belém:(38.691889460164866, -9.21589003972094); Padrão dos Descobrimentos: (38.693795263086024, -9.205796627939964); Distance between the two coordinates: $\approx 920m$

4.3.2 Geographical information

The system needs access to a reliable and easy to use geographical system that allows safe, efficient, and legal path planning. GISs can contain different types of data, from objects, such as roads and infrastructures, to more abstract details, such as population-relation details (e.g., population-levels). At a basic level, a GIS acts as a simple map. Nevertheless, in each information layer, one can get various insights and data visualization solutions for different purposes. Thus a GIS allows the exploration of sophisticated spatial questions that one cannot answer with a traditional map [40].

However, the data present in a GIS is usually too dense for a single task or application to digest. Hence, it's necessary to filter out the relevant features by selecting a group of conditions that suit the problem at hand. One can extract feature information in GIS's layers by selecting criteria that satisfy a particular requirement. For example, if one needs to find and map all the technology stores in the city, would perform a query with the following keywords: building, technology, store, business.

OpenStreetMaps (OSM) [22] is a user-generated mapping project that emerged from the need for an open-access system where anybody can get free and reliable geographic information. This GIS allows for querying large amounts of data through a structured language, Overpass QL, with easy composition through a friendly Python API, Overpy. In this project case, it queries any restriction to the drone's navigation, for example, buildings, airports, military areas, water-spots, and woods. Such features can be previewed in fig. 4.4.



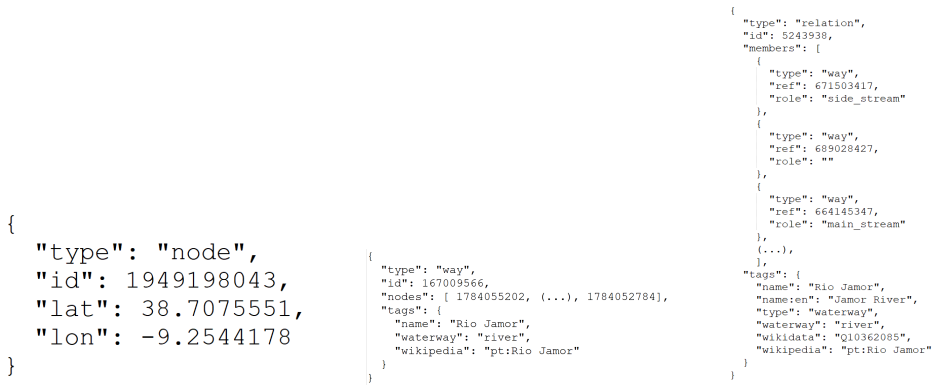
(a) Buildings. **(b)** Military Zones.

(c) Hospital Areas. **(d)** Woods and Forests.

Figure 4.4: Examples of different queried features from OSM's API.

Usually, and if no errors occur, the returning response to the queries should be a structured collection of nodes, ways, and relations. These are specific data structures of OSM used to represent different

complexities of geographical information. Each of these data structures represents a group of the previous one, meaning nodes are the base data structure and represent a single coordinate point. Following the logic, ways group a set of nodes to form a line or closed line and are used to draw roads and rivers or simple shapes like buildings outlines. Lastly, relations group a set of ways, making it possible to represent areas with gaps or complex formations of the several possible objects in GIS layers. All these structures can be visualize by the examples in fig. 4.5.



(a) Node. **(b)** Way. **(c)** Relation.

Figure 4.5: The different structures of OSM's query results.

Although the data structures seem easy to understand, for data transmission purposes, the more complex objects don't englobe all the related parts. For example, relations do not embody the whole structure of the ways that compose them but rather their ids, making the overall query answer lighter.

Nevertheless, as all the data objects have a unique identifier, they relate to each other by passing references in the case of one being part of the other, for example, a node being part of a way and consequently being also part of a relation. It's important to note that only nodes have fixed coordinate points, so it is required to perambulate the several levels of data structure to get the precise geographical coordinates of each object.

After getting the results from the API, it is stored in the system the geographic information of the queried objects, and it advances in the environmental information assembling process.

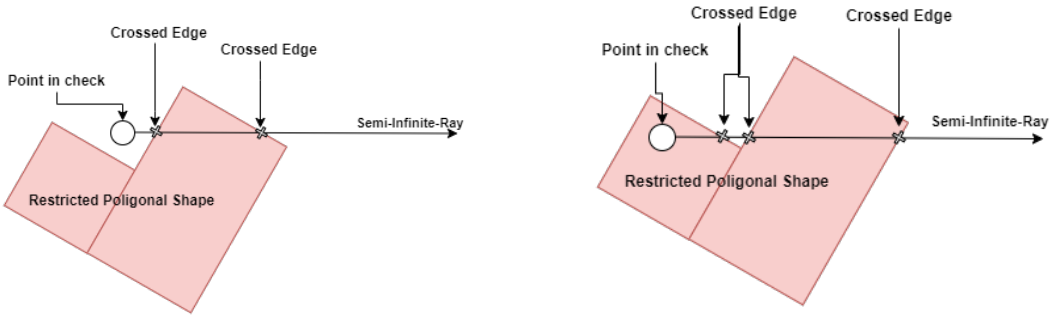
4.3.3 Obstacle filtration

Furthermore, still in the environmental information assembling task, the system advances to an essential process called obstacle filtering.

Obstacle filtering is a task performed by the 'Restriction Checker' module and is the primary GIS data processing task. The problem this task faces is to say if a given coordinate is an obstacle or not, meaning that it returns if that position is clear to overfly or there's a presence of a restrictive obstruction to the drone path. Although simple to explain, this task alone handles a large portion of the system

complexity as the numerous coordinate points exponentially increase regarding the granularity of the problem at hand.

Additionally, as the GIS's data does not provide this information directly, the system has to compute a point inclusion function to determine if any of the pinpoints in the previously calculated collection is inside of any of the obstacles or restriction outline gathered through the GIS API. As the results of the API usually are shapes defined by a closed line, the process transpires to be a point inclusion problem, in which case, given a point and a list of edges, the system must detect if it is inside or outside of a restricted area.



(a) Point outside the polygon. **(b)** Point inside the polygon.

Figure 4.6: Illustration of the PNPoly Algorithm functioning.

Figure 4.7: Example of a pinpoint collection with all the restricted points identified.

The system's implemented code function is an adaptation of the PNPOLY from W. Randolph Franklin [41], a method based on the Jordan Curve Theorem [42]. This method runs a semi-infinite horizontal ray out from the testing point, counting how many edges it crosses. At each crossing, the ray switches

between inside and outside, as shown on fig. 4.6. Thus, we can distinguish if any pinpoint is suitable for the drone's trajectories, as the system can detect if they are inside any restricted area, as demonstrated in fig. 4.7.

4.4 Structuring data

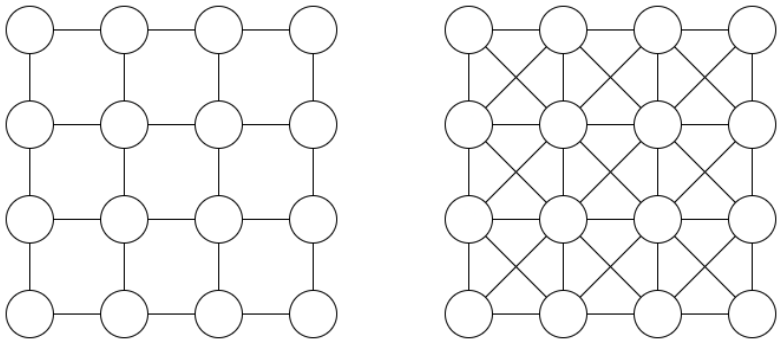
Data structuring is responsible for the performance and good-behavior of all the systems. The main concern in this phase is to structure all the gathered and inferred information of the environment in a single and suitable data structure to be traversed by the search and planning algorithms.

4.4.1 Graph Creation

Onwards to the main functionality of the system, there's an essential task called Graph Creation. The graph creation process, performed by the Planner module, starts when the information gathering and object filtration processes finish.

As the shape of the collected information area is rectangular, and its points are equally distant from each other, the system creates a grid-graph with the same dimensions as the shaped data collection and, further, starts to populate the nodes with the coordinate information of every point.

Resorting to a specialized graphs library, NetworkX [43], the system maps the previous unstructured collection of coordinate points into a structured two-dimensional grid-graph. As the library provides a grid-graph structure, the implementation within the system was straightforward, needing just a few adjustments on the neighbor nodes. Since the standard grid-graph considered only the four closest neighbors, as in fig. 4.8a, the system received an extended version where the implementation accommodates the eight adjacent nodes, adding the diagonal neighbors, as in fig. 4.8b.



(a) NetworkX's two-dimensional Grid Graph representation.

(b) The system extended version of the NetworkX's Grid-Graph.

Figure 4.8: Illustration of the changes made to the original Grid-Graph Structure.

This difference makes it possible for the planning algorithm to make diagonal pathing choices in the grid in just one step, making the trajectories more fluid, removing some excess of unnecessary points.

Further, the edges of the graph are updated accordingly, as illustrated in fig. 4.9, to the distance between neighbor nodes (representing the coordinate points) they connect, as closer nodes are (Up, Right, Down, Left), connected in bold edges to the center-grey node, and further neighbors (Up-Right, Down-Right, Down-Left, Up-Left), in thin edges.

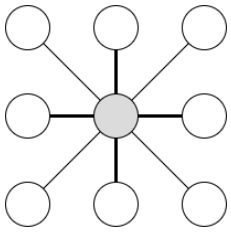
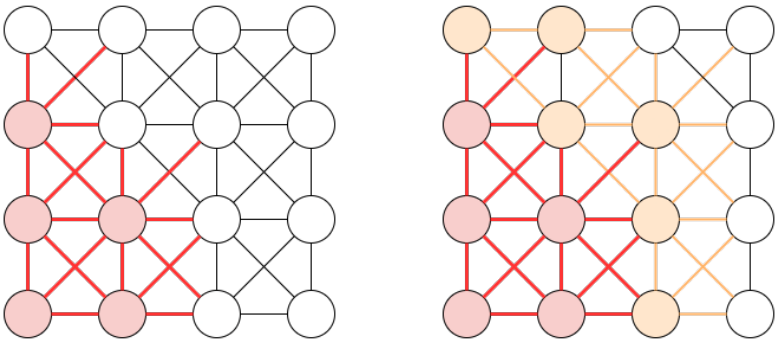


Figure 4.9: Distinction between closer and further neighbors.

Additionally, the system codifies the obstacle information from the obstacle filtration process into the edge cost. As the system knows which coordinate nodes are within an obstacle, while it's updating the cost of the edges, it certifies that the edges leading to obstructed nodes have a high-cost value, red bold edges in fig. 4.10a. Thus, once the planning algorithm comes across these high-cost edges, it's expected that it would try to avoid choosing them for its trajectory.



(a) Illustration of the obstacle's information.

(b) Illustration of the obstacle and safety margin layer (section 4.4.2).

Figure 4.10: Illustration of the coded information onto the Grid Graph data Structure.

4.4.2 Safety Margins

One of the first logical issues encountered by testing the system's output was the risky paths it would plan in some situations. As the algorithm would try to go around the obstacles optimally, the emerging problem of this behavior is that the trajectories would go too close to the obstacle limit.

The introduction of the concept of safety margins came to allow for any deviations or sudden events to the drone's navigation without putting the UAV and its payload at imminent risk. The safety margins act as restriction layers around the actual obstacles or restrictions, as shown in fig. 4.10b. Thus, through the cost updates of the graph edges, the system updates not only the edges from the closest nodes to the obstacle node but also from the nearest nodes' neighbors to them, recursively to the number of margin layer levels.

The weight applied to the safety margin affected edges' is relative to the distance to the obstacle node, thus being a fraction of the value assigned to edges closer to an obstacle node, expressed by the formula: $edge\ weight\ =\ \frac{obstacle\ weight}{margin\ level}$.

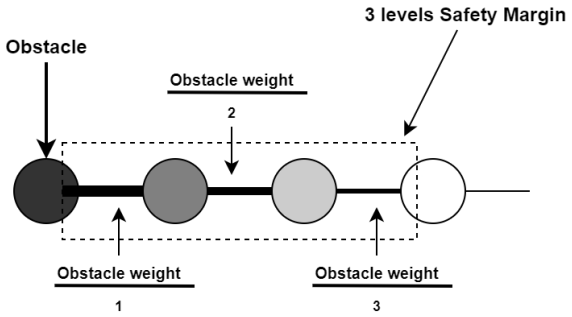


Figure 4.11: Image explanation of safety margins.

From the planner algorithm perspective, the closer it gets to an obstacle, the higher is the price of traversing the respective edges, ultimately making the algorithm choose a path that respects a safety margin around obstacle restrictions.

4.5 Trajectory Planning

At this stage, and with all the previous procedures complete, the system has all the required information and can start its principal shore: planning the path for the drone's mission.

There are two distinct approaches to perform the planning task in the system, both relying on the previously mentioned information gathering and structuring, but with different processes and information management that makes them more or less flexible depending on the problem at hand. Hence, one can describe the two approaches as a static and a dynamic version of the system's planner, and throughout the rest of the document, they will be called Offline and Online algorithms, respectively.

As the goal of the work is not to develop a new planning algorithm but rather provide a complete and intuitive system to help drone owners and services to plan their UAV flights, for both mentioned alternatives, the core planning algorithm used by the system is the established A* algorithm. Due to its efficiency, A* is one of the most popular search and planning algorithms. Further, as categorized as an

informed algorithm, A* bases its search choices on a heuristic, suiting our purpose since it's not hard to find one for our problem. The system heuristic relies on the euclidean distance between the current and target node since it's one of the most popular heuristics in path planning due to its admissible properties ensuring the optimality of the A* algorithm.

Hence, the system output can be as close as possible to an optimal path for the drones' missions, accounting for the imposed obstacle restrictions.

4.5.1 Offline Approach

The offline version of the system's planner does the complete computation of a given area's graph. Once computed, reusing it for multiple mission planning tasks within the same environment, without further computational expenses, is one of this approach's best advantages. Due to the overhead of information, this approach tends to be slower than its alternative, presented in the following section.

In its process, the offline approach goes through all the methodology steps, described in this chapter, creating a grid and marking the points within obstacles section 4.3, gathering all the obstacle data section 4.3.2, and graph creation as referred in section 4.4.

Once the previous steps are complete, the planner module applies the A* search method to be able to find the best possible path, considering the restrictions geographical information, the granularity of the way-points, and the level of the margin for the problem at hand. As the grid-graph for the offline approach is complete with all the information gathered by the system, the expectation of the performance of the planner are one-pass executions of the planning algorithm, outputting the optimal path for such conditions. A graphic representation of the process follows in fig. 4.12.

(a) Highlighted restrictions on the mission's area.

(b) Illustrated planning algorithm's result.

Figure 4.12: Offline Approach Example.

```
""" Pseudo-Code for Offline Approach """
def offline_approach(start, goal, granularity, margin_lvl, restrictions, area):
    ### Create the environment analysis structure
    missionGrid = pinpointGrid(start, goal, granularity)
    if area is None:
        area = automaticAreaAdjust(missionGrid)
    ### Handle Restrictions
    restriction_data =  fetchRestrictionsData(restrictions, area)
    missionGrid = applyRestrictions(missionGrid, restrictrion_data)
    ### Generate a graph
    graph = constructGridGraph(missionGrid, restriction_data, margin_lvl)
    ### Utility for reuse
    saveGraph(graph)
    ### Plan & Optimize trajectory
    path = optimize(a_star_path(missionGrid, graph))
    return path
```

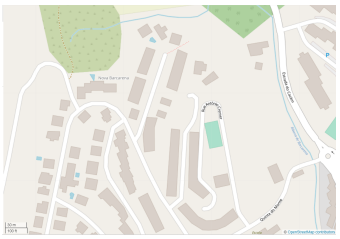
Listing 4.1: Offline Approach pseudo-code

4.5.2 Online Approach

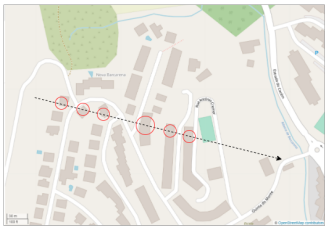
The online version it's characterized by the dynamic completion of the grid-graph structure, avoiding the expensive computation of the whole graph to streamline the planning process. Although it's faster than the previous alternative, it cannot be reused as the graph's configuration varies on the selected start and goal positions.

This approach gathers the obstacle data, as in section 4.3.2, but post-pones the obstacle-filtration task section 4.3, avoiding this expensive task to conserve time, executing the graph creation method section 4.4 with no information on what coordinate points represent an obstructed location. Thus, the grid-graph will have only the structured geographical coordinate points collection of the available way-points, with no restrictions at all.

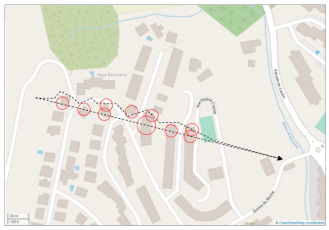
As the planning algorithm chooses paths between the starting position and the goal position, at each iteration, the chosen path passes through a function of the restriction checker, ensuring that all the geographical nodes of the planned trajectory do not correspond to obstacle restricted positions. Additionally, when this function finds one or more obstacle nodes, it updates all the identified graph edges with the respective values to enhance the choices of the planner for the next iteration. Although this process can take several iterations until it finds a valid path, the leading expectation for the online method is to save some computational resources and time since it only processes the minimal amount of nodes to find a valid path. The step-by-step illustration of this process is presented in fig. 4.13.



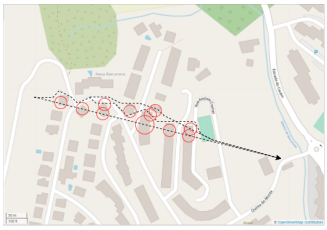
(a) Base-map of the Graph's location.



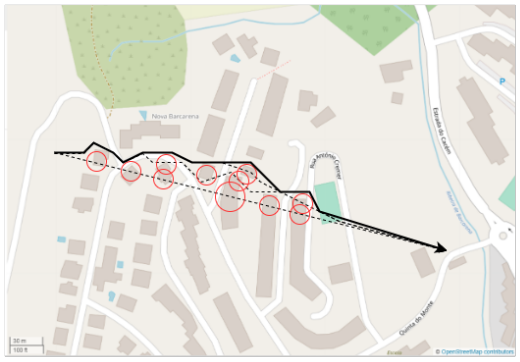
(b) First iteration of the Online method.



(c) Second iteration of the Online method.



(d) Third iteration of the Online method.



(e) Illustrated planning algorithm's navigable solution.

Figure 4.13: Online Approach Example.

```
""" Pseudo-Code for Online Approach """
def online_approach(start, goal, granularity, margin_lvl, restrictions):
    ### Create the environment analysis structure
    missionGrid = pinpointGrid(start, goal, granularity)
    area = automaticAreaAdjust(missionGrid)
    ### Fetch Restrictions
    restriction_data =  fetchRestrictionsData(restrictions, area)
    ### Generate a graph
    graph = constructGridGraph(missionGrid, margin_lvl)
    ### Plan Trajectory
    path = a_star_path(missionGrid, graph)
    while (not isClean(path)):
        # Iterative graph completion
        restrictedLocations = cleanTrajectory(path, restrictions)
        graph.updateWeights(restrictedLocations)
        # Re-plan Trajectory
        path = a_star_path(missionGrid, graph)
    ### Optimize Trajectory
    path = optimize(path)
    return path
```

Listing 4.2: Online Approach pseudo-code

The main difference between both approaches lies in the graph's construction. In the Online method, obstacle weights and margins update dynamically when the algorithm comes across the restrained nodes in the planning task. Meanwhile, in the Offline's alternative, this update is static, occurring in the graph's creation phase. Although their differences, both should be able to find a valid and sound path, if it exists, for any mission, concerning a set of arbitrary geographical restrictions to achieve the best performing trajectory for the UAV.

4.6 Output Optimization

Output optimization tasks are address by the Path Optimizer module. In this, the system treats the planner's output trajectory, through excess information cleaning and path smoothing functions, which is then rechecked after these made adjustments to the pathing to ensure correctness. The present section addresses the system output's clarity, and complexity, as well as a control problem, the sharp angles.

4.6.1 Trajectory Cleaning

The number of geographic points in a trajectory is related to the problem's resolution, meaning that higher resolution problems generally result in a more accurate path hence higher node count. Trajectory cleaning then becomes valuable through eliminating excessive information on path sections where, for example, the UAV is supposed to fly in a straight line.

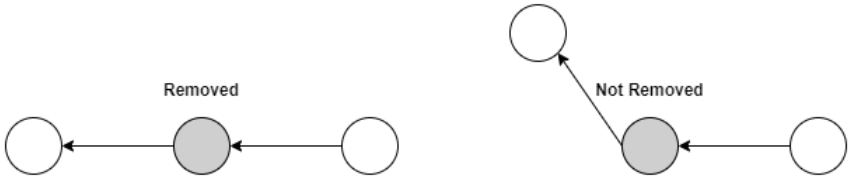


Figure 4.14: Image explanation of colinear points removal.

The method takes the vector orientation from the previous and the next node to evaluate if the geographic point in question is co-linear, as illustrated in fig. 4.14.

Thus, if the geographic point belongs to a trajectory's straight-line section, it's removed as an excess of information to optimize the overall output's simplicity. With just a few iterations of this method, the optimizer module can detect most of the colinear points and reduce the output's number of nodes, as shown and analyzed in section 5.5.1.

4.6.2 Path Smoothing

The purpose of path smoothing is to remove abrupt changes in the planned trajectory. As the range of possible adjacent nodes is limited, the planning algorithm (A*) makes the decision to move to the next best position. Therefore, it can make trajectories that contain several sharp angles to move in one specific direction due to the displacement of the adjacent cells.

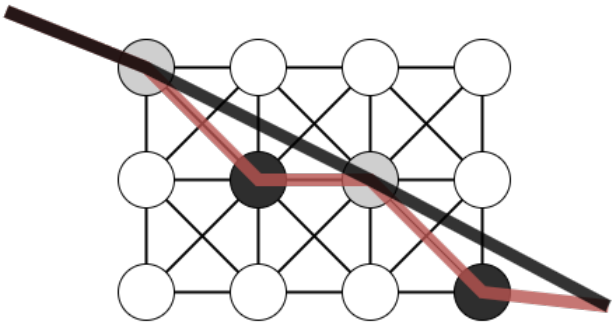


Figure 4.15: Image explanation of smoothing points removal, where the sharp is illustrated in red and smoothing method's resulting path in black.

We implemented a verification for these cases that detects when such situations happen and tries to smooth the trajectory removing the in-between nodes, represented as the dark-grey circles in fig. 4.15, and leaving the light-grey nodes.

The objective is to achieve smoother paths and mitigate several frequent changes of direction on particular trajectories' sections, enhancing the path's clarity and remove a significant portion of redundant nodes that define it, resulting in less complex trajectories, as shown further in section 5.5.2.

4.7 Output Options

The objective of the system is to provide a concise and hands-on open-source tool to aid individuals or organizational entities to plan UAVs' delivery missions. As the developed work does not aim to control a drone, neither has the appropriate navigation information to do so, this system's output needs to be generalized enough to be perceived by another system's input. From this premise, the system has a specific module to handle the final product of these interconnected subsystems, represented by the system modules, as detailed in section 4.1. Hence, both path visualization tools and drone navigation-control applications were considered and taken into account for the presented options.

A – GeoJson The GeoJson is a geospatial data interchange format based on JavaScript Object Notation (JSON) that became a standard in 2016 [44]. The first traces of this format started to be used in late 2006 and gain traction due to its simplistic notation and mild structure of features properties.

The foremost reason for choosing this format for one of the system's output was its interchangeability property based on JSON, one of the most used for structure and transmit data over the internet, with the specific composition for geospatial information.

B – HTML/JavaScript For ease of understanding what the system produces, it provides a static browser output, only requiring the most lightweight, flexible, and interactable JavaScript library Leaflet [45], which can be instantly visually analyzed and interacted with by the user as soon as the process terminates.

These web development languages serve as a base for a pleasant visual finishing result of the system in a simple HyperText Markup Language (HTML) file, ideal for previewing the output trajectory before engaging in a blind-mission with the UAV hardware.

C – KML Although not so friendly, the Keyhole Markup Language (KML) is a standard file format used to display geographic data, and it's the file format of the world-famous Google Earth software. As one of the most popular geographic file formats, its implementation, as an output option for the developed system, results in a broader scope of possible uses for this system, as providing mission plans for the most variant applications is one of the objectives already mentioned.

4.8 Web Interface

In addition to the standard Command-Line Interface (CLI), demonstrated in appendix A, and with the vast usability objective in mind, as our system's complementation, we designed a simple interactive demonstration in the form of a web interface for the tool. Such an interface allows for sample testing trajectories with fast and visual feedback to prove our system's utility and deployability.

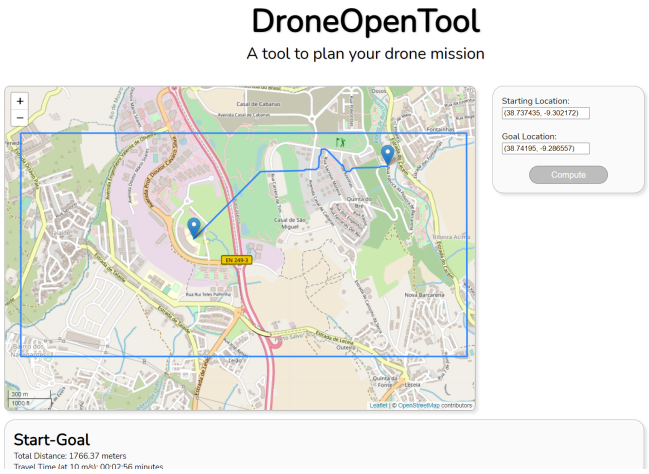


Figure 4.16: DroneOpenTool available web app.

We deployed a web app that integrates the software with a pleasant interface composed of: a visual map, a parameterization panel, and a details section.

In fig. 4.16, we have a map on the left that can shift around the globe to the location desired for the mission. This component provides an accurate representation of the environmental elements that the user can query through the GIS, such as buildings, landmarks, and different geographical or political areas. On the right, we got a parametrization panel where we can define all the system parameters required to submit and compute the best trajectory for the desired mission.

Specifically, the users can choose the values for parameters analyzed in section 5.2 and set the type of restrictions to avoid, as shown in fig. 4.17, to tune the system in their best interest and needs. The last component, the trajectory detail's section, shows the mission's information aiding the user to understand the planned path's properties. Information such as start to goal distance, the planned trajectory length, and estimated time and energy costs for a reference drone are displayed.

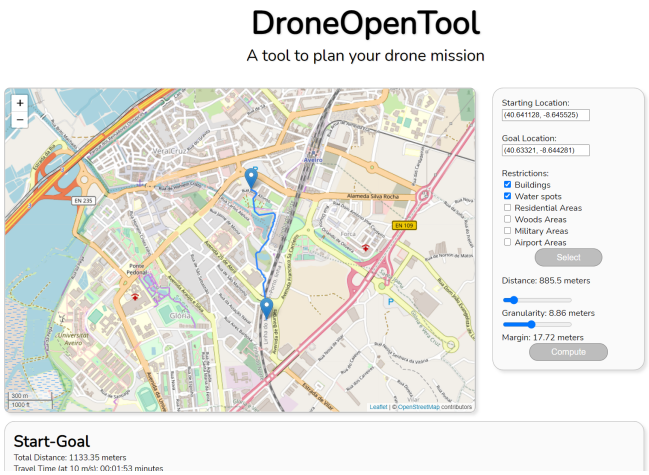


Figure 4.17: DroneOpenTool full-featured web app.

Although the online web app is just a demonstration of our system capabilities, it aims to provide the essential user-friendly features of an automatic trajectory planning system. Hence, the interface draws the trajectory on the graphical map and shows all its details for the users to quickly evaluate before uploading it to a drone navigation control system.

5

Results

Contents

This chapter presents the results produced by the system and their analysis. We considered houses as obstacles, even if other restrictions could be considered, as further depicted. Our choice was based partially on legal requirements that do not allow drones to travel on top of concentrations of people, so schools and offices are to be avoided. Also, drone usage in urban areas presents a higher risk due to environmental complexity, so we concentrate there. Throughout the following sections, we describe the targeted drone product, evaluate the availability of reliable information in OSM, how the different parameters influence the resulting drones' trajectories, the planner's operation methods, and run a series of system's computational complexity tests.

5.1 Drone Characteristics

The designed system does not rely on any drone's specific software or hardware. Although the project's development was motivated for drone delivery purposes, the results apply to any autonomous drone flight missions.

Our approach is drone agnostic, but we considered a reference drone compatible with many applications. A drone with the requirements of being easy to control and has enough payload for small delivery tasks is the DJI Inspire 2 [1]. The stated drone can flight with a payload of up to 1 kg, has a navigation accuracy of around 1.5 meters, and a capacity of 27 minutes of battery autonomy. Such specifications make this example excel in photo capture and cinematography, as it was built for, although it allows for different workloads such as small-package delivery. Further, considering control range and speed, the drone can sustain tasks in an operating radius of 7 Km, the maximum reach considered throughout the experiments.

5.2 System Parameterization

The present section describes the considered parameters during the system's development, introducing each one and providing a detailed explanation of how they influence the produced trajectories. Mission area definition, granularity and margin configurations, restraints choices, and which planning method to employ are the target for reflection and demonstration of the system's capabilities and limitations.

5.2.1 Area & Restrictions

The area definition gives the user the possibility of choosing the best suited operational sector for the problem at hand. The input parameter is a 4-tuple composed of the south-most latitude, west-most longitude, north-most latitude, and east-most longitude coordinates, forming a rectangular bounding

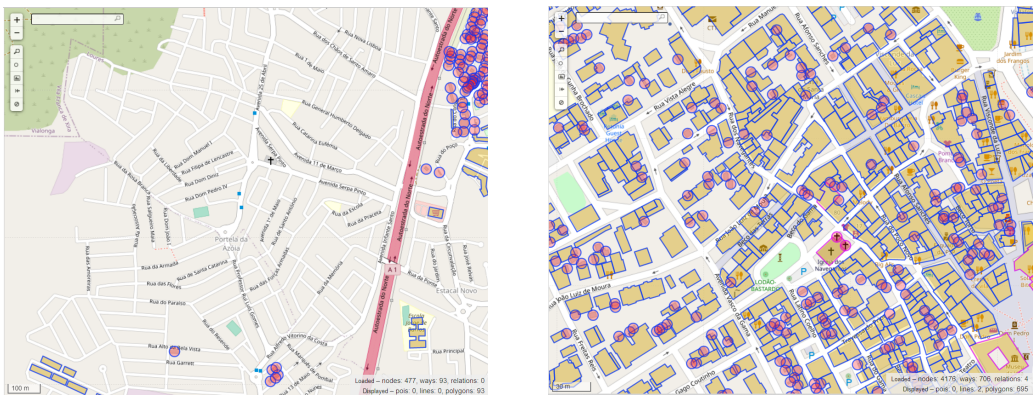
[1] Drone & detailed specifications DJI Inspire 2

box, as observed in the example in fig. 5.1. Further, this bounding box serves to query the GIS's API and fetch all the path planning restrictions. By default, if the user does not input any area, the considered area is the minimum rectangular enclosing the obstacle detection area (section 4.3.1), as depicted in section 5.2.1.B examples.

Figure 5.1: Example of a bounding box for the specific input: Area = (38.73487,-9.30590,38.73949,-9.29642).

5.2.1.A GIS Information Quality

The quality of information present in OSM is not heterogeneous in its literal globe of geographic data, as one can observe in fig. 5.2. One of the system liabilities is inherent to this parameter's definition. The user can define a valid area and, unfortunately, due to missing data of the GIS, get an invalid path. Such a problem is hard to detect preemptively and cannot be resolved by our system.



(a) Low building tags percentage area. **(b)** High building tags percentage area.

Figure 5.2: The two-ends of GIS's buildings information completeness status in Portugal.

Nonetheless, most cities have information about the majority of the geographic features annotated

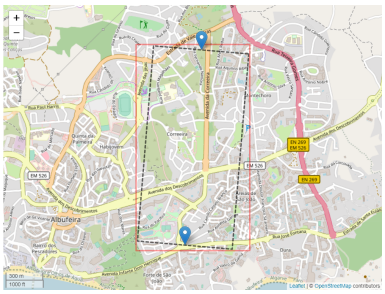
by the OSM contributors. Houses, airports, law enforcement and military areas, along with woods and lakes, wrap the generality of the restraints tested throughout the system's development, demonstrated in section 5.2.5. With this in mind, we can only advise the user to inspect the OSM's area information prior to the mission planning task to ensure sound results.

5.2.1.B Default Restriction Area Adjustment

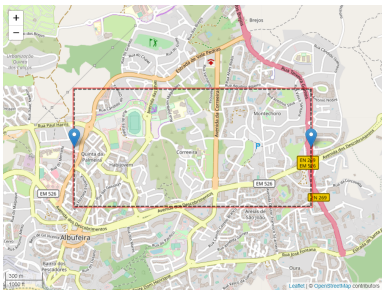
Addressing a slightly different issue related to the over-definition of the mission area, we introduced the Default Restriction Area adjustment.

The Default Restriction Area Adjustment is a feature that removes the area definition from the required arguments, meaning that a user does not have to specify the operational area of the drone mission, only the start and goal locations, for the system to evaluate the smallest surrounding area. In terms of complexity, the more obstacles the system has to process, the more expensive it is. Even though the area's size does not influence the number of restrictions directly, commonly, larger sectors can accommodate more restraints.

With this in mind, we tried to reduce the area for the restrictions query to the minimum information needed for completely covering the obstacle detection area. The following example, fig. 5.3, shows both the Obstacle Detection Area (dashed delimited area) and the Restriction Queried Area (solid red delimited area) in different mission setups. This feature reduces possible input errors while offering suitable automatic parametrization for the atypical user.



(a) Slight vertical deviation setup.



(b) Completely horizontal setup.

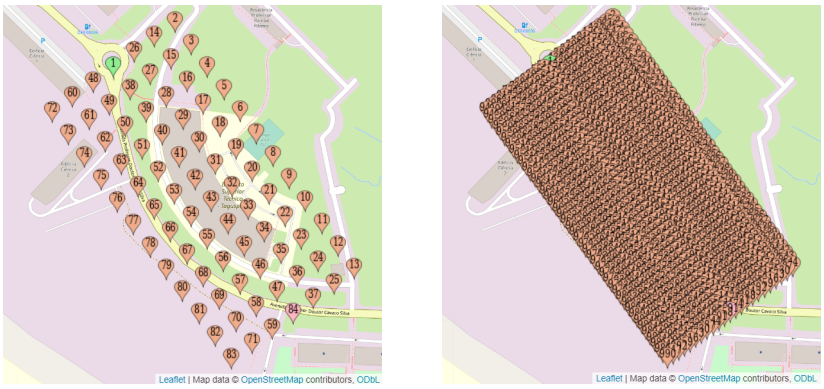
(c) Diagonal setup.

Figure 5.3: Automatic Restriction Queried Area's setup examples.

5.2.2 Path Resolution

As previously touched in section 4.3.1, the granularity widely impacts the definitive path. This parameter represents the number of points the user wants to fit in the straight line connecting Start to Goal.

The granularity, in conjunction with the distance between Start & Goal, estimates the path resolution. For instance, far apart objectives and a small granularity translate into a low path resolution, as the pinpointed nodes are distant from each other. In such cases, the probability of not detecting obstacles is higher but considerably enhances the solution's computation times. On the other hand, if computation time is not a capping consideration, one can specify more granularity to suitable amounts so the system can draw more meticulous paths.



(a) Point-to-Point Resolution: 29 meters. **(b)** Point-to-Point Resolution: 8 meters.

Figure 5.4: 2D Grid Resolution Comparison.

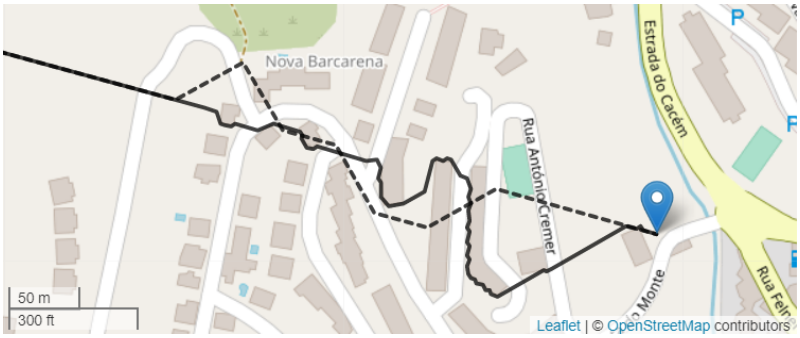
In the example in fig. 5.4, one can observe two different resolutions on a small area where the distance between Start & Goal is relatively short. The example in fig. 5.4b with around 1 000 points, has ten times the amount of nodes of fig. 5.4a. Such illustrated configurations are the backbone of all the path planning tasks to follow.

The fig. 5.5 examples show two paths around our campus. The drone starts on the pinned location on the left (Start) and must travel to the pinned location on the right (Goal). We consider either a grid with high resolution of 2 meters (solid line) or a low resolution of 38 meters (dashed line). The 2-meters' grid might seem excessive, considering the drone's dynamics, but our results show why it's needed.

Although the two trajectories are similar as a flight plan, in terms of obstacle avoidance there are relevant differences. In the final part of the path, a few buildings are "ignored" when the resolution is too low. According to fig. 5.5b, one can see the dashed example going over some of the buildings, as the granularity is too low and the way-points are scattered apart. The system is doomed to assume no obstacles in the path as the resolution is too low to detect the restrictions.

On the other hand, the high-resolution path (solid path), in fig. 5.5b, detects the buildings and manages to fly around all the obstacles, achieving a legitimate trajectory. This second comes with a massive

(a) Path from Instituto Superior Técnico (IST) Tagus to Location1.



(b) Path from IST Tagus to Location1 final part Close Up.

Figure 5.5: Path from IST Tagus to Location1 fig. 5.5a and its final part close up fig. 5.5b, with distance between way-points 38 meters (dashed path) and 2 meters (solid path).

trade-off in problem complexity and computational resources, namely computing memory and time, as the number of nodes is much larger.

From our experiments, we concluded that the problem's resolution should be defined carefully and accounting for the smaller obstacles the user expects to encounter in the problem's environment. Such-like definition of this parameter avoids two undesirable scenarios: the insufficient resolution, leading to faulty trajectory plans; and the area's overestimation, caused by the excessive redundant granularity.

5.2.3 Safety Margin

The feature described in this section was born due to the system's first results' analysis. Looking back to fig. 5.5b, one can notice the trajectory abutting some of the buildings' outline. This effect is better noticeable when the resolution is higher and becomes an issue in certain conditions.

This issue's reason is the combined goal of avoiding obstacles and aiming for the shortest path, which originates the circumvent trajectories. In the example of the solid path, from fig. 5.5b, the resolution is so high that the algorithm planned its best route very close to the obstacles, leaving no margin at all. Accounting for external errors, such as GPS accuracy, wind gusts, and hanging objects outside buildings and other relevant restrictions (street lighting, billboards, etc.), we realized that we should consider a safety margin around obstacles.

Further, as presented in the heretofore section 4.4.2, the safety margins are relative to the resolution, i.e., the margin cannot be smaller than the grids' point-to-point distance, and the higher this distance is, the greater is the applied margin. Likewise, smaller grid resolutions tend to demand less safety margin levels as the points are already distant from each other.

Figure 5.6: Path from IST Tagus to Location2, with distance between way-points: 1 meter with margin to buildings of 5 meters (dotted path); 2 meters with margin to buildings of 10 meters (dashed path); and, 5 meters with margin to buildings of 20 meters (solid path);.

In the fig. 5.6, one can see three trajectories obtained with different margin levels and resolutions. Starting with the higher-resolution path (dotted line), we can observe a highly detailed path with a barely perceptible safety margin, thus, representing a dangerous scenario for the UAV in case of any slight deviation in the drone's navigation.

The dashed line represents a version of the same plan with a decreased resolution, meaning a greater distance between neighbor nodes, given the same safety margin level. Hence, the margin becomes perceptible and, due to the new distancing factor, the planned trajectory takes a distinct route from the dotted path.

Additionally, we used a smaller resolution once again (solid line), achieving a satisfying margin, with a visible safe distance to the obstacles, without losing any valuable information about the geographical position of the restrictions, issue addressed previously in section 5.2.2. Note that in the examples, the safety margin levels used were 5, 5, and 4, respectively, i,e., margin level times the point-to-point distance of the nodes, due to the resolution effect mentioned above.

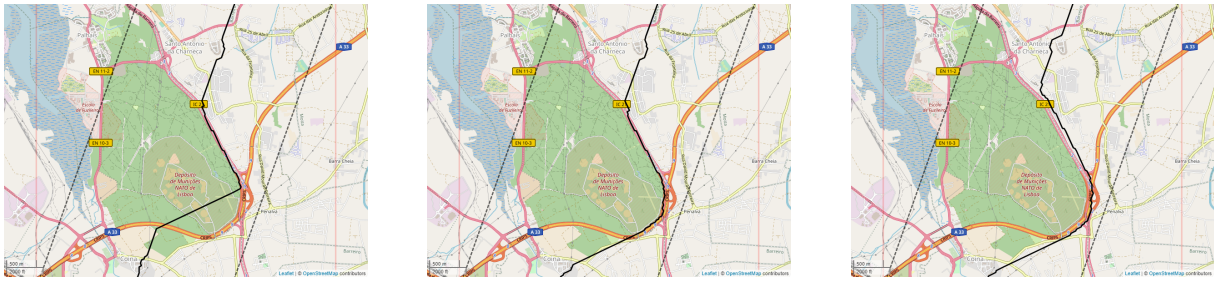
The results show that the system reacts to the input adjusting the trajectory as the safety margin requirements change. Consequently, the same mission can have disparate outcomes based solely on the margin's parameter level.

5.2.4 Obstacle Weight

In the first experiments with the safety margins and their relation with the obstacle weight, we notice some defective trajectories that we considered to test and analyze.

The weight values can have particular effects from ineffective to inflexible if the weight is too low or too high, respectively. A symptom of an undervalued obstacle weight is when the algorithm opts by a trajectory that overflies an obstacle rather than circumvent it. On the contrary, if there's a gross obstacle weight value, the algorithm might be too strict and discards perfectly sound trajectories within gaps between obstacles, covered by safety margins.

From such observations, it was clear that the algorithm is sensitive to different obstacle weight values. Adjusting the obstacle weight value is pivotal to avoid situations where the algorithm ignores some, or parts of, obstacles due to inadequate weight value, such as in fig. 5.7a, or when considering a flexibility trade-off in situations where it's preferable to overlook an optimal path over a high-valued safety margin restriction (or the inverse), depicted in fig. 5.7b and fig. 5.7c.



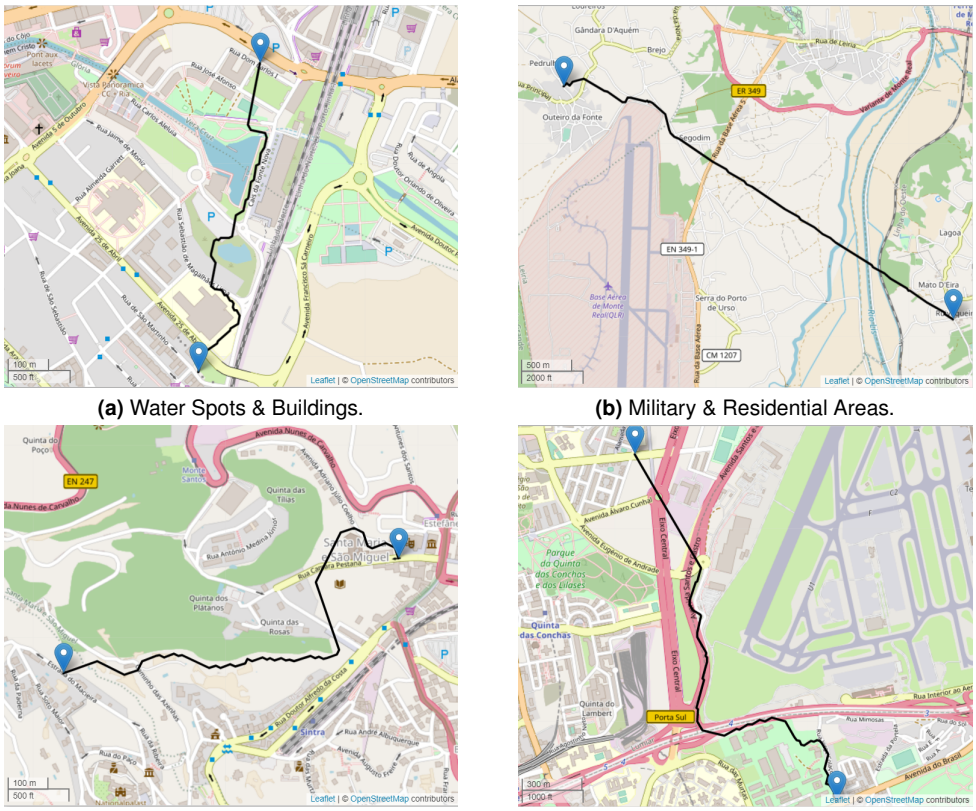
(a) Insufficient obstacle weight. **(b)** Soft obstacle weight. **(c)** Strict obstacle weight.

Figure 5.7: Different obstacle weight effects on trajectories.

5.2.5 Arbitrary Restraints

The previous examples all considered buildings as restrictions, thereby in this short section, we wrapped a compilation of different obstacles. For the demonstration's sake, we pass some other types of restraints to the system to ensure the arbitrarity of the system's restriction avoidance. In fig. 5.8, we perform different path planning missions avoiding the respective geographic restraints imposed.

For instance, we threw some multiple-restriction mission problems to understand the system's behavior, and not only deviates from all restraints but does it without any considerable interference or performance issues between each.



(a) Water Spots & Buildings.

(b) Military & Residential Areas.

(c) Woods & Buildings.

(d) Airport & Buildings.

Figure 5.8: Variety of pertinent drone pathing restraints.

5.3 Online Vs Offline

Throughout this chapter, we have been analyzing different results and performances based on the system's parameterization. This section covers a whole distinctive aspect of the system that is the planning method.

As mention and detailed in section 4.5, the different approaches to the system's core planning task can have contrasting performances. Considering the graph's Offline computation method, we can draw many mission paths in the same area as the graph is computed completely, saving efforts in close-by drone navigation tasks. Additionally, as the planner calculates the trajectories with full knowledge of the surrounding obstacles, it generates complete and more precise results. Notwithstanding, one might not need to detect restrictions away from the shortest path. Thus, the Online approach focuses solely on the one-time type of plan, being much faster, trading-off some precision in certain conditions, as shown in the fig. 5.9 example.

The idea behind the Online approach is to be faster computing a one-time trajectory, using only the necessary geographical points to compute the path to the goal, minimizing the graph construction and the overall task's complexity, clutching a fraction of time of the corresponding offline's procedure. On the

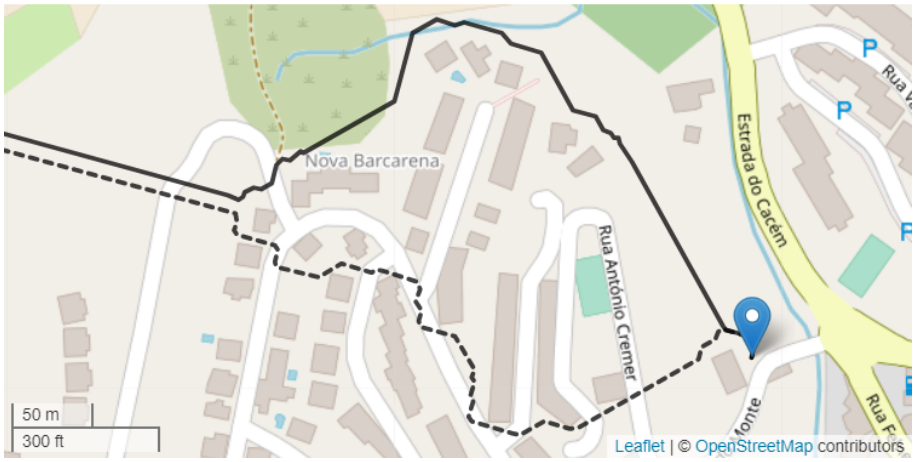


Figure 5.9: Final part's closeup of the trajectory from IST to Location1 mission, with resolution of 2 meters and margin to buildings of 10 meters. Online approach's path represented by the dashed line and Offline approach's path by the solid line.

other hand, the Offline approach relies on a reusable graph generation for the area in question, allowing for stable and numerous planning tasks within the area's perimeter, trading off the first computation complexity for long-line faster-planning tasks.

The Online's approach inherent disadvantage is some loss of information that is not crucial for the planning task but the path's viability. This little nuance, present in fig. 5.9, makes some parts of the Online's trajectory (dashed line) go too close to the restriction outline, not respecting the imposed safety margin, thus violating the security procedure. The root of such a problem comes from the situation that the path drawn by the algorithm does not intersect any restriction, and hence the point's neighbors' weights do not update accordingly. In other words, geographic nodes that should be affected by an obstacle margin are not weight-updated because the algorithm didn't find a nearby obstacle. Suchlike issues are not present in the Offline's approach (solid line) as all the obstacles in the area are identified and accounted for ahead of the planning phase. Thus, as observed in fig. 5.9, both methods take disparate paths to reach the same goal.

5.4 Troublesome Cases

As with all things in life, there's no perfect solution for every type of mission a user can throw at our system. In this section, we will showcase some hard-to-deal problems for our current solution.

In fig. 5.10, we depict a geographic coordinate that can be used as system's input, both as Start or Goal, that disturbs the algorithms' functioning.

As the point location is inside a closed buildings' compound, the algorithm has no gap between buildings, if used as a restriction for the mission, to plan a path where the drone can get in or get out of

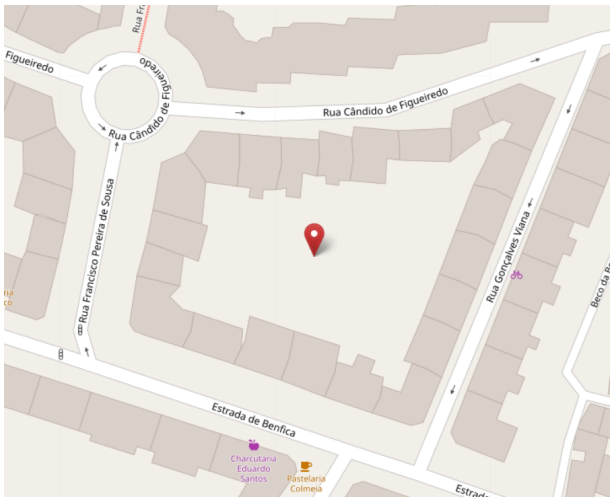


Figure 5.10: Closed buildings' compound troublesome case.

that area. In such a case, the Online algorithm would try to compute many useless trajectories, ending up exhausting every single option, thus becoming very inefficient. Although this case would not impact the Offline computation as much, the output would be flagged as inappropriate due to the violation of the imposed restrictions in the best possible trajectory.

Further, concerning the Mission Area Adjustment feature (section 5.2.1.B), there are cases where an obstacle restriction can be larger than the automatic default area dimensions, as demonstrated in fig. 5.11.

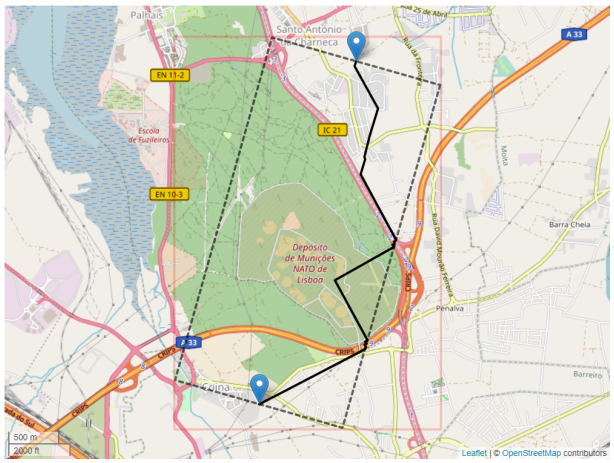


Figure 5.11: Large Restriction Obstacle troublesome case.

The outputs from this type of mission would be inappropriate as there is no possible path around such restriction within the considered environment area. Thus, the only way to address this problem is not to use the default restriction area and provide adjusted operation inputs to the mission planner.

5.5 Output Optimization

The implemented optimization methods considerably improved the system's outputs, reducing the output size and polishing rough paths sections. In the following subsections, we demonstrate examples of this methods' results.

5.5.1 Trajectory Cleaning

Depending on the mission resolution, the output's size reduction can go from 40% to 85% upon applying the trajectory cleaning method alone. Moreover, if the mission profile lies on a few obstacles and the planner can perform mostly straight line sections, where trajectory cleaning is most effective, it discards a large quantity of redundant information that otherwise would be passed on to the drone navigation controller.

(a) Planner's output path. **(b)** In-process output cleaning. **(c)** Optimized output.

Figure 5.12: Iterative cleaning results' visual representation.

After a few iterations, once the trajectory reduction stabilizes, we are left with the essential path around the obstacles, as depicted in fig. 5.12c. Moreover, this is a rattling task, as it takes less than half a second to process complex trajectories (200 000+ nodes), shown in fig. 5.16b.

5.5.2 Path Smoothing

The path smoothing method is quite orientation-specific, meaning that if the planned trajectory has some troubled orientation, the algorithm makes several fuzzy decisions, hence, creating the direction changes to achieve the best result, as explained in section 4.6.2.

The example on fig. 5.13 shows precisely the importance of such an optimization method appliance as it notably enhances the trajectory path, removing all the 'noise' from several sections on an obstacle-dense scenario.

(a) Before smoothing process.

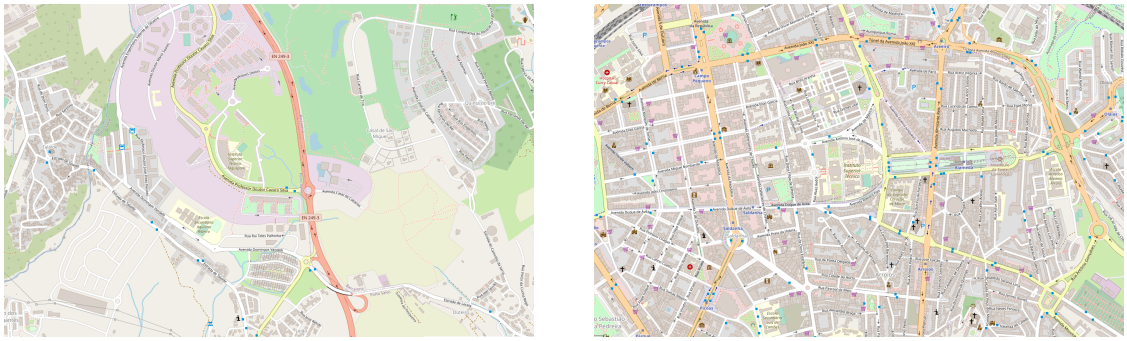
(b) After smoothing process.

Figure 5.13: Smoothing method appliance result.

Moreover, we found that applying path smoothing after the trajectory cleaning is reliably better when using both methods to minimize information loss and increased performance. In conjunction, these optimization steps serve as a post-planning quality filter for our tool outputs, minimizing the information overhead and overall aspect of its trajectories.

5.6 Computational Complexity

In this section, we study the computational complexity of the system. As in many applications, we might want to run the system frequently, or even in an embedded system, we need to solve the path planning in a short period. We considered several system aspects, such as the amount of node usage, individual process times, and total run-time, to compare and state the different approaches in a series of test cases in two distinct scenarios, presented in fig. 5.14.



(a) Location1 - Area around IST Taguspark, Porto Salvo - Lisbon, as low obstacle density area.

(b) Location2 - Area around IST Alameda, Lisbon city center, as high obstacle density area.

Figure 5.14: Complexity testing scenarios.

One aspect that contributes to a high system's workload is the total number of geographical points considered. This aspect, relative to the granularity (described in section 5.2.2), linearly enlarges the amount of memory needed to compute a path as it grows. Moreover, when processed against the total number of restrictions in the obstacles filtration process (section 4.3.3), it increases the trajectories computation complexity exponentially.

Nonetheless, it's relevant to note that the Online algorithm was conceived as the main alternative to this expensive operation, conserving most of the performance possible.

In section 5.6, we present the analysis of the total amount of points in the grid (or nodes in the graph) and the number of nodes considered by the planner, or in other words, the nodes treated obstacles by the system. The presented figure provides a ratio estimation of processed nodes, what we called Obstructed Nodes Ratio (ONR), throughout the different resolutions.

ONR is a density indicator for each mission's environment, table 5.1. This indicator behaves differently for each approach. In Offline's case, it remains constant through the different resolutions, directly providing an obstacle density ratio per area, as the territory is the same and all obstacles are processed. Alternatively, in the Online's case, the ONR decreases as the resolution increases, providing the minimum percentage of processed nodes for the same region. With its dynamic method, the higher the resolution, the fewer the percentage points are processed when achieving the most optimal mission's trajectory.

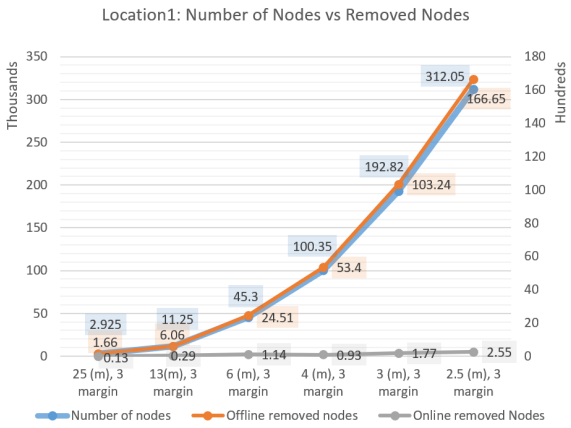
Table 5.1: Obstructed Nodes Ratio table, for both low and high density scenarios' comparison.

Resolution (m)	Obstructed Nodes Ratio (ONR)			
	Low Density		High Density	
	Online	Offline	Online	Offline
25	0.49 %	5.4 %	20.69 %	39.75 %
13	0.24 %		15.65 %	
6	0.17 %		11.29 %	
4	0.10 %		8.03 %	
2.5	0.06 %		4.04 %	

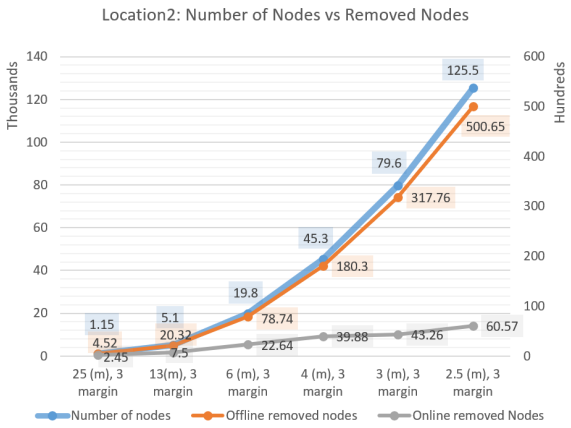
One of the results' principal insights is the considerable difference in the number of obstacles nodes handled by the two algorithm's methods, as we can observe in fig. 5.15a and fig. 5.15b. Online's approach achieved a much lower number of obstacle node processing than Offline's, with comparison values in table 5.1, thus making the development goals meet the expectations.

Further, studying the two methods' differences and implications, we analyzed the system's time complexity by directly measuring each mission's trajectory planning stage with the different graph resolutions and safety margins in the two different obstacle density areas, as shown in fig. 5.16a and fig. 5.16b.

A first observation shows that the total time difference between Online's and Offline's approaches is evident, with the online system being much faster. More interestingly, the two individual processes that directly handle the restriction's filtration have long process times, being the root of the biggest

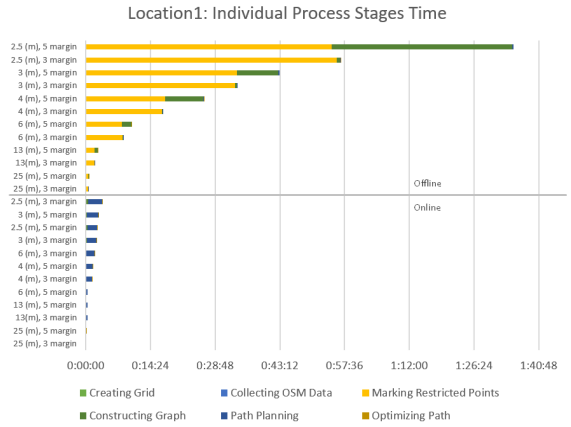


(a) Low obstacle density test mission (IST-Taguspark).

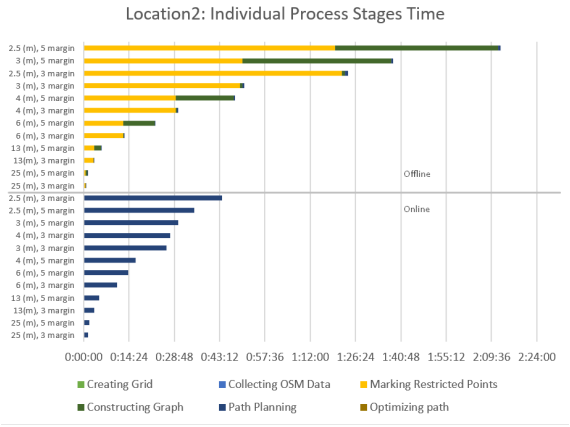


(b) High obstacle density test mission (IST-Alameda).

Figure 5.15: Complexity comparison between the amount of nodes treated and marked as obstacles in online and offline methods.



(a) Low obstacle density test mission (IST-TagusPark).



(b) High obstacle density test mission (IST-Alameda).

Figure 5.16: Complexity comparison between each individual process stage's time (hh:mm:ss) for mission planing.

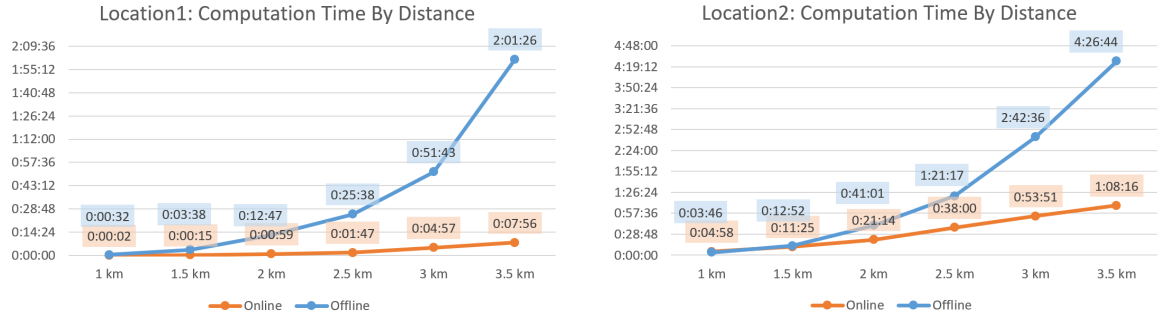
Offline's solution complexity problem. Therefore, Online's method, by avoiding these stages, outperforms immensely the Offline method's run-time when solely comparing total execution times. Such difference can be in the order of fifteen times in a low-density area and up to three times in a high-density one when the Offline's algorithm is running without any pre-computation of the mission's environment.

Subsequently, we can see the impact of the several resolutions and margin levels. Although highly visible on the offline method, we can witness higher resolution values (meaning more points and less distance in between) impacting the system's performance almost linearly. Additionally, the margin levels are very significant in higher resolution plans. For instance, increasing the margin level from 3 to 5, as depicted in both section 5.6, raised an exponential delay on the graph's construction process for the Offline's approach, and similar behavior in the path planning task of the analogous Online's method.

Considering other processing stages, the time amount that the system spends collecting data from

the APIs is almost negligible, being constant throughout the experiments, where the area is the same. Likewise, the amount of time that the system requires to optimize the output path is insignificant when compared to other process stages, even in the more complex scenarios.

Further, into the system's performance analysis line, we computed a series of time complexity over the objective distance to evaluate the influence of the varying distance length between the start and goal positions, from 1 km up to 3.5 km, as demonstrated in fig. 5.17. The tests were performed in two different areas (low and high obstacle density), each with similar obstacle distribution throughout the computed paths, and calculated one-by-one using a 6 meters resolution with 18 meters safety margin to buildings.



(a) Low obstacle density area.

(b) High obstacle density area.

Figure 5.17: Computation time of different distance missions.

The leading aspect of the two graphs is the curves' similarity. In both cases, despite their difference in obstructed nodes ratio (ONR), the Offline algorithm presents itself with an exponential curve, while the Online algorithm offers a more linear solution.

Furthermore, it's perceptible the performance difference when the obstacle ratio is high, translating in a two-hour difference in the computation of the 3.5 kilometers' mission. Additionally, the disparity between Online and Offline calculation varies with the obstacle number, reaching around 15 times faster in the low-density scenario but only 3-4 times in a high-density environment.

6

Conclusion

Contents

Performing trajectory planning for UAVs is essential to the future ahead of this technology capabilities. Drawing the roads for these aerial agents should not be a tedious manual task. Using a system that can perform environmental analyses of geographic features and further plan a path that avoids any risk of compliance and safety for the vehicles brings the possibility of tooling such technology for an enormous variety of tasks. We focused on drone delivery tasks, motivated by this application growth in recent years, creating means for casual users to adopt such drone solutions.

6.1 Conclusions

Our approach consists in applying grid-based search methods to solve real-world drone navigation problems. For this, we developed a system of aggregated components, each solving a unique issue within drone trajectory planning, including: geographical data acquiring, obstacle detection and filtration, treated data structuring, trajectory planning, and output formatting. With such mechanisms compilation, we designed a high level arbitrary restriction avoidance and trajectory planner tool for drones [1].

Within the scope of assessing the tool creation and viability, we implemented two differentiated methods of trajectory planning, each focusing on different use cases for the tool, varying the way of data handling howbeit the path's computation. For the first method, we used static obstacle identification within the creation of the searchable grid-graph structure. As for the second, we adopted a dynamic approach, starting with an empty graph and feeding it the obstacle information during the path's planning search. With both methods, we aimed for complete, safe, and optimal trajectory paths as two variant solutions for one succinct result: the best trajectory for a drone to navigate given an arbitrary set of geographical restrictions.

To test the system's capabilities, we ran a series of experiments tracking down the time performance at each stage of the trajectory planning process. Additionally, we compared the results of both previously mentioned methods throughout the experiments, analyzing the details of each solution's pros and cons.

The main aspects we wanted to evaluated to validate our system's practicability were the results' quality and the system's performance. We ended up with a flexible system that is input-dependent to ensure both of these properties, requiring a trade-off between results' precision and computation complexity. Despite this, we believe the project represents a first-stage automation tool, showing promising results.

This project contemplates an original attempt to get an automatic object restricted navigation scheme in real-world environments for UAV missions relying on open-source data.

Although we did not perform practical experiments, we believe that the system's outputs are strictly shaped to apply on a real drone mission and achieve the expected approximate safe and optimized

[1] The described tool is available, and it's open for use and inspection https://github.com/AndrCarvalho/DroneOpenTool

navigation outcomes. Base on the demonstrated results, we can state that the taken approach is viable within the initial expectations, yet far from a stable performance required by a reliable drone mission control system – at least under the followed design.

6.2 System Limitations and Future Work

The developed system has some limitations, for example, the one directly addressed in section 5.2.1 – concerning the reliability of geographic information source, the edges cases discussed in section 5.4, or performance-wise when overloading the computation with arbitrary parameterization. Nonetheless, we will focus on enumerating different logical features that could enhance similar works to come.

One interesting factor to study would be the 3D property of the drone environment, contrasted by the 2D road planning problems. Considering height for drone-related navigation problem is highly relevant for an overall solution. Subsequently, given a more detailed energy model, see if one could predict and more effectively plan trajectories around discrepant terrain.

Further, concerning the performance of the grid-based employed approach, one could consider different obstacle filtration strategies. Additionally, options outside of grid-based search could be tested and compared, such as Visibility Graph searches, given that we already have the obstacle topology.

As an extent of the tool's usability, aggregating restriction models based on several scenarios, such as urban-focused restrictions, rural-relaxed restrictions, law-based restrictions, and more, could enhance the tool's interaction practicality, simplifying the characterization of the respective mission environments most suitable restraints.

Lastly, addressing the following development of akin works with a mechanical tuning perspective, adjusting drone specific variables to more properly test these types of tools' results and analyze the UAV's response on physical navigation missions.

Bibliography

[1] S. Jung and H. Kim, "Analysis of amazon prime air uav delivery service," *Journal of Knowledge Information Technology and Systems*, vol. 12, no. 2, pp. 253–266, 2017.

[2] D. Deploy, "Commercial drone industry trends," *Report Drone Deploy, San Francisco*, 2018.

[3] A. Inc., *Amazon Prime Air*, Accessed on 21 June, 2020. [Online]. Available: www.amazon.com/primeair

[4] D. I. GmbH., *DHL parcelcopter launches initial operations for research purposes*, 24 September, 2014. [Online]. Available: https://www.dhl.com/en/press/releases/releases_2014/group/dhl_parcelcopter_launches_initial_operations_for_research_purposes.html

[5] J. Stewart, *Google tests drone deliveries in Project Wing trials ,BBC*, Published 28 August, 2014. [Online]. Available: https://www.dhl.com/en/press/releases/releases_2014/group/dhl_parcelcopter_launches_initial_operations_for_research_purposes.html

[6] K. Gdowska, A. Viana, and J. P. Pedroso, "Stochastic last-mile delivery with crowdshipping," *Transportation research procedia*, vol. 30, pp. 90–100, 2018.

[7] C. Barrado, M. Boyero, L. Brucculeri, G. Ferrara, A. Hately, P. Hullah, D. Martin-Marrero, E. Pastor, A. P. Rushton, and A. Volkert, "U-space concept of operations: A key enabler for opening airspace to emerging low-altitude operations," *Aerospace*, vol. 7, no. 3, p. 24, 2020.

[8] ANAC, *Voa na Boa - Código Drone*, accessed May 10, 2020. [Online]. Available: https://voanaboa.pt/codigo-drone

[9] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2016.

[10] G. J. Leishman, *Principles of helicopter aerodynamics with CD extra*. Cambridge university press, 2006.

[11] G. Navstar, “User equipment introduction,” *Department of Defense Document MZ10298*, vol. 1, 1996.

[12] D. T. Wooden, “Graph-based path planning for mobile robots,” Ph.D. dissertation, Georgia Institute of Technology, 2006.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[15] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] S. M. LaValle, *Combinatorial Motion Planning*. Cambridge University Press, 2006, p. 206–256.

[17] E. Zurich, “Qgroundcontrol: Ground control station for small air land water autonomous unmanned systems,” 2013.

[18] A. D. Team, “Apm planner 2. open source project,” 2016.

[19] C. Ramirez-Atencia and D. Camacho, “Extending qgroundcontrol for automated mission planning of uavs,” *Sensors*, vol. 18, no. 7, p. 2339, 2018.

[20] S. R. Haque, R. Kormokar, and A. U. Zaman, “Drone ground control station with enhanced safety features,” in *2017 2nd International Conference for Convergence in Technology (I2CT)*. IEEE, 2017, pp. 1207–1210.

[21] V. T. Nguyen, K. Jung, and T. Dang, “Dronevr: A web virtual reality simulator for drone operator.” in *AIVR*, 2019, pp. 257–262.

[22] M. Haklay and P. Weber, “Openstreetmap: User-generated street maps,” *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[23] K. Yakovlev, D. Makarov, and E. Baskin, “Automatic path planning for an unmanned drone with constrained flight dynamics,” *Scientific and Technical Information Processing*, vol. 42, no. 5, pp. 347–358, 2015.

[24] S. V. Ragavan and V. Ganapathy, “A unified framework for a robust conflict-free robot navigation,” in *Proceedings of World Academy of Science, Engineering and Technology*, vol. 21. Citeseer, 2007.

[25] G. H. Elkaim, F. A. P. Lie, and D. Gebre-Egziabher, “Principles of guidance, navigation, and control of uavs,” *Handbook of unmanned aerial vehicles*, pp. 347–380, 2015.

[26] K. Sundar and S. Rathinam, “Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots,” *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 287–294, 2013.

[27] F. Guerriero, R. Surace, V. Loscri, and E. Natalizio, “A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints,” *Applied Mathematical Modelling*, vol. 38, no. 3, pp. 839–852, 2014.

[28] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, “Ub-anc planner: Energy efficient coverage path planning with multiple drones,” in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 6182–6189.

[29] S. Behnke, “Local multiresolution path planning,” in *Robot Soccer World Cup*. Springer, 2003, pp. 332–343.

[30] V. Ayala-Ramirez, A. Pérez-García, F.-J. Montecillo-Puente, R. Sanchez-Yanez, and E. Martinez-Labrada, “Path planning using genetic algorithms for mini-robotic tasks,” in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, vol. 4. IEEE, 2004, pp. 3746–3750.

[31] T. W. Manikas, K. Ashenayi, and R. L. Wainwright, “Genetic algorithms for autonomous robot navigation,” *IEEE Instrumentation & Measurement Magazine*, vol. 10, no. 6, pp. 26–31, 2007.

[32] R. Kala, A. Shukla, R. Tiwari, S. Rungta, and R. R. Janghel, “Mobile robot navigation control in moving obstacle environment using genetic algorithm, artificial neural networks and a* algorithm,” in *2009 WRI World Congress on computer science and information engineering*, vol. 4. IEEE, 2009, pp. 705–713.

[33] L. Xia, X. Jun, C. Manyi, X. Ming, and W. Zhike, “Path planning for uav based on improved heuristic a$*$ algorithm,” in *2009 9th International Conference on Electronic Measurement & Instruments*. IEEE, 2009, pp. 3–488.

[34] A. Chatterjee and H. Reza, “Path planning algorithm to enable low altitude delivery drones at the city scale,” in *2019 international conference on computational science and computational intelligence (csci)*. IEEE, 2019, pp. 750–753.

[35] S. M. LaValle *et al.*, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

[36] J. R. Rufa and E. M. Atkins, “Unmanned aircraft system navigation in the urban environment: A systems analysis,” *Journal of Aerospace Information Systems*, vol. 13, no. 4, pp. 143–160, 2016.

[37] J. Besada, I. Campaña, L. Bergesio, A. Bernardos, and G. de Miguel, “Drone flight planning for safe urban operations,” *Personal and Ubiquitous Computing*, pp. 1–20, 2020.

[38] A. D. Snow, J. Whitaker, M. Cochran, J. V. den Bossche, C. Mayo, J. de Kloe, C. Karney, G. Ouzounoudis, J. Dearing, G. Lostis, and et al., “pyproj4/pyproj: 2.6.1 release,” May 2020.

[39] PROJ contributors, *PROJ coordinate transformation software library*, Open Source Geospatial Foundation, 2020. [Online]. Available: https://proj.org/

[40] J. D. Greenberg, M. G. Logsdon, and J. F. Franklin, “Introduction to geographic information systems (gis),” in *Learning Landscape Ecology*. Springer, 2002, pp. 17–31.

[41] W. R. Franklin, *PNPOLY - Point Inclusion in Polygon Test*, 1994-2006. [Online]. Available: https://wrf.ecse.rpi.edu//Research/Short_Notes/pnpoly.html

[42] T. C. Hales, “The jordan curve theorem, formally and informally,” *The American Mathematical Monthly*, vol. 114, no. 10, pp. 882–894, 2007.

[43] A. Hagberg, P. Swart, and D. S Chult, “Exploring network structure, dynamics, and function using networkx,” Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[44] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub *et al.*, “The geojson format,” *Internet Engineering Task Force (IETF)*, 2016.

[45] P. Crickard III, *Leaflet. js essentials*. Packt Publishing Ltd, 2014.

CLI DroneOpenTool

```
usage: DroneOpenTool.py [-h] [--outputPath path] --mission name --start start
                        --goal goal [--planner planner] [--output output]
                        --gran granularity --margin marginlvl --restr
                        restrictions [restrictions ...] [--area area]
                        [--obs_weight weight]

DroneOpenTool: Create and Plan your drone mission.

optional arguments:
  -h, --help            show this help message and exit

Mission details:
  --outputPath path     the path to workspace
  --mission name        name to the mission
  --start start         starting location: (lat, lon)
  --goal goal           ending location: (lat, lon)
  --planner planner     planner type: Online, Offline
  --output output       output type [default: Html]

Computation Parameters:
  --gran granularity    trajectory granularity
  --margin marginlvl    level of margin to obstacles
  --restr restrictions [restrictions ...]
                        list of restriction type e.g.['buildings', 'airways',
                        'residential', 'water', 'woods', 'military'] (add
                        multiple --restr restriction if more than 1)
  --area area           custom defined rectangular area:
                        (south,west,north,east)
  --obs_weight weight   obstacle's weight
```

Figure A.1: ꜜhćommand output with options to run the tool.

```
DroneOpenTool.py --outputPath D:\Examples --mission Testing_Online_Runner --start (38.6371,-9.0195) --goal (38.5816,-9.0444) --planner Online --gran 200 --margin 3 --restr buildings woods
2021-07-30 00:22:50.634192
distance start to end: 6531.537656705138 m
distance between nodes: 33 m
furthest neighbour: 46 m
Starting point: (38.6371, -9.0195)
Ending point: (38.5816, -9.0444)
Number of grid nodes: 19800
Number of buildings: 571
Number of woods: 5
Problem Structure: Collected
Margin: 3 - 97.97306485057706
Structure Complete
Planning...
Initial path: 215  | Remove Excess path: 26
Difference path excess: 189
Non-excess path: 26  | Smooth path: 19
Difference path smoothing: 7
Creating Grid: 0:00:00.107007
Collecting OSM data: 0:00:01.347623
Restricted points: 0:00:00
Constructing Graph: 0:00:01.544967
Path Planning: 0:02:18.583847
Optimizing Path 0:00:00.002999
Total time: 0:02:21.586443
Success!
```

Figure A.2: Example run command and respective console logs.

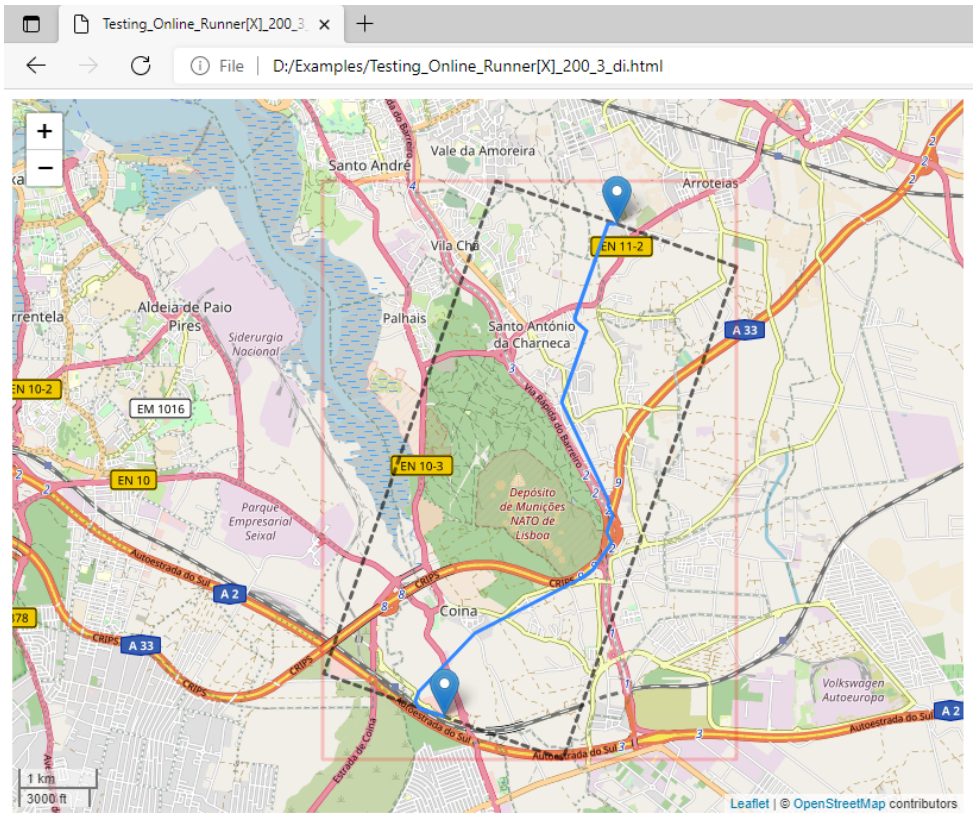


Figure A.3: Produced result from fig. A.2's command.