POLITECNICO DI TORINO

Master's Degree in Mechatronics Engineering

Master's Degree Thesis

Path planning strategies for drone delivery for life-saving pharmaceuticals

Supervisors

Prof. Giorgio Guglieri

Dr. Stefano Primatesta

Prof. Enrico Cestino

Candidate

Francesca Fiorentino

Academic Year 2020/2021

Abstract

In the last years, Unmanned Arial Vehicles (UAVs) are getting popular and are widely used to provide many applications. Specifically, a promising UAV application discussed in this thesis is the distribution of life-saving pharmaceutics using UAVs. This thesis aims at defining a path planning strategy to be used to provide the distribution of life-saving pharmaceutics using UAVs in populated areas.

The proposed strategy is based on a method already suggested in the literature by the research group, consisting in two phases: first, a risk-map is generated analyzing the risk of a specific area; then, a path planning algorithm based on the well-known Rapidly-exploring Random Trees (RRT) algorithm searches for the optimal path that minimizes the overall risk and the flight time.

Different optimization strategies are investigated and implemented to plan a specific flight mission for drone delivery, while satisfying specific requirements, such as: (i) minimization of the risk; (ii) minimization of the flight time; (iii) minimization of the risk, while forcing a constraint on the flight time; and, (iv), minimization of the flight time, while forcing a constraint on the risk. These strategies return different solutions adopted based on the type of life-saving pharmaceutics to be delivered.

The proposed strategies are tested in simulations. First, simplified risk maps are used to demonstrate the effectiveness of the proposed solutions. Then, realistic risk maps are taken into account emulating a realistic scenario of life-saving pharmaceutics delivery in the city of Turin, Italy.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the last decades the increased urbanization and traffic situation is pushing ground transport systems to their limits. Without skillful planning and innovative, smart solutions, cities are risking diseconomies such as congestion and pollution, starting to outweigh scale benefits and leading to a deteriorating quality of life and a loss of economic dynamism [1].
As a result, cities are demanding for efficient and effective mobility solutions, so it has been necessary to integrate concepts for logistics and mobility: micro-hubs, smart car-sharing models, micromobility in the last mile, drone transport, cable cars and underground solutions [2].
Therefore, major aviation and automotive manufacturers, city authorities and technology companies are developing innovative Urban Air Mobility (UAM) solutions. Indeed, Unmanned Aerial Vehicles (UAV) have the potential to overcome current difficulties and to create a faster, cleaner, safer, and more integrated transportation system [1].
Nowadays, drones are used for various purposes in urban environments, such as traffic monitoring, photography, and weather forecasting [3]. Moreover, they can be used for commercial operations, such as the delivery of goods and materials. This has been possible due to new technologies, like electric propulsion and enhanced battery capacity, applied to vertical take-off and landing systems [4]. Therefore, a high number of different drones are available, which makes difficult a selection among them. To not be overwhelmed by all this information, some algorithms have been developed to select the best drone to run an operation [5].

Another important scenario, in which the UAM is developing, is the medical one: air ambulance, emergency supply delivery, transport of organs or search and rescue support. Actually, which each donation (almost 150 every year in Piedmont and 1700 in Italy), it is needed that blood samples of the donor are delivered in the shortest time possible at the regional reference laboratories. Therefore, the

application of drone delivery is crucial to overcome time limits and unforeseen events due to traffic.
Some researches have been conducted to UAVs applications, in order to find the best trade-off between efficiency and the costs, mass of the material to transport and payload of drones, when the transportation of medical supply are delivered among two hospitals.

A preliminary study has been conducted by Amoroso et al [6], that have been analyzed the features of the transports and of the medical supply to be delivered, the point of departure and arrival of the biological material transported as part of the donation and transplantation network of organs, tissues and cells of the regions of Piedmont and Valle D'Aosta. Additionally, on their publication has been reported all the features that the box, containing pharmaceuticals or organs, must own to avoid the deterioration of the material, to reduce as possible the cost of the transport and to satisfy the regulatory framework.
In the master's degree thesis of Cavallo [7] effective research on the distribution of life-saving medications using UAVs have been provided. The research has been based on currently available mid-size drones and for each of them the optimal trajectory, from Ospedale Molinette to Ospedale San Giovanni Bosco, has been computed, by means of the Optimal Rapidly-exploring Random Tree (RRT*) algorithm. Moreover, a comparison of the features of each drone has been presented to select the most appropriate for a specific mission.

A further step from the work of Cavallo [7], has been done in this thesis. In his study the optimal trajectory was the path that minimized the risk of the map and the flight time. Object of this thesis has been the investigation of different optimization strategies, thus the optimal trajectory is not only the minimization of the risk of the flight time, but also the compliance with specific requirements.

This thesis is composed by 6 chapters. In Chapter 2 the applications of drones in medical delivery is analysed, considering researches that highlight the conditions at which it is subjected, the implemented technologies thus the different type of drones and their features, and the rules that it must satisfy. In Chapter 3 the path planning algorithm, applied for the search of the trajectory that the drone will follow, is described. Therefore, a comparison between the different type of algorithm is made, emphasizing the advantages of the chosen algorithm. In Chapter 4 the implemented algorithm is portrayed and so the risk map used for the computation of the total risk of the path found by the algorithm. In Chapter 5 all the simulations are reported, which are done on the ideal and real maps, obtained from the implemented optimization. Eventually, in Chapter 6, the results are evaluated and the conclusions are made.

Chapter 2

Drones for medical transport: State of the art

2.1 State of the art of Urban delivery applications

Nowadays, drones are employed for commercial and medical transports. In particular, in the healthcare sector, drones have shown great potential in the delivery of medicinal supplies. They are exploited for the advantage of overcoming logistic challenges, such as traffic delays, insufficient transport network to reach isolated areas and natural disasters. However, they can be employed only if they do not change the quality of transported products. In the literature, there are several case studied that address the use of UAVs in healthcare.

A general insight into the application of drones in medical delivery is described in the study conducted by Thiels et al. [8]. Their work is focused on the different kinds of hospitals in the US. Smaller hospitals have limited resources, so the patients have to be transferred to a trauma center to have all the medical supplies. The work shows that this problem could be addressed by UAVs, which can quickly deliver medicines and blood products from larger hospitals to smaller ones and even directly to the injured patients at the scene. Their work have also highlighted that UAVs, depending on their size, require very little space to land with little to no specialized facilities. Thus, they are ideally suited for this type of application.

The implication of drone delivery on the safety and quality of medicines has been subject of the study of Dr Paul Royall and Patrick Courtney [9]. On their article they have published that, during drone transportation, stresses (like vibration,

g-force, rapid changes in pressure, humidity and temperature excursions) may affects the medicine.
Speed and precision of a UAVs are paramount when transporting life-preserving medicines to emergency situations within the golden hour. However, during the transportation, the quality of the medicine must be maintained, and the impact of drone failure on the patient safety must be thoroughly modelled.
In their work it is highlighted how the security of the supply chain is important. The delivery packaging must include extra security technologies to protect the contents since regular transportation to a particular address may gain the attention of those with the intent of stealing the medicines.
On their work delivery projects have been conduct and the impact of drone flight on the quality of vaccines, antimalarials and blood products have been reported. As result, when large distances are involved, the delivery has been peer-to-peer, i.e. medical professional from a medical hub to a field surgery or clinic, providing the potential for assessing the quality of the medicines on arrival by specifically trained staff.
Summing up, for Royall and Courtney, when delivering medicines, five rules must be considered:

1. Flight time and range conditions;
2. On board conditions;
3. Impact of drone failure;
4. Ensure quality post flight;
5. Security of supply chain.

To understand the impact of drone transportation on the quality of a medicine another study has been led by Hii, Courtney and Royall [10]. They recommend that, when considering the drone delivery of medicines, five tests need to be applied. These tests must determine the safe flight time and range, the quality of the medicine post flight, the onboard conditions experienced by the medicine, the security of the drone supply chain and the effect of drone failure on both the medicine and the environment. They used the pharmacopoeia methods to investigate on the effects of temperature and vibration of a drone flight on insulin. This medicine is a peptide-based drug that easily unfolds to cause irreversible aggregation when subjected to environmental stresses. The result of the study emphasize that drone transportation of insulin is feasible, its quality was maintained after exposure to environmental stresses that simulate a 30-minute UAV delivery, which involves temperatures from -20°C to +40°C and vibration frequencies in the range of 0–40 Hz).

The quality requirements of transportation is also subject to the work produced by Amukele et al. [11] that demonstrates the safety of the delivery of blood products using drones. Their analysis regard Red Blood Cells (RBCs), PLaTelets (PLTs), and Plasma units Frozen within 24 hours of collection (FP24). Since blood products for the purpose of transfusion are not subject to the IATA regulations of infectious substances, the adopted approach for packaging was to approximate methods used for automobile transportation of blood products, while minimizing the amount of passive temperature buffers, due to payload constraints, by means of a cooler.
Their work demonstrates that the transportation of RBCs, PLTs, and FP24 units using UAVs has no adverse consequences. Furthermore, to overcome the problem of acceleration, they selected the multi-rotor drone, able also to guarantee an accurate takeoff and landing.

The assignment proposed by Eichleay et al. [12] reports how UAVs should be set up in parallel to the logistic systems of hospitals and medical warehouses. Indeed, optimizing routing will require a new set of variables to determine the most efficient route. Moreover, the financial assessment must be taken into account, since the trade-off between distance, weight, and cost may represent a barrier to some applications of UAVs in healthcare. For this reason, transportation using drones will likely supplement medical supply chains, rather than replace road transport. In addition, this study shows how crucial and complex it is to understand under which conditions drones are cost-effective.

Finally, an insight into the critical role played by the governance in regulating air space and the transportation of medical goods through drones is reported.
In the report of Eichleay et al. [12] a UAV Delivery Decision Tool is also analysed. It is designed to help the users to understand the context within which their products need to operate and to overcome all the issues written above. The Tool is a 4-part: first users define the transport problem they are trying to solve using UAVs, then transportation parameters are included. The last 2 parts regard offline worksheets to identify and analyze stakeholders and to select preliminary sites for conducting UAV operations.

The analyzed articles have shown how important is the implementation of UAVs in the healthcare sector. The transportation of medical supplies by means of UAVs is increasing in the last years, since has been proved that this kind of delivery has no adverse consequence on transported goods. Moreover, universities, hospitals, and private companies are investing in this new technology to find new and innovative ways to implement them.

2.2 Regulatory Framework

In the report of Konert, Smereka and Szarpak [13], has been demonstrated that, in the use of drones for medical purposes, it is very important to be aware of the existing regulations, the so-called Beyond Visual Line Of Sight (BVLOS) operations. In most countries, it is now possible to operate with the Visual Line Of Sight (VLOS), i.e. operations in which the operator or observer of the flying model maintains a direct eye contact with the flying unmanned aircraft.

The Urban scenario is a critical one. The flight in urban areas is strongly limited by the National aviation agencies, such as ENAC (Ente Nazionale per l'Aviazione Civile) in Italy and the FAA (Federal Aviation Administration) in the United States, which are in continuous update to keep track of the evolution of aerial robotics in urban areas.
The Italian Regulation [14] follows a distinction based on the level of risk of the operation. A primary division hinges on critical (BVLOS) and non-critical (VLOS) operations, while a second division is on the maximum takeoff mass (below or above a maximum takeoff mass of 25 kg).

For what reported in the Italian Regulation [14], VLOS flights avoids to flight over people and congested areas. In particular, UAVs must keep a distance of 150 m to congested areas and 50 m to people who are not under the remote pilot's direct control. In order to run non-critical operations, the remote pilot must complete an online course available on the ENAC website.
Moreover, drones with a takeoff mass less or equal to 2 kg are part of the non critical operations, provided that design aspects and the remote pilot have features ascertained by ENAC.

As reported in the Italian Regulation [14], the critical operations are those where the requirements of a non-critical operations are not respect, even partially. From what reported in ENAC there are different scenarios: case in which the operation is in VLOS and falls within an existing scenario, and case in which the operation cannot be associated with a standard scenario, including operations in EVLOS (Extended Visual Line Of Sight) and BVLOS.

As declared in the article [13] and in the Italian Regulation [14], BVLOS flights will be allowed for drones weighing up to 25 kg. Anticollision, precision lighting, camera to observe the surroundings and devices must be part of these drones. The devices are used to automatically maintain altitude and distance from the operator below the maximum allowable value. Moreover, to handle situations of communication loss, the drones must be equipped with monitor flight parameters,

enable location of the basic drone (location, speed, altitude, and direction of flight), and emergency location.
Furthermore, such drones must be able to automatically perform emergency procedures: (1) continue the flight along the programmed route; (2) end the flight by emergency landing; and (3) make a flight to the pre-programmed location.
Another important aspect concerning the BVLOS operations is that some flights will be allowed outside the dedicated zone: operational (for the needs of police, firefighting, health, search and protection of state security); specialized (for the purposes of supervision, control or protection); automatic (for the needs of supervision, monitoring, and agrotechnical activities); and training.

In medical field, it is allowed also the use of drones with a takeoff mass less than 25 kg. However, it is required the consent of the President of Civil Aviation Autority (CAA), so the drone must be registered in the D-Flight portal [15] and have a specific QR code printed on the aircraft. Additionally, information on the planned flights needs to be published by an air traffic service provider, that also has to be informed by the entity at least 7 days before the flight dates.

As already said the urban areas are a critical scenario due to they are inhabited environments. For this reason it is mandatory to follow the aviation rules, which are listed in “Circolare ENAC ATM-09” [16], to overfly cities.
The airspace can be divided into different controlled areas (airports, heliports, ATZ (Aerodrome Traffic Zone), and CTR (Control Zone)), outside which the UAV operations are allowed.
VLOS and EVLOS operations are approved without reservation in the airspace only if the drone has a takeoff mass below 25 kg and flights below 120 m of altitude AGL (Above Ground Level).
On the other hand, nearby airports or in ATZ and CTR areas, several limitations are applied to the allowed maximum altitude and they can be found in “Circolare ENAC ATM-09” [16].
In Figure 2.1 is shown an example reported in “Circolare ENAC ATM-09” [16], where a civil airport with instrumental procedures is considered.

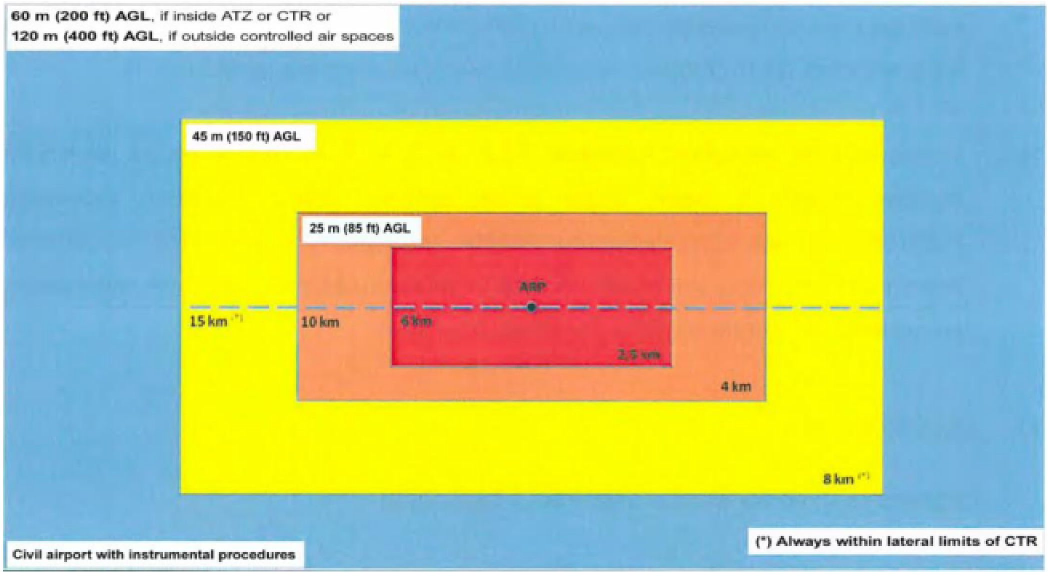

60 m (200 ft) AGL, if inside ATZ or CTR or
120 m (400 ft) AGL, if outside controlled air spaces
45 m (150 ft) AGL
25 m (85 ft) AGL
ARP
15 km (*)
10 km
6 km
2,5 km
4 km
8 km (*)
Civil airport with instrumental procedures
(*) Always within lateral limits of CTR

Figure 2.1: Airspace restrictions when flying close to airports [16].

Each different coloured area indicates which restriction needs to be applied at UAVs application:

- Red Area: UAV applications are not allowed within 6 km from the airports, lengthwise either direction of the runway, and 2.5 km sideways;
- Orange Area: UAV operations are allowed up to 25 m of altitude AGL and between from 6 km to 10 km, longitudinally either direction of the runway, and from 2.5 km to 4 km sideways;
- Yellow Area: UAV operations are allowed within 10 km and 15 km, lengthwise either direction of the runway, while from 2.5 km to 4 km sideways, and up to 45 m of altitude AGL;
- Blue Area: UAV operations are allowed up to 15 km longitudinally either direction of the runway and 8 km sideways. Moreover, operations are allowed below 60 m, inside CTR, or 120 m, outside controlled air spaces, of altitude AGL.

It could happen that UAV operations require to fly outside the regulated areas. So it is necessary an authorization from ENAC to fly in reserved airspace. In particular, BVLOS operations could utilize segregated airspace, which is for the exclusive use of specific operators for a limited amount of time.

The urban area in which this current project is applied, is the city of Turin. In Figure 2.2 is shown a picture of the regulated airspace nearby this city.

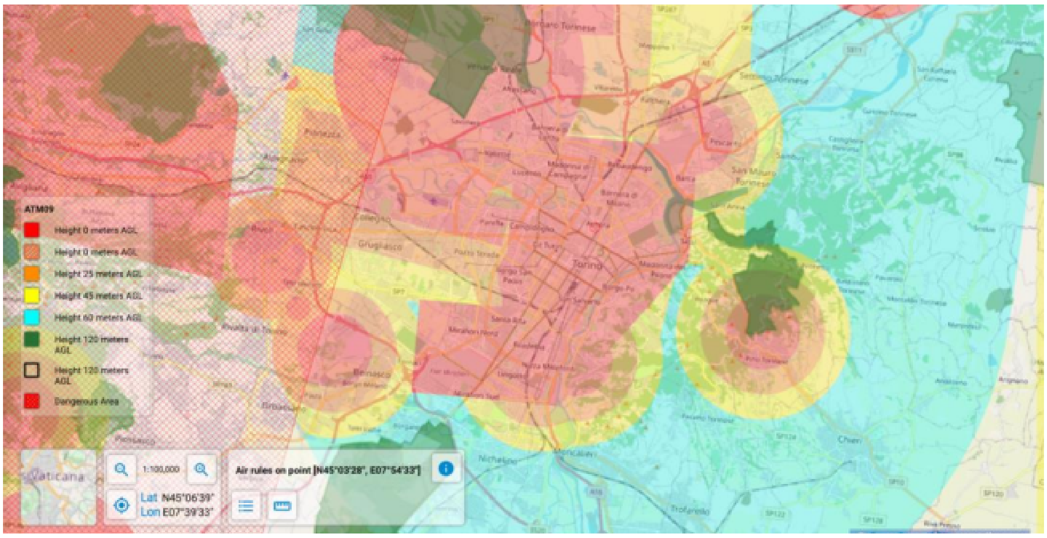


Figure 2.2: Map of the regulated airspace in Turin area [15].

From Figure 2.2, to have a wider scenario in which is possible to compute the optimal path between hospitals that are apart from each other, it is possible to visualize that the operation must be conducted in BVLOS. Therefore, authorization from ENAC is also requiring the exclusive use of the airspace in the operation area as defined in the "Circolare ENAC ATM-09". Time is needed to get the authorization, so a trade-off with the regulator authority could be achieved. However, the flight authorization process is not detailed in this thesis because it is not the aim of this work; it is just outlined to be aware of the regulation presents if it is wanted that a drone overflies an area.

2.3 Drone classifications for Delivery

Unmanned aircraft systems, for delivery, can be classified on their features. A first division is on the type of drone: fixed-wing and multi-rotor, on those categories the drones are also distinguished considering other features, as:

- **Endurance**;
- **Maximum flight time** is half of the endurance wile flying at the cruise speed;
- **Weight**: mass of the drone and the payload is considered;

- **Radius** is the minimum circle that includes the UAV. It is computed by dividing the width or length by two;
- **Frontal Area** is measured by multiplying the width and height of the drone;
- **Maximum payload weight**;
- **Cruise speed** is the 70% of the UAV's maximum speed reachable;
- **Max wind resistance**: an average wind speed thought the year is already considered in the cruise speed, since it has not a predominant direction in the whole year. The city of Turin has an average wind velocity of 7km/h;
- **Price**.

Speed, cost and the flexibility to traverse physical barriers are the attributes that make drones particularly suitable for the delivery of medicines to emergency situations, within the urban environment and over geographically-challenging locations [9].

In the article of Singhal et al [17], all the type of existing drone are classified, moreover it is here reported the table in which the drones are listed, based on weight and range.

Type	**Maximum Weight**	**Maximum Range**	**Category**
Nano	200 gms	5 Km	Fixed wing, multirotor
Micro	2 Kg	25 Km	Fixed wing, multirotor
Mini	20 Kg	40 Km	Fixed wing, multirotor
Light	50 Kg	70 Km	Fixed wing, multirotor
Small	150 Kg	150 Km	Fixed wing
Tactical	600 Kg	150 Km	Fixed wing
MALE	1000 Kg	200 Km	Fixed wing
HALE	1000 Kg	250 Km	Fixed wing
Heavy	2000 Kg	1000 Km	Fixed wing
Super Heavy	2500 Kg	1500 Km	Fixed wing

Table 2.1: UAVs classification based on weight and range.

Fixed-wing UAV provides high endurance, since the wing generates lift instead of the propeller. For this reason most of the energy is consumed for the forward flight. Another aspect of this cluster of drones is that they have longer ranges and greater payload then the rotary-wing.

According with Table 2.1, in the fixed-wing UAV there are: eBee SQ, eBee X, and WingtraOne. A distinction on these families is that eBee family requires an initial forward velocity to takeoff and an obstacle-free site where to land, while the WingtraOne can take off and land vertically, thanks to the combination of its propellers and flaps.

Multi-rotor UAVs take off and land vertically with the ability to hover during flight [10]. However, they have limited endurance and speed, so are unsuitable for large scale aerial mapping, long endurance monitoring and long distance inspection [18]. They can be classified on the size of the drone.

Small-size Multi-rotor UAVs are freedom to fly over an urban area, as written in the section of the Regulatory Framework. Indeed, thanks to their small mass, they have the clearance from ENAC than heavier drones. As consequence of this advantage, small-size multirotor can overfly areas with a high population density, so they also have a shorter path lengths than the fixed-wing. A disadvantage of this cluster is the capability to carry a light payload, so they can be applied only for the transportation of few doses of antidotes.
The drones included in this cluster are the Evo II, the Mavic 2, and the Phantom 4 RTK.

Mid-size Multi-rotor UAVs, due to their weight, are subjects to more restrictions than the small-size family, by the regulatory authorities. Nevertheless, they can carry a larger payload and keeping high cruise velocities. Furthermore, in medical delivery, they can also transport biological samples or blood products.
The drones included in this cluster are the Falcon x8 and the Matrice 200 V2.

A new category, which merging the benefits of fixed-wing UAVs with the ability to hover, has been made by Udroiu and Blaj [10]. This hybrid category is called VTOL (Vertical Take-Off and Landing) incorporates the benefits of both the fixed-wing and multi-rotor aircraft.

In this thesis, the drones with which the optimal path has been computed in the real scenario, are Mavic 2, to evaluate a drone with small mass that can carry light weight; Matrice 200, to analyze a drone with average mass and Matrice 600, to consider a heavy drone.
To make a comparison of all the drones, their features have been reported in the table 2.2.

Mavic 2 has been chosen over Evo II and Phantom 4 RTK since it is the lightest and it has been possible to evaluate a maximum payload.
Mavic 200, with respect to Falcon x8, is fastest and has a smoothest path, moreover its high cruise speed allows it to be also the quickest drone, apart from the eBee SQ/X, to reach the destination. These aspects make it the best choice for ths class.
Matrice 600 has not been analyzed by Cavallo [7] since he has studied only the behavior of drones with small and mid weight. For completeness also heavy drones are considered in this thesis, and Matrice 600 has been chosen over his class, due to the best trade-off between maximum payload and cruise speed.

Manufacture	**Name**	**Type**	**Endurance (min)**	**Range (km)**	**Weight (kg)**	**Dimensions (cm)**			**Payload max weight (kg)**	**Cruise speed (m/s)**	**Max wind (m/s)**
						Width	**Length**	**Height**			
Airborne Drones	Falcon x8 (33)	Rotor	32	18.5	4	53	45	10	2	15	6.9
	Falcon x4 (33)	Rotor	50	18.5	4	53	45	10	0.5	15	4.1
	Vanguard (34)	Rotor	70	35	9.5	83.5	83.5	35	0.85	18	10.3
Autel Drones	Evo II (35)	Rotor	40	9	1.15	39.7	39.7	-	-	20	17
DJI	Matrice 200 V2 (36)	Rotor	26	8	4.69	88.3	88.6	39.8	1.45	17	12
	Matice 600 (37)	Rotor	30	5	9.6	168.8	151.8	-	6	18	8
	MG-1P (38)	Rotor	20	-	13.7	150	150	57.8	10	10	8
	Mavic 2 (39)	Rotor	31	8	0.9	32.2	24.2	8.4	0.2	20	9.7
	Inspire 2 (40)	Rotor	23	7	4.25	42.5	42.7	31.7	0.81	26.1	10
	Phantom 4 RTK (41)	Rotor	30	7	1.4	35	35	20	-	20	-

Table 2.2: Drones comparison with their features.

Chapter 3

Path planning: state of the art

3.1 Path planning definition

Path planning is one of the most crucial research problems in robotics from the perspective of the control engineer. It is applied in guiding the drone to reach a particular objective from very simple trajectory planning to the selection of a suitable sequence of action. By choosing a proper algorithm, path planning is able to generate an appropriate trajectory that guide the drone from the start to the ending point through several intermediate states. There may be more than one optimal path or even no possible optimal trajectories, where optimal means the ideal path that minimize a specific cost from start and goal and that is far from obstacles/collision-free, and spend the shortest time to reach the goal state [19].

There exist two types of path planning: a global path planning, where the optimal path is computed from the environment information, that are completely known in the control design; a local path planning, in which the path is generated in real-time taking data from sensors during the movement of the drone. The first type is more expensive in implementation, while the second is more complicated in design.

The path planning applied in this thesis is the global path planning that involves two parts: establishment of the environmental model and the path planning strategy.

3.2 Path planning state of the art

An environmental map is required in the path planning, to the establishment of an accurate spatial location description of various objects in the environment in which the drone will be located, including obstacles, road signs, and so on. The purpose of constructing the environmental map is to help the mobile robot plan an optimal path from the starting point to the target point in the established environment model with obstacles [20]. There are many methods for set up an environment model for mobile robot path planning. Some models mainly include a grid decomposition map, quad split graph and a visibility graph.

After the environmental map is built, global path planning is carried out. Algorithms of global path planning are mainly divided into two types: sampling-based algorithms and search planning methods.

The most significant sampling-based motion planning algorithms are Probabilistic RoadMaps (PRMs) and Rapidly-exploring Random Trees (RRTs), both connect points sampled randomly from the state space but differ in the way they construct a graph connecting these points. Other sampling-based planners include Expansive Space Trees (EST) and Sampling-based Roadmap of Trees (SRT). The latter combines the main features of multiple-query algorithms such as PRM with those of single-query algorithms such as RRT and EST [21].

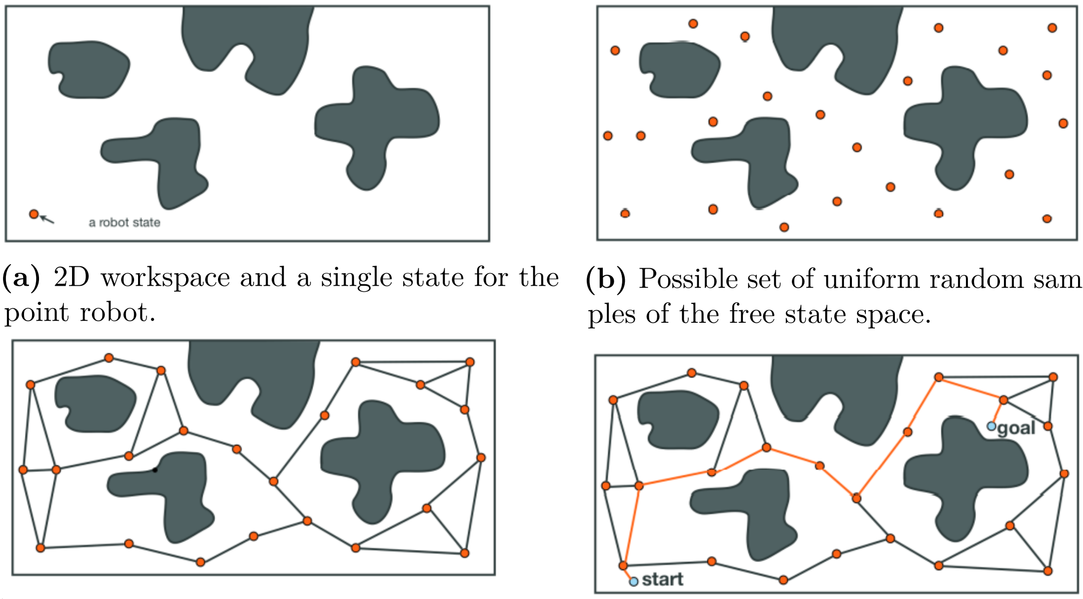
Methods based on graph search algorithms are Dijkstra algorithm, A* (includes heuristics in the Dijkstra algorithm) and Ant Colony algorithm (ACO), to name a few. These algorithms have been analyzed in the research of Xinting Hu et al [22] where, instead of minimizing the total distance, the object has been to find the risk-cost-effective path. These algorithms have been applied over a finite discretization (based, e.g., on a grid, or a cell decomposition of the configuration space) that is generated offline. With respect to the sampling-based planning algorithms, the search planning algorithms often encapsulate the distance in the heuristic function and may not consider the risk a path contains [22]. Thus, sampling-based algorithms are preferred, since the optimality of the search algorithms depends only on the grid resolution, and the number of grid points grows exponentially with the dimensionality of the state space, so does the (worst-case) running time of these algorithms.

3.3 Comparison of sampling-based planners

In this subsection it has been explained how the sampling-based algorithms construct the tree graph and the optimal path is found.

All algorithms take as input a path planning problem $(\chi_{free}, x_{init}, \chi_{goal})$, an integer $n \in \mathbb{N}$, and a cost function $c : \Sigma \rightarrow \mathbb{R} \geq 0$, if appropriate. These inputs are shared with functions and procedures called within the algorithms. All algorithms return a graph $G = (V, E)$, where $V \subset \chi_{free}$, card $(V) \leq n{+}1$, and $E \in V \times V$. The solution of the path planning problem can be easily computed from such a graph, e.g., using standard shortest-path algorithms [21].

Probabilistic RoadMaps (PRM) algorithm is aimed at multi-query applications. In Figure 3.1 is illustrated the behavior of this algorithm. It consists of a pre-processing phase, in which a roadmap is constructed by attempting connections among randomly-sampled points (Figure 3.1c), and a query phase, in which paths connecting initial and final conditions through the roadmap are sought (Figure 3.1d) [21].



(a) 2D workspace and a single state for the point robot.

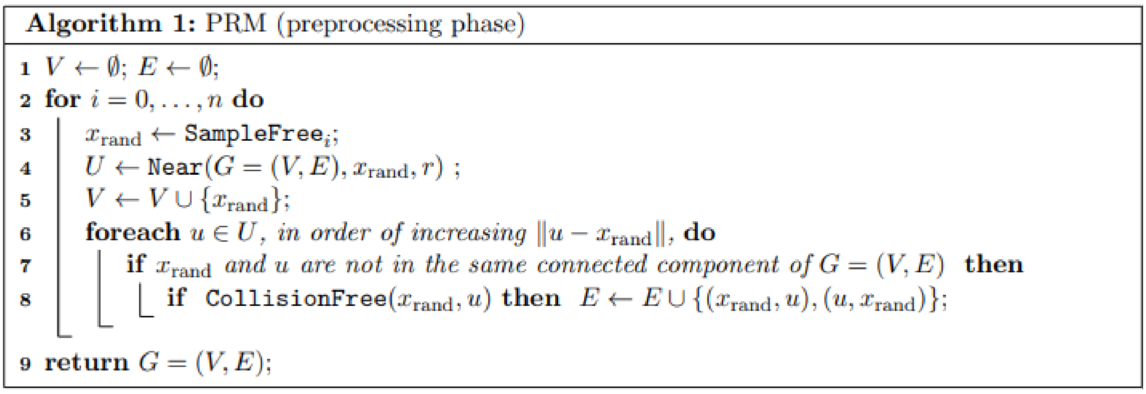
(b) Possible set of uniform random samples of the free state space.

(c) Connection of samples that are close to each other, by a straight path.

(d) Optimal path.

Figure 3.1: Start and goal connected and the shortest path in the graph is found [23].

In Figure 3.2 it is outlined how the algorithm works in the preprocessing phase. Starting from an empty graph (line 1), at each iteration a random point x_{rand} is sampled (line 3) and it is added to the vertex set V (line 5). Then the random point, centered in a ball of radius r, is connected with other vertices which are inside this ball in order of increasing distance from x_{rand} (lines 6 to 8), using a simple local planner (e.g., straight line connection). Successful (i.e., collision-free) connections result in the addition of a new edge to the edge set E (line 9).



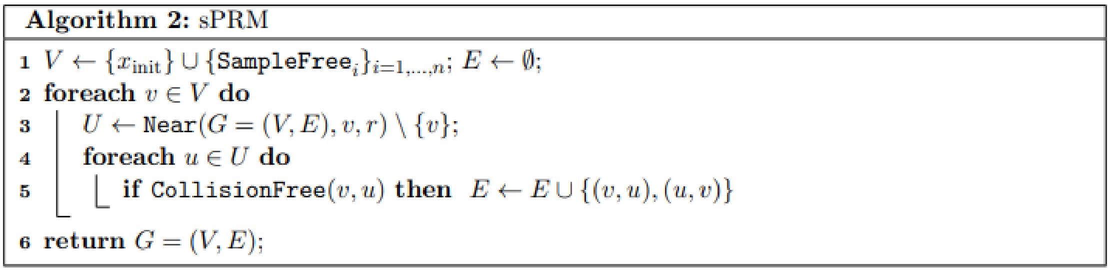
Algorithm 1: PRM (preprocessing phase)

```
$V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
for $i = 0, \dots, n$ do
    $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
    $U \leftarrow \texttt{Near}(G = (V, E), x_{\text{rand}}, r)$ ;
    $V \leftarrow V \cup \{x_{\text{rand}}\}$;
    foreach $u \in U$, in order of increasing $\|u - x_{\text{rand}}\|$, do
        if $x_{\text{rand}}$ and $u$ are not in the same connected component of $G = (V, E)$ then
            if $\texttt{CollisionFree}(x_{\text{rand}}, u)$ then $E \leftarrow E \cup \{(x_{\text{rand}}, u), (u, x_{\text{rand}})\}$;
return $G = (V, E)$;
```

Figure 3.2: Outline of the Preprocessing phase of PRM algorithm.

Moreover, to avoid unnecessary computations connections between x_{rand} and vertices in the same connected component are avoided. Hence, the roadmap constructed by PRM is a forest, i.e. a collection of trees.

Furthermore, the PRM algorithm is probabilistically complete, and such that the probability of failure decays to zero exponentially with the number of samples used in the construction of the roadmap in some applications, computing a roadmap a priori may be computationally challenging or even infeasible [21].
A simplified version of PRM algorithm (sPRM) connects points using a similar logic of PRM but stands out since the vertex set is initialized with the initial conditions (line 1), and the connection between vertices is allowed in the same connected component (lines 2 to 5). Its outline algorithm is reported in Figure 3.3.

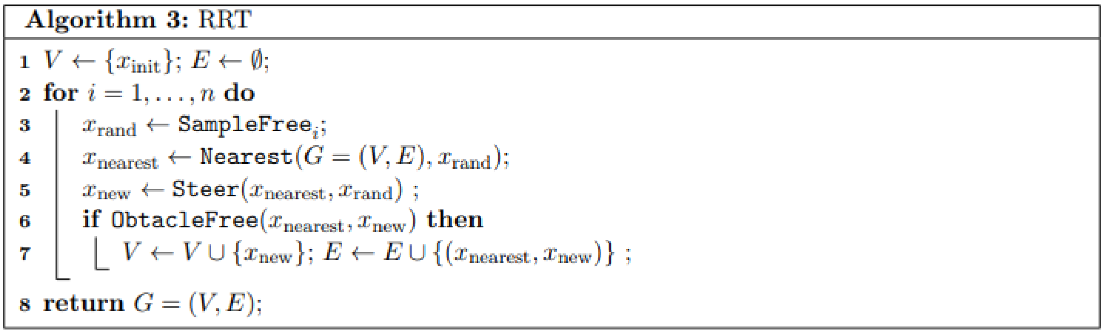


Algorithm 2: sPRM

```
1 $V \leftarrow \{x_{\text{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\dots,n}$; $E \leftarrow \emptyset$;
2 foreach $v \in V$ do
3     $U \leftarrow \texttt{Near}(G = (V, E), v, r) \setminus \{v\}$;
4     foreach $u \in U$ do
5         if $\texttt{CollisionFree}(v, u)$ then $E \leftarrow E \cup \{(v, u), (u, v)\}$
6 return $G = (V, E)$;
```

Figure 3.3: Outline of sPRM algorithm.

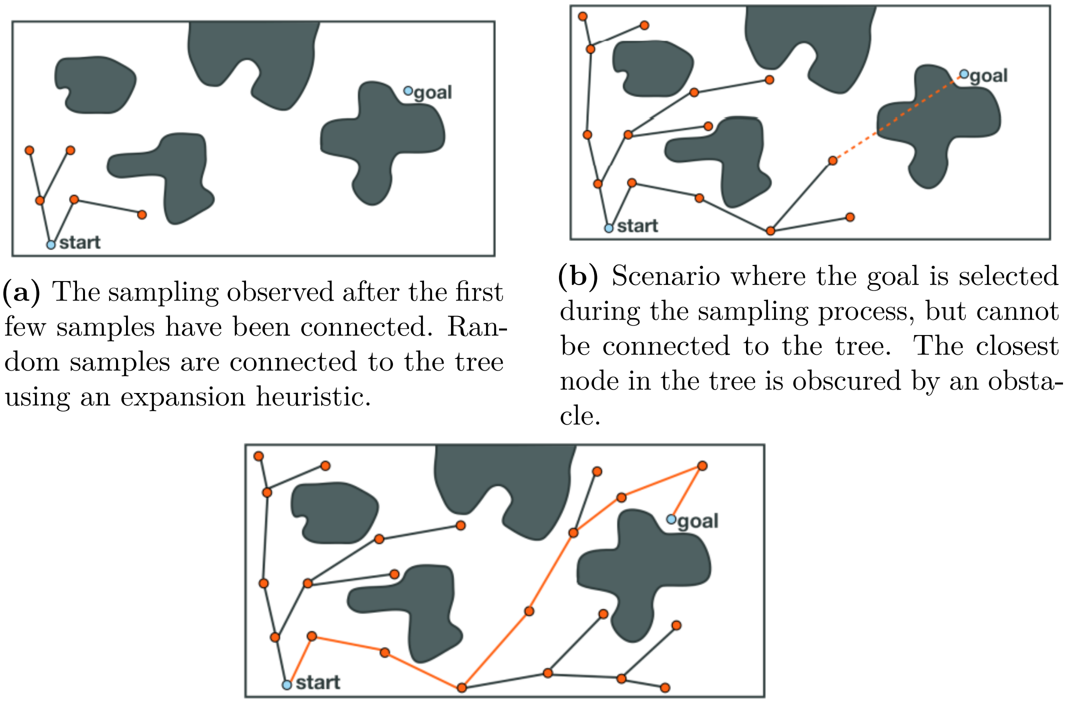
Rapidly-exploring Random Tree (RRT) algorithm is aimed at single-query applications. This algorithm (outlined in Figure 3.4) is initialized with a graph that includes the initial state as its single vertex and not edges (line 1), likewise the PRM algorithm. At each iteration, a point x_{rand} is sampled (line 3). Only if the connection of the new sample, with the nearest vertex ($v \in V$) in the tree, is successful, the point x_{rand} is added to the vertex set (lines 4 to 7) . The iteration ends when the tree contains a node in the goal region. Moreover, in the absence of obstacles, the tree constructed in this way is an online nearest neighbor graph [21].



```
Algorithm 3: RRT
1 $V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
2 for $i = 1, \ldots, n$ do
3     $x_{\text{rand}} \leftarrow$ SampleFree$_i$;
4     $x_{\text{nearest}} \leftarrow$ Nearest$(G = (V, E), x_{\text{rand}})$;
5     $x_{\text{new}} \leftarrow$ Steer$(x_{\text{nearest}}, x_{\text{rand}})$ ;
6     if ObtacleFree$(x_{\text{nearest}}, x_{\text{new}})$ then
7         $V \leftarrow V \cup \{x_{\text{new}}\}$; $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$ ;
8 return $G = (V, E)$;
```

Figure 3.4: Outline of the RRT algorithm.

The RRT algorithm is probabilistically complete with an exponential rate of decay for the probability of failure.
The behavior of the tree-based planners as RRT is exemplified in Figure 3.5 where there is a 2D workspace scenario in which the trend of this algorithm is highlighted. In Figure 3.5a the first few valid samples are connected to the tree. Since it is highly improbable that the sampling process will ever sample the goal state exactly, the methods often bias the expansion of the tree toward the goal state. If it is possible to connect the goal to the existing tree, then the search is complete; a path through the free state space has been found from the start to the goal, Figure 3.5b while in Figure 3.5c it is shown a case where the goal cannot be connected to the tree [23].



(a) The sampling observed after the first few samples have been connected. Random samples are connected to the tree using an expansion heuristic.

(b) Scenario where the goal is selected during the sampling process, but cannot be connected to the tree. The closest node in the tree is obscured by an obstacle.

(c) Scenario where the goal is selected during the sampling process, and is connected to the tree, ending the search.

Figure 3.5: Start and goal are connected with the tree-based planner and the shortest path in the graph is found [23].

The algorithms just analyzed are a basic version that have been upgraded with versions that are asymptotically optimal and computationally efficient.

Optimal Probabilistic RoadMaps (PRM*) is similar to sPRM algorithm with the only difference that the connection radius r is chosen as a function of n (line 3), so it decreases with the number of samples. Moreover, the rate at which the connection radius is scaled is $log(n)/n^{1/d}$ [21].

```
Algorithm 4: PRM*
$V \leftarrow \{x_{\mathrm{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\dots,n}$; $E \leftarrow \emptyset$;
foreach $v \in V$ do
    $U \leftarrow \texttt{Near}(G = (V, E), v, \gamma_{\mathrm{PRM}}(\log(n)/n)^{1/d}) \setminus \{v\}$;
    foreach $u \in U$ do
        if CollisionFree$(v, u)$ then $E \leftarrow E \cup \{(v, u), (u, v)\}$
return $G = (V, E)$;
```

Figure 3.6: Outline of PRM* algorithm.

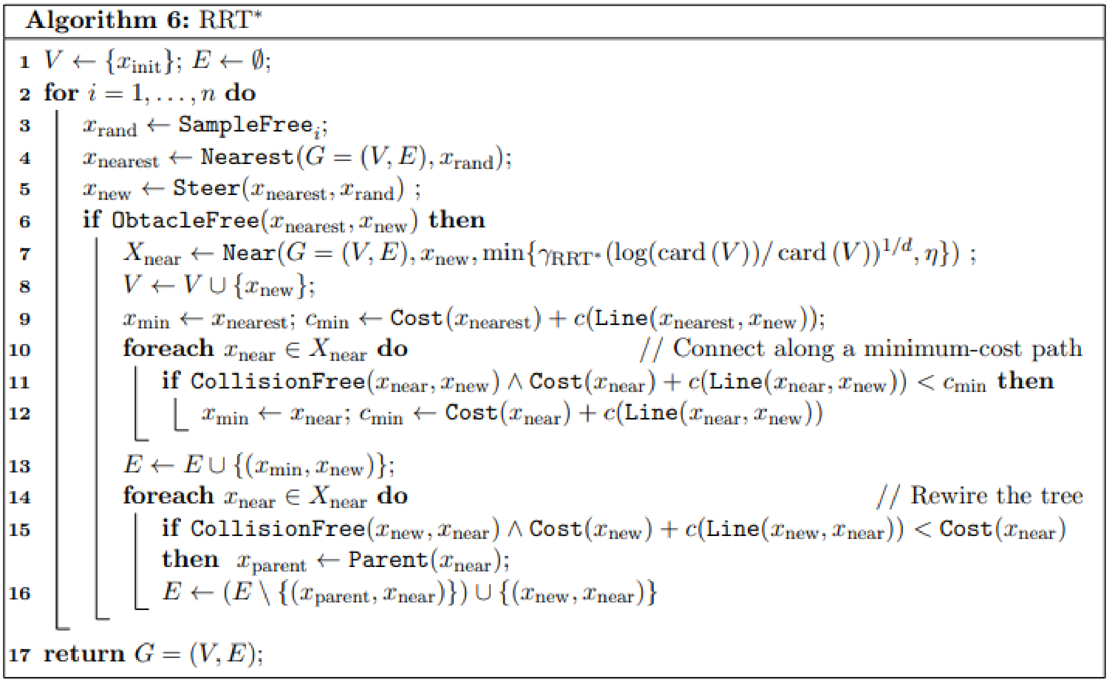
Rapidly-exploring Random Graph (RRG) is opposed to batch algorithm since builds a connected roadmap, possibly containing cycles. It is similar to RRT with the difference that, every time a new point x_{new} is added to the vertex set V, the connections are attempted from all other vertices in V that are within a ball of radius r (lines 7 - 8). Moreover, for each successful connection, a new edge is added to the edge set E (lines 9 - 10).

```
Algorithm 5: RRG
1  $V \leftarrow \{x_{\mathrm{init}}\}$; $E \leftarrow \emptyset$;
2  for $i = 1, \dots, n$ do
3      $x_{\mathrm{rand}} \leftarrow \texttt{SampleFree}_i$;
4      $x_{\mathrm{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\mathrm{rand}})$;
5      $x_{\mathrm{new}} \leftarrow \texttt{Steer}(x_{\mathrm{nearest}}, x_{\mathrm{rand}})$ ;
6      if ObtacleFree$(x_{\mathrm{nearest}}, x_{\mathrm{new}})$ then
7          $X_{\mathrm{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\mathrm{new}}, \min\{\gamma_{\mathrm{RRG}}(\log(\mathrm{card}\,(V))/\,\mathrm{card}\,(V))^{1/d}, \eta\})$ ;
8          $V \leftarrow V \cup \{x_{\mathrm{new}}\}$; $E \leftarrow E \cup \{(x_{\mathrm{nearest}}, x_{\mathrm{new}}), (x_{\mathrm{new}}, x_{\mathrm{nearest}})\}$ ;
9          foreach $x_{\mathrm{near}} \in X_{\mathrm{near}}$ do
10             if CollisionFree$(x_{\mathrm{near}}, x_{\mathrm{new}})$ then $E \leftarrow E \cup \{(x_{\mathrm{near}}, x_{\mathrm{new}}), (x_{\mathrm{new}}, x_{\mathrm{near}})\}$
11 return $G = (V, E)$;
```

Figure 3.7: Outline of RRG algorithm.

Comparing Figure 3.7 with Figure 3.4 it is clear that, for the same sampling sequence, the RRT graph is a subgraph of the RRG graph. In particular, the two graphs share the same vertex set, and the edge set of the RRT graph is a subset of that of the RRG graph [21].

Optimal RRT (RRT*) is a variant of RRG algorithm. Indeed, the formation of cycles is avoided, by removing edges that are not part of a shortest path from the initial state to a vertex (line 1). This ensures that vertices are reached through a minimum-cost path. Moreover, maintaining a tree structure rather than a graph is not only economical in terms of memory requirements, but may also be advantageous in some applications, due to, for instance, relatively easy extensions to motion planning problems with differential constraints, or to cope with modeling errors.



```
Algorithm 6: RRT*
1  $V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
2  for $i = 1, \ldots, n$ do
3      $x_{\text{rand}} \leftarrow$ SampleFree$_i$;
4      $x_{\text{nearest}} \leftarrow$ Nearest$(G = (V, E), x_{\text{rand}})$;
5      $x_{\text{new}} \leftarrow$ Steer$(x_{\text{nearest}}, x_{\text{rand}})$ ;
6      if ObtacleFree$(x_{\text{nearest}}, x_{\text{new}})$ then
7          $X_{\text{near}} \leftarrow$ Near$(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT*}}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$ ;
8          $V \leftarrow V \cup \{x_{\text{new}}\}$;
9          $x_{\text{min}} \leftarrow x_{\text{nearest}}$; $c_{\text{min}} \leftarrow$ Cost$(x_{\text{nearest}}) + c($Line$(x_{\text{nearest}}, x_{\text{new}}))$;
10         foreach $x_{\text{near}} \in X_{\text{near}}$ do                // Connect along a minimum-cost path
11             if CollisionFree$(x_{\text{near}}, x_{\text{new}}) \land$ Cost$(x_{\text{near}}) + c($Line$(x_{\text{near}}, x_{\text{new}})) < c_{\text{min}}$ then
12                 $x_{\text{min}} \leftarrow x_{\text{near}}$; $c_{\text{min}} \leftarrow$ Cost$(x_{\text{near}}) + c($Line$(x_{\text{near}}, x_{\text{new}}))$
13         $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
14         foreach $x_{\text{near}} \in X_{\text{near}}$ do                // Rewire the tree
15             if CollisionFree$(x_{\text{new}}, x_{\text{near}}) \land$ Cost$(x_{\text{new}}) + c($Line$(x_{\text{new}}, x_{\text{near}})) <$ Cost$(x_{\text{near}})$
               then $x_{\text{parent}} \leftarrow$ Parent$(x_{\text{near}})$;
16             $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
17 return $G = (V, E)$;
```

Figure 3.8: Outline of RRT* algorithm.

In the outline of RRT* algorithm, shown in Figure 3.8, it is evident that points are added to the vertex set V in the same way of RRT and RRG algorithms (lines 3 to 8). However, to maintain the tree structure, not all the feasible connections are inserted in the edge set E. Two cases are possible: (i) an edge is created from the vertex in X_{near} that can be connected to x_{new} along a path with minimum cost (lines 10 to 13), and (ii) new edges are created from x_{new} to vertices in X_{near}, if the path through x_{new} has lower cost than the path through the current parent (lines 14 to 16).

3.4 Chosen algorithm

The algorithm chosen in this thesis is RRT* due to its advantages, especially with respect to RRT. Indeed, it has been proved by Karaman and Fazzoli [21], that:

- RRT algorithm does not converge to the optimum solution, while RRT* algorithm improves the paths in the tree to lower cost ones, thus converges to the optimal solution;

- The variance over different RRT runs approaches 2.5, while that of the RRT* approaches zero;
- in presence of obstacles, formerly RRT and RRT* rapidly explore the state space, however, as the number of samples increase, the RRT* improves its tree computing an asymptotically optimal graph and, then, obtaining paths with smaller cost.

All these dissimilarities are highlighted in Figure 3.10 and Figure 3.9.

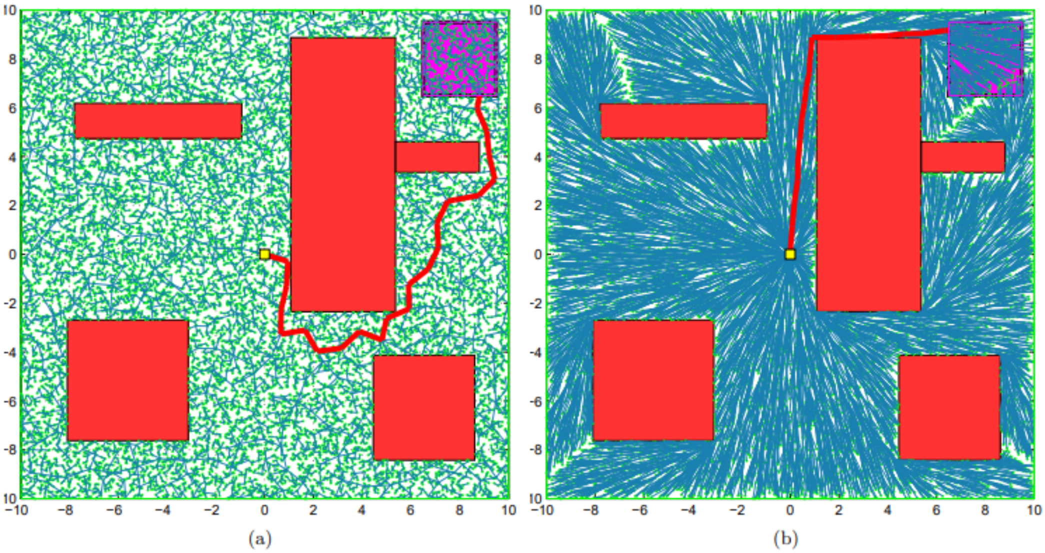
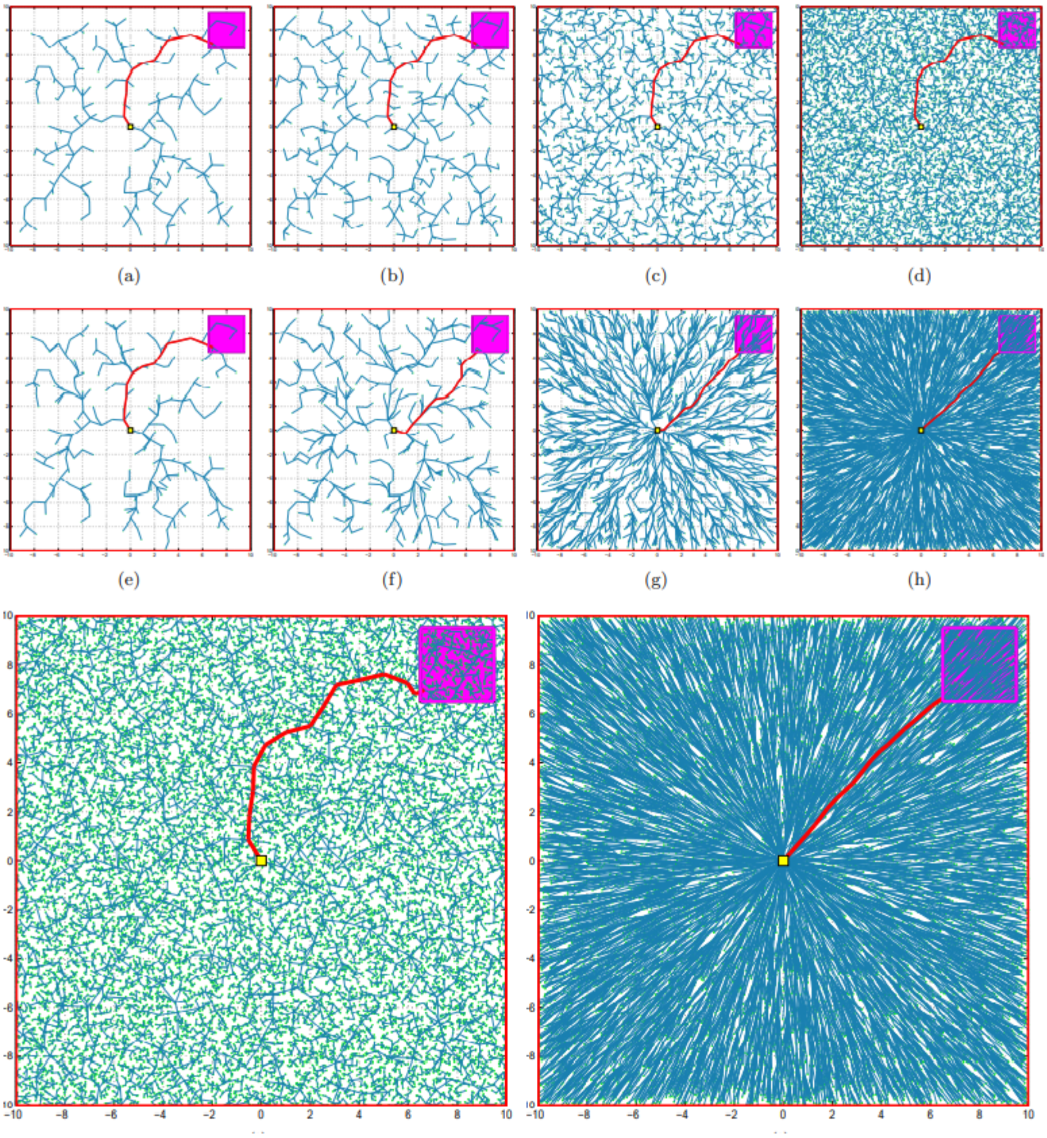


Figure 3.9: A Comparison of the RRT (shown in (a)) and RRT* (shown in (b)) algorithms on a simulation example with obstacles. Both algorithms were run with the same sample sequence for 20,000 samples.



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

(e) (f) (g) (h)

Figure 3.10: A Comparison of the RRT* and RRT algorithms on a simulation example with no obstacles. Both algorithms were run with the same sample sequence. Consequently, in this case, the vertices of the trees at a given iteration number are the same for both of the algorithms; only the edges differ. The edges formed by the RRT algorithm are shown in (a)-(d) and (i), whereas those formed by the RRT* algorithm are shown in (e)-(h) and (j). The goal regions are shown in magenta (in upper right). The best paths that reach the target in all the trees are highlighted with red [21].

Chapter 4

Algorithm

This chapter aims at analyzing the characteristics of the algorithm developed and used for the simulation. In the first part, the implementation of the RRT* algorithm is described, in the second section the process behind the generation of a risk map is portrayed, whilst in the last section the optimization functions, that have been implemented in this thesis, are expounded.

4.1 RRT*: implementation of the algorithm

The algorithm has been implemented in C++ as an executable process in ROS (Robot Operating System), an open-source meta-operating system for robots. The path planning is implemented using the Open Motion Planning Library (OMPL), an open source library specialized in sampling-based motion planning and it consists of many state-of-the-art algorithms [24].
The Open Motion Planning Library provides an abstract representation for all of the core concepts in motion planning, including the state space, control space, state validity, sampling, goal representations, and planners [23].
The hierarchy of the major components of OMPL, and how they are related, are shown in Figure 4.1.

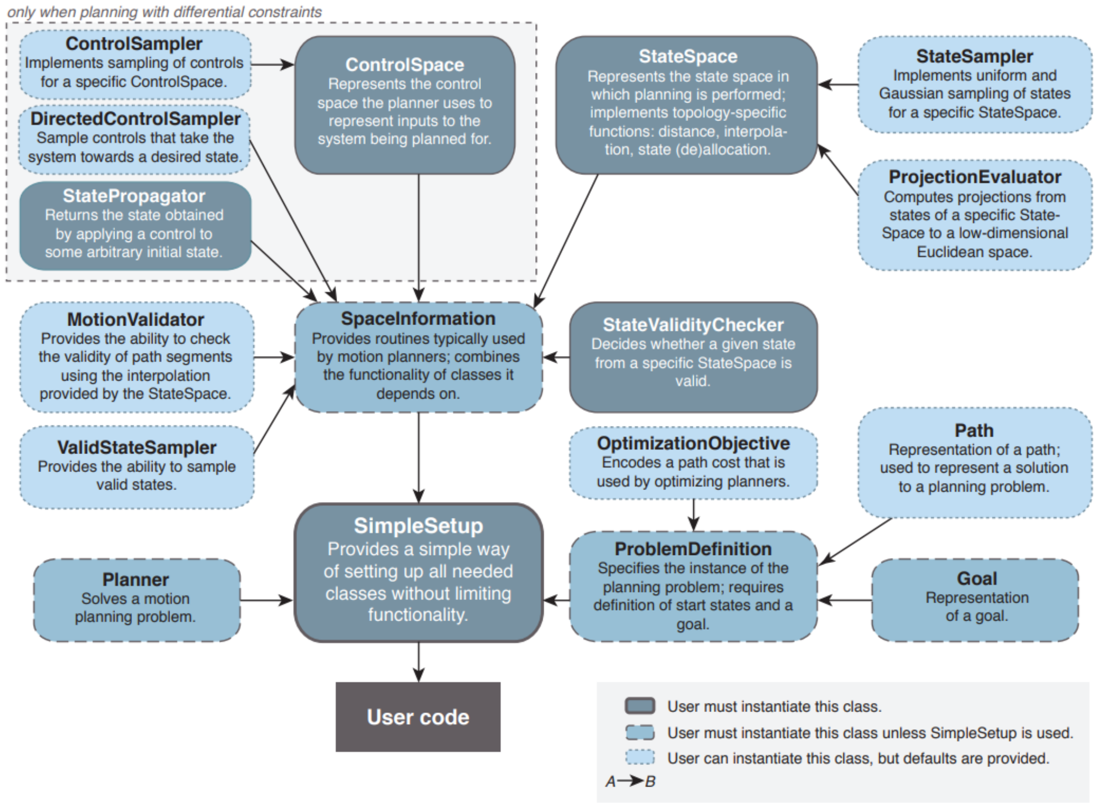

only when planning with differential constraints
ControlSampler
Implements sampling of controls for a specific ControlSpace.
DirectedControlSampler
Sample controls that take the system towards a desired state.
StatePropagator
Returns the state obtained by applying a control to some arbitrary initial state.
ControlSpace
Represents the control space the planner uses to represent inputs to the system being planned for.
StateSpace
Represents the state space in which planning is performed; implements topology-specific functions: distance, interpolation, state (de)allocation.
StateSampler
Implements uniform and Gaussian sampling of states for a specific StateSpace.
ProjectionEvaluator
Computes projections from states of a specific StateSpace to a low-dimensional Euclidean space.
MotionValidator
Provides the ability to check the validity of path segments using the interpolation provided by the StateSpace.
SpaceInformation
Provides routines typically used by motion planners; combines the functionality of classes it depends on.
StateValidityChecker
Decides whether a given state from a specific StateSpace is valid.
ValidStateSampler
Provides the ability to sample valid states.
OptimizationObjective
Encodes a path cost that is used by optimizing planners.
Path
Representation of a path; used to represent a solution to a planning problem.
Planner
Solves a motion planning problem.
SimpleSetup
Provides a simple way of setting up all needed classes without limiting functionality.
ProblemDefinition
Specifies the instance of the planning problem; requires definition of start states and a goal.
Goal
Representation of a goal.
User code
User must instantiate this class.
User must instantiate this class unless SimpleSetup is used.
User can instantiate this class, but defaults are provided.
A→B

Figure 4.1: The hierarchy of the high level components of OMPL. Objects highlighted in dark blue are required to be instantiated by the user [23].

A subset of the classes used in OMPL, which provides the components that are necessary to solve a planning query and that appear in Figure 4.1, are:

- **StateSampler** provides methods for uniform and Gaussian sampling in the most common state space configurations. Moreover, they are methods for sampling in Euclidean spaces and the 2D and 3D rotations.
- **NearestNeighbors** is an abstract class used to provide a common interface to the planners to perform a nearest neighbor search among samples in the state space.
- **StateValidityChecker** determines if the configuration of a single state collides with an environment obstacle and respects the constraint of the robot.
- **MotionValidator** is analog to the local planner. It checks if the motion of the robot between two states is valid, i.e. if it is collision free and all the constraints of the robot are respected.

- **OptimizationObjective** useful to implement different cost functions. It provides an abstract interface to the relevant operations with costs that the planners need. This class has been utilized to implement the different optimization strategies studied for this thesis.

- **ProblemDefinition** specifies a motion planning query. A start state and goal configuration for the robot, and the optimization objective to meet are defined in this class. Moreover, this class is retrieved for solutions to motion planning queries.

The just itemized classes have default implementations that are sufficient for general planning. However, there are objects that must have instantiate, like:

- **Geometric Planning** requires the definition and allocate the state space object for the robot; moreover it provides a start and goal configuration in that space to define the motion planning query. Any planner defined in the geometric namespace can then be employed to solve the motion planning query.

- **Planning with Controls** needs the same definition of the geometric planning. Moreover, valid control inputs via a ControlSpace object must be defined by the user, and a method for computing how the system state changes when applying valid controls, must be provided.

- **SimpleSetup** provides a method to encapsulate the various objects necessary to solve a geometric or control query in OMPL. When using SimpleSetup, the user only supplies the state space, start state, and goal state (and state propagator for planning with controls). Additionally, SimpleSetup allows for the retrieval of all of these subcomponents for further customization. This means that SimpleSetup does not inhibit any native functionality of OMPL, and ensures that all objects are properly created before planning takes place, [23].

4.2 Risk-map

The input of the path planning used in this thesis is the risk map that defines the risk to people on the ground associated to accidents with unmanned aircraft, over a specific area. The risk is defined as the probability to cause a casualty expressed in hours of flight (h^{-1}), considering different descent events [24].

The risk map is a two-dimensional location based map, in which each cell is associated with a specific risk value. They are defined considering the risk level, the no-fly zones and the presence of obstacles at the flight altitude.

The risk map can be defined as a matrix R, whose element $R(x, y)$ imposes the risk of UAV overflying over the position (x, y).
The generation of the risk map, used for evaluating the risk of the city of Turin, is well explained in the work of Primatesta et al [25]. In their work the risk map results to be the combination of the following layers:

- **Population density layer** defines the population density and distribution in the map; it has effect on the probability to impact with a person on ground. In the considered framework, it is a 2D location based map with the same risk map dimension, in which each cell holds the population density value at the relative location, expressed in people/m^2. It can be defined as a matrix D, whose element $D(x, y)$ is linked to a cell of the grid. In Turin city the average population density is 6939 people/km^2;

- **Obstacles layer** establishes the height of obstacle in the map; the probability of collision with buildings is not directly taken into account, since it is used to define area in which the UAV cannot overfly. It is a 2D location based map represented as a matrix O, in which each cell $O(x, y)$ assesses the maximum height of objects in that area. In addition, by rating all obstacles that are under the flight altitude, this layer is employed to define the sheltering factor layer;

- **Sheltering layer** is a positive number used to delineates the sheltering level, supplied by objects (as tree or buildings) to people in each cell of the map. These elements reduce the kinetic energy at impact, and thus the probability of failure. It is a location based map defined by the matrix S, each element $S(x, y)$ corresponds to a cell of the grid, which consistent element have one of the value in the table 4.1;

Area	Sheltering factor
No obstacles	0
Sparse trees	2.5
Vehicles and low buildings	5
High buildings	7.5
Industrial building	10

Table 4.1: Sheltering factor value for each area.

- **No-fly zone layer** identifies in the map the area in which the UAV cannot overfly. These zones are established by National regulation agencies, nature sensitive areas and security zones. This layer is a location based map, defined by a matrix F, where each cell $F(x, y)$ assumes the value of 0 if the flight is allowed over that area, -1 otherwise.

Population density and sheltering factor layers, with drones and environment characteristics, are the inputs of the risk assessment, which is the fist step to generate the risk-map. This phase provides the creation of four maps, called event maps, one for each descent event taken into account. Then, these event maps are combined with no-fly zones and obstacles layers to generate the final risk map. This architecture is illustrated in Figure 4.2.

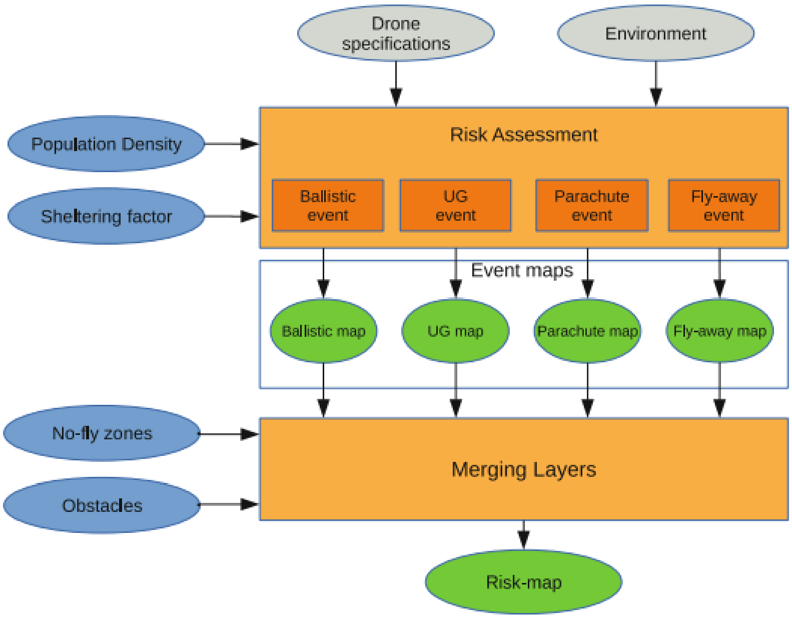


Figure 4.2: Architecture of the risk map generation [25].

The computation of the risk is done considering a probabilistic approach, thus the risk is the probability to have a casualty ($P_{casualty}(x, y)$) in time by three conditional events:

1. P_{event}: probability to loss the control of the vehicle with uncontrolled descent (which are ballistic descent, uncontrolled glide, parachute descent and fly-away) crash on ground;
2. $P_{impact}(x, y)$: probability to impact a person when the UAV crashes on ground due to an uncontrolled descent;
3. $P_{fatality}(x, y)$: probability of a fatal impact with a person.

Therefore, the probability to have a casualty is:

$$P_{casualty}(x, y) = P_{event} \cdot P_{impact}(x, y) \cdot P_{fatality}(x, y)$$

It is computed for each descent event type and for every cell of the map. A deeper analyses on how these events are measure is made.

Probability to loss the control of UAV (P_{event}) is computed for each of the four descent type. For each of them, the probable impact area is settled by a mathematical model that also considers drone specifications and initial conditions.

- **Ballistic Descent** occurs when UAV cannot generate its lift, so the drone is only affect by gravity and drag force. Its model is:

  $$m\dot{\boldsymbol{v}} = mg - c|\boldsymbol{v}|\boldsymbol{v}$$

  where m is the mass of the drone, c is a constant drag coefficient, drag area and air density, g is the gravitational acceleration, and v is the velocity vector of the vehicle.

- **Uncontrolled Glide** happens when the UAV enters in an uncontrolled descent governed by the glide ratio (γ) (ratio between the horizontal traveled distance and vertical one). A distinction between the type of drone must have taken into account: in a fixed wing aircraft, it occurs when a loss of thrust or a loss of power for the flight control surface takes place; on the other hand, for a single rotorcraft, it is verified when a loss of thrust on the main rotor and the aircraft descends with autopiloted autorotation descent; the same could happen with multi rotor drones.
  Knowing the flight altitude (h), the distance covered by each UAV during an uncontrolled glide is:

  $$distance(h) = \gamma \cdot h$$

- **Parachute descent** takes place when UAV begins a descent with a fully deployed parachute. Its effect is changed by the wind that modifies the descent direction and velocity.

- **Fly-away** succeeds in the presence of a complete loss of operator control authority, while the autopilot on board maintains the vehicle stable. In that case, the UAV may flight in any direction as long as the autonomy of the vehicle is exhausted or until the UAV crashes on ground.
  This event is modeled as a normal distribution with the mean value centered in the UAV position.

Probability to impact a person (P_{impact}) is defined as the probability that the aircraft hits at least one person after the uncontrolled impact on ground:

$$P_{impact}(x, y) = \rho(x, y) \cdot A_{exp}$$

where ρ is the population density , that may be expressed with the population density layer D, and A_{exp} is the area exposed to the crash (*lethal area*), defined as:

$$A_{exp}(\theta) = 2(r_p + r_{UAV})\frac{h_p}{tan(\theta)} + \pi(r + r_p)^2$$

in which r_p and h_p are the average radius and the height of a person, r_{UAV} is the radius of the vehicle and θ is the impact angle on ground. The value of the angle θ is the only one that may assume different values based on the descent event, so it is the only parameter that it is not constant.
The UAV could impact any point of the exposed area, so the probability of impacting a person can be expressed using the expected value of the considered zone and the expected value of the population density in the layer D based on a georeferenced two-dimensional PDF:

$$P_{impact}(x, y) = \sum_{x,y} PDF \cdot D(x, y) \cdot A_{exp}(\theta(x, y))$$

Probability of a fatal impact with a person ($P_{fatality}$) is computed considering the kinetic energy at impact and, to reduce significantly the probability of fatality, the sheltering factor that, by means of the obstacles, reduces the kinetic energy at impact. It is expressed by the two-dimensional PDF:

$$P_{fatality}(x, y) = \frac{1 - k}{1 - 2k + \sqrt{\frac{\alpha}{\beta}}\Big[\frac{\beta}{E[E_{imp}(x, y)]}\Big]^{\frac{3}{E[S(x,y)]}}}$$

in which :

- $k = min\Big[1, \frac{\beta}{E[E_{imp}(x,y)]}^{\frac{3}{E[S(x,y)]}}\Big]$;
- $S(x, y)$ is the sheltering factor at a precise position, extracted by using the sheltering factor layer S;
- $E_{imp}(x, y) = \frac{1}{2}m \cdot v_{imp}(x, y)^2$ is the kinetic energy at impact, evaluated according to the uncontrolled descent type. In the formula m is the mass of UAV and v is the velocity at the impact, that, for the uncontrolled glide it is the glide speed considered, whereas, for the ballistic descent it is the combination of its horizontal and vertical velocities;

- α is the impact energy to obtain 50% of the fatality probability when $S(x, y) = 6$;
- β is the impact energy, when $S(x, y)$ reaches 0, to cause a fatality;
- $E[\cdot]$ represents the expected value:

$$E[E_{imp}(x,y)] = \sum_{x,y} PDF \cdot E_{imp}(x,y) \quad and \quad E[S(x,y)] = \sum_{x,y} PDF \cdot S(x,y)$$

Eventually, the risk map is generated by the combination of the resulting event maps with the no-fly zone and the obstacles layers. This happens when occurs that $F(x, y) = -1 \vee O(x, y) \geq h$, in which h is the flight altitude of the drone. Moreover, the risk map elements (x, y) are defined as:

$$R(x,y) = \begin{cases} -1, & \text{if flight is not allowed} \\ P_{casualty}(x,y) & \text{if flight is allowed} \end{cases}$$

Hence, the probability if casualty results to be as the combination of

$$P_{casualty}(x,y) = P^{un.glide}_{casualty}(x,y) + P^{bal}_{casualty}(x,y)$$

As the descent events are independent from each other, they can be added to each other. Thus the resultant $P_{casualty}(x, y)$ quantifies the risk involved by the UAV when overfly the element at (x, y) area.

In this thesis the risk-map is generated using the Grid map library [24], a C++ library interfaced with ROS capable of handling the two-dimensional grid maps with multiple data layers.

4.3 Implemented optimization functions

The core of this thesis has been the implementation of optimization strategies to plan a specific flight mission for drone delivery. Four different optimizer have been studied:

1. minimization of the risk cost in the risk map;
2. minimization of the flight time in the risk map;
3. minimization of the risk cost in the risk map, while forcing a constraint on the flight time;
4. minimization of the flight time, while forcing a constraint on the overall risk cost;

These strategies have been associated with the *RRT** algorithm, in order to compute the optimal path that minimizes the object of that specific optimizer. Moreover, each optimization is composed by OMPL functions, applied for the needed goal. The most important are here highlighted:

- **OptimizationObjective** is the constructor. The space information is defined, as well as the parameters used for the computation of the objective (i.e. risk constant and velocity of the drone);
- **stateCost** returns a cost computed evaluating the risks of the risk map defined on the state space at a state s;
- **motionCost** gets the cost that corresponds to the motion segment between two states;
- **motionCostHeuristic** defines an admissible estimate on the optimal cost on the motion between two established states. However the admissible estimate always undervalues the true optimal cost of the motion. The default implementation of this method returns this objective's identity cost, which is sure to be an admissible heuristic if there are no negative costs.
- **costToGo** uses a cost-to-go heuristic to calculate an admissible estimate of the optimal cost from a given state to a given goal.
- **infinityCost** gets a cost greater than all other costs in the OptimizationObjective.
- **PublishGraph** prints on the risk map the tree graph built by the RRT$*$ algorithm.

The functions modified in this thesis has been the *motionCost* function, which returns the risk cost or the flight time of the overall path, depending on the optimization with which it is associated.
The operating principle of the different strategies is explained by means of flowchart, to a better comprehension of how the risk-aware path planning optimization.

Minimization of the risk cost

In order to calculate the risk cost of an area, all the cells of the risk map must be taken into account. Thus, an optimal path can be obtained by considering the cells that have the less risk cost from the starting to the ending point.
This behavior is well explained in the paper of Primatesta [24], from which Figure 4.3 is taken. It shows how the risk is graphically computed: formerly the UAV is

positioned on the cell under analysis; secondly, from the position of the UAV in the map, all the cells interested by the descent event (called **hazardous area**) are computed; lastly, only these cells are taken into account in the computation of the risk cost.

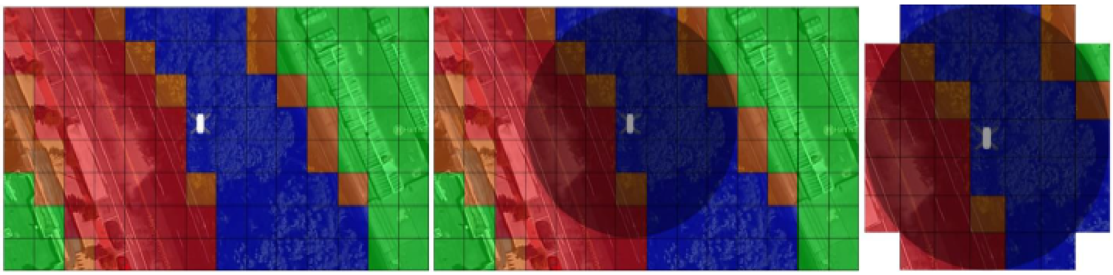


Figure 4.3: From the left to the right: an example of general layer with UAV positioned on the analyzed cell; the hazardous area is computed considering the UAV position: only cells interest by the hazardous area are taken into consideration [24].

Moreover, it has to be highlighted that the direction of the UAV is unknown when generating the risk map, thus the hazardous area is described by a circle which radius is defined by the maximum distance traveled by the UAV in the descent event.

The optimizer that aims at minimizing the total risk cost in the risk map, is implemented in the function **RiskCostOptimizationObjective** that has the property of the *OptimizationObjective* above explained, and it also exploits the combination of other mentioned functions to compute the minimum risk path.
Among all the recalled OMPL functions, the one modified in this thesis is the *motionCost* function, which behavior is explained in the following flow-chart:

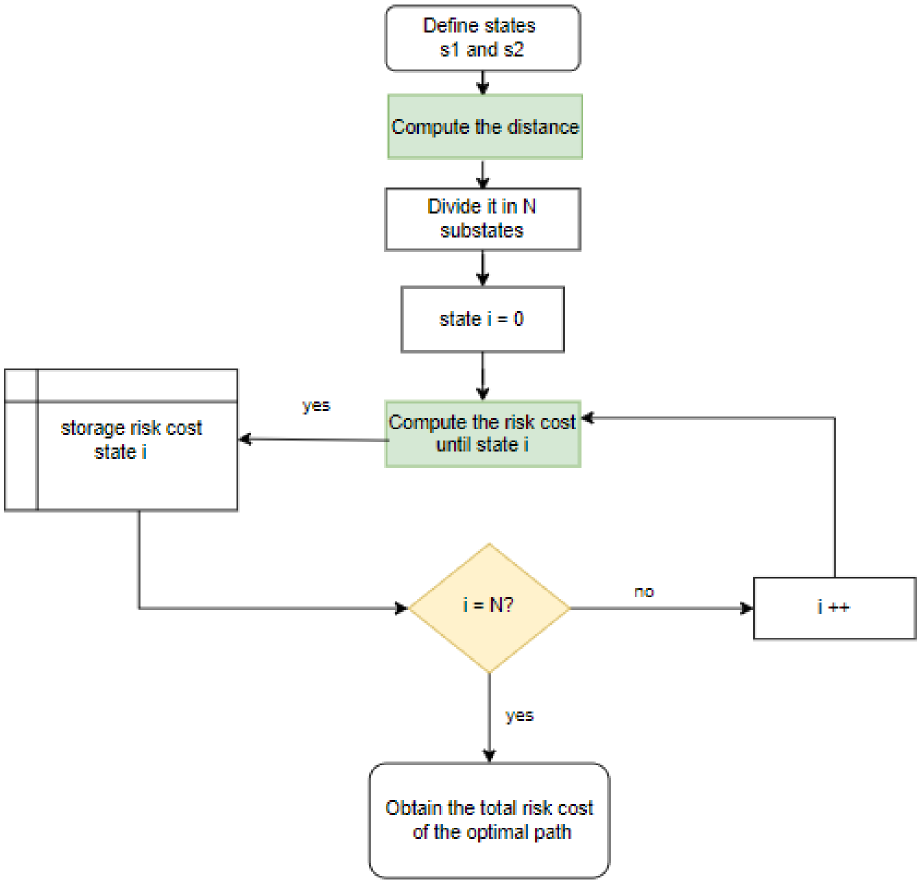


Figure 4.4: Minimization risk cost flowchart.

First, two generics states (s1 and s2) are defined in the stateSpace, thus the distance between them is computed. The *motionCost* function divides the total length in segments of equal dimension and for each of them the risk cost is evaluated. During the path planning, the *motionCost* function is called every time it is wanted to compute the risk cost between two states, in order to connect them in the tree constructed by the RRT*.
This optimizer, working with the path planning algorithm RRT* is able to determine the path, between start and goal, that has the minimum risk cost. Following the above, *RRT** will compute the optimal path evaluating the tree constructed by the *MotionCost*. The optimal path is computed and it is shown on the risk map.

Minimization of the flight time

Once the optimal path that minimizes the risk cost of the total path has been computed, the study has been focused on the detection of the optimal path that minimizes the total flight time. For this scope the function *FlightTimeOptimizationObjective* has been implemented that, as the *RiskCostOptimizationObjective*, is based on the *OptimizationObjective* function and it is made by the combination of the OMPL function. Even in this case, the function *motioncost*, over the all OMPL function used, has been adaptive to reach the desired goal. Its operation is illustrated in the flowchart of Figure 4.5.

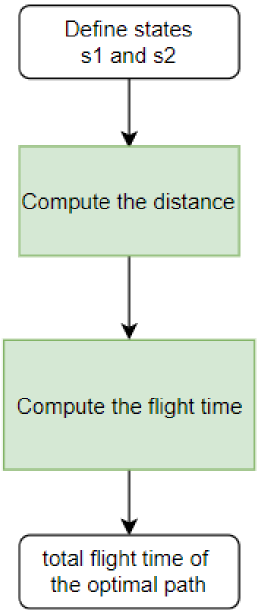


Figure 4.5: Minimization flight time flowchart.

Likewise the previous optimizer, two states are defined and the distance between them is computed. However, this optimization differs from the previous one for the calculation of the flight time overall the total length, without splitting it in small segments. This happens since the flight time is derived from the inverse formula of the velocity, i.e.:

$$velocity = \frac{distance}{time} \quad \rightarrow \quad time = \frac{distance}{velocity}$$

Since the velocity of the drone is a priori known information and the distance is got by the function, it is easily to compute the total flight time of the path.

Minimization of the risk cost, while forcing a constraint on the flight time

A further improvement has been done associating, to the optimizers just described, a constraint. For what regard the minimization of the risk cost a requirement on the flight time has been imposed; to evaluate it, an average solution between the previous optimizers has been found.
Since the aim of this optimization is the minimization of the risk cost, the function *RiskCostOptimizationObjective* has been considered, combined with the OMPL functions. The *motioncost* function has been adapted in the following way:

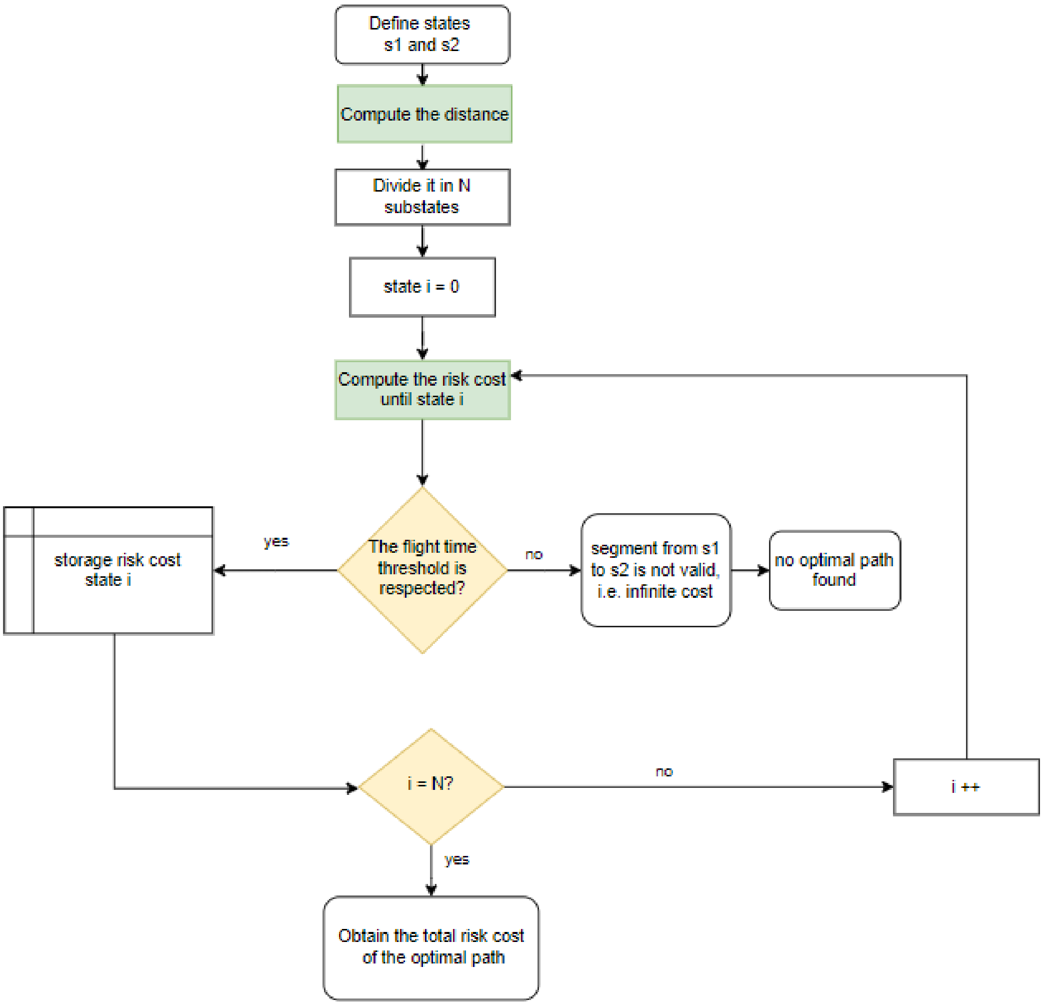

Define states s1 and s2
Compute the distance
Divide it in N substates
state i = 0
Compute the risk cost until state i
The flight time threshold is respected?
yes
storage risk cost state i
no
segment from s1 to s2 is not valid, i.e. infinite cost
no optimal path found
i = N?
no
i ++
yes
Obtain the total risk cost of the optimal path

Figure 4.6: Flowchart of the minimization of the risk cost while forcing a constraint on the flight time.

This flowchart shows a similar behavior of the one in Figure 4.4. The only odds is present after the computation of the risk cost of the segment state: here it is calculated the flight time until this state, by the same formula used for the only optimization of the flight time, and it is evaluated if the constraint on it is still satisfied. If this is verified, the computation proceeds until the goal state is reached, otherwise the computation stops since the segment between states s1 and s2 is deemed invalid and an infinite cost is associated to the computed path, which means that no optimal path can be found.

Minimization of the flight time, while forcing a constraint on the risk cost

The opposite case has been implemented: the optimal path that minimizes the total flight time while considering a constraint on the risk cost.
Similarly to the previous optimizer the function *RiskCostOptimizationObjective* has been used for the computation of the risk cost, whilst the flight time has been measured by means of the formula. The modified *motioncost* function behaves as illustrated by the flowchart of Figure 4.7.

The below flowchart acts exactly has the one in Figure 4.6. The only change is applied on the computation of the flight time for each state, and so the verification of the overall risk cost till that state. Thus, if it is successful, the computation proceeds till the last segment state is reached; if the constraint is no more satisfied, the computation ends and there exist no optimal path from start and goal that is able to satisfy the chosen threshold.

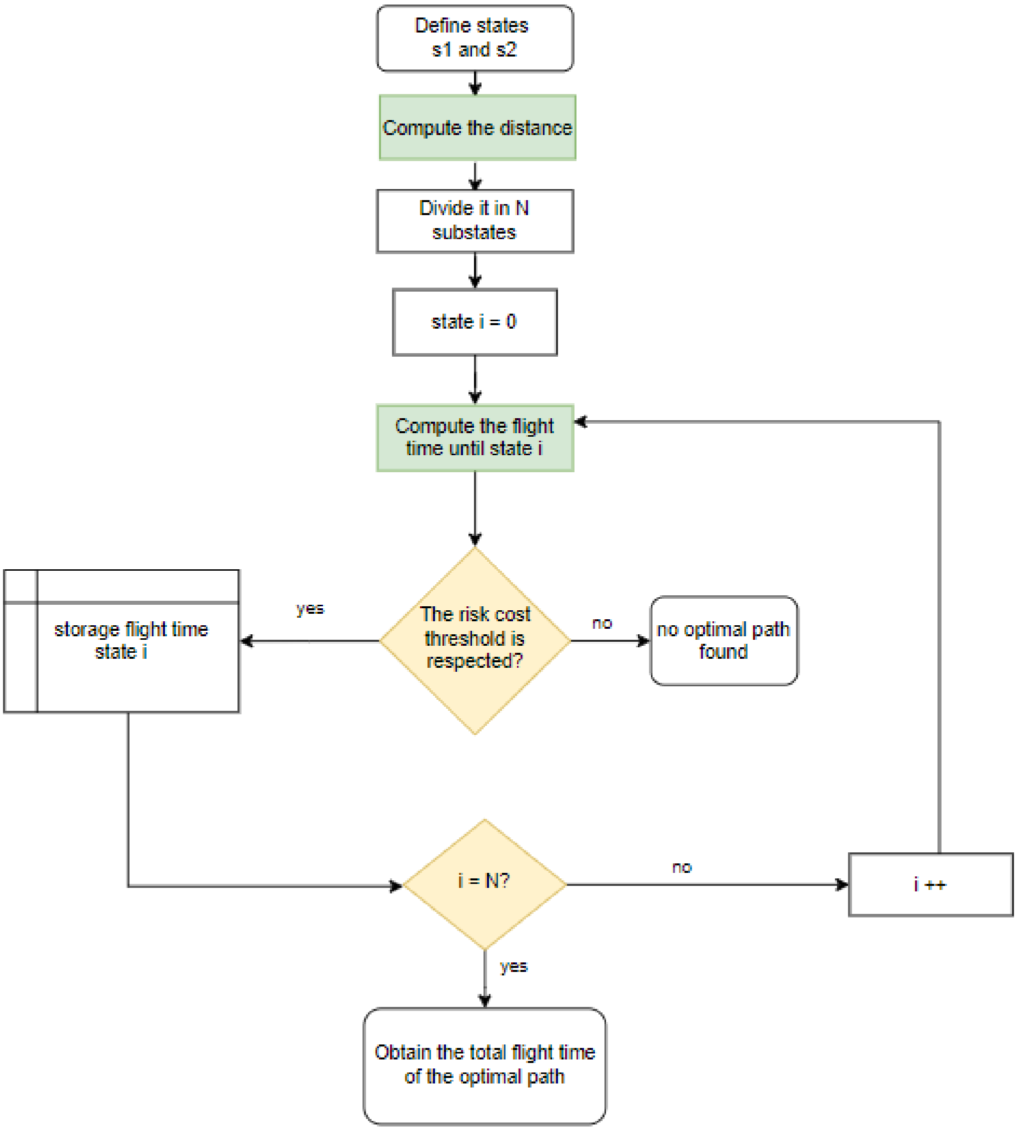


Figure 4.7: Flowchart of the minimization of the risk cost while forcing a constraint on the flight time.

Chapter 5

Simulations

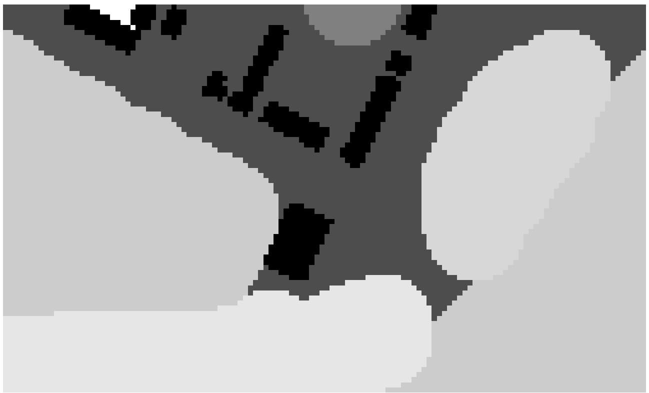
The designed optimizers have been simulated, first on an ideal maps and then on risk maps computed taking into account the city of Turin.
The ideal maps are characterized by different grayscale regions. These regions represent areas with different values of risk, in particular the regions with light grey are the ones with lower values of risk, while the regions with dark grey are the ones with higher values of risk.
The real map is represented by the city of Turin, which is the area to consider for the computation of the optimization path, since it is the site in which the drone as to be applied.
All the tests have been done by means of the framework *ROS* and they have been visualized by the software *Rviz*, which shows the optimal path chosen by the optimizer.

5.1 Tests on ideal maps

The ideal maps have been used as a preliminary simulation, to better understand the behaviour of each optimizers. So by choosing a start point and a goal, the path is computed using the *RRT** algorithm exploiting OMPL and, then, the optimal path is visualized on the ideal maps in *Rviz*.
The four implemented optimizers (*optimization of the risk cost, optimization of the flight time, minimization of the risk cost considering the flight time and the minimization of the flight time considering the risk cost*) have been tested on ideal maps (which resolution is 5 m) with different scales of risk. Different type of simulations have been done to underline the correct behaviour of these implementations, considering various grayscale factors.

5.1.1 Tests with minimization risk

The analysis on the optimization of the risk cost has been done considering the function *RiskCostOptimizationObjective*, since the goal of this optimizer is to find the optimized path with the lowest value of the risk.
The first simulation has been done with the map shown in the Figure 5.1, and the table next to it represents the grayscale factor.



Colour	Risk value
□	0
░	10
▒	20
▓	50
▓	70
■	100

Figure 5.1: Ideal map for simulations of the optimization of the risk cost and its grayscale factor.

A first simulation has been done and the Figure 5.2 shows the optimal path found by this first analyzed optimizer.

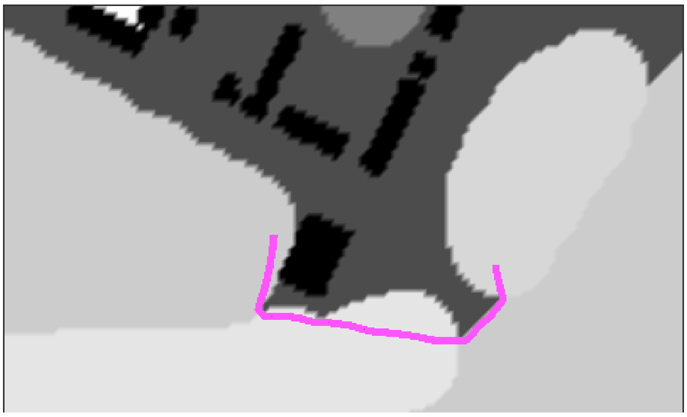


Figure 5.2: path found with the first simulation.

Looking at this Figure and the relative legend it is possible to observe that the optimal path found, which is the magenta line, pass through the regions with lowest risk cost. As result the total risk cost of the path is 5018.093. Without the optimizer on the risk cost, start and goal could have been connected through

a direct line that, however, across the areas with the highest level of risk cost. Consequentially, a significant higher value of risk cost would have been obtained. Nevertheless the total distance of the path increase from 260.01 m, if start and goal would have been connected directly, to 351.700203 m.
A further test has been done considering a longer distance between start and goal. The result is the expected one: a path with a minimum risk cost has been computed, despite the computation of the total path length increase. The Figure 5.3 clarify what has been obtained.

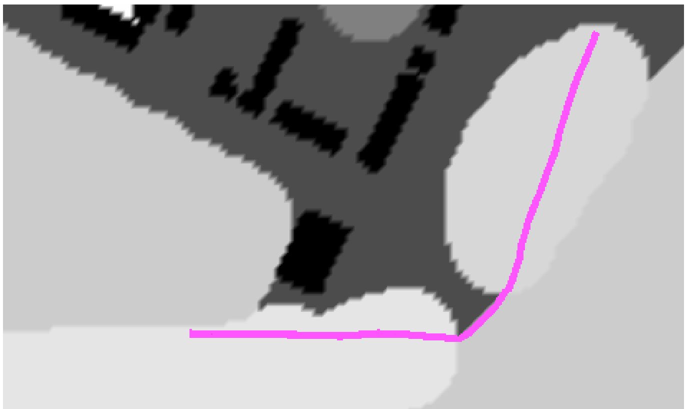
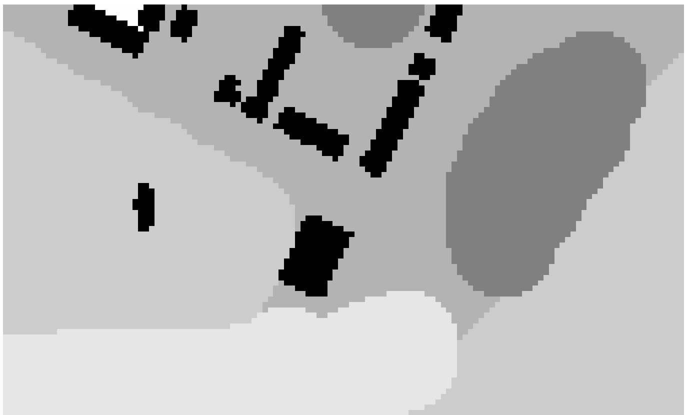


Figure 5.3: Path found with a second simulation.

In this last simulation a total risk cost of 7603.798 has been achieved with a total distance of 559.639561 m instead of 471.851 m of the direct path.

The grayscale factor of the map has been modified, as shown in the Figure 5.4 and more analysis have been conducted.



Colour	Risk value
□	0
░	10
░	20
▒	30
▓	50
■	100

Figure 5.4: Modified ideal map and its grayscale factor.

A further obstacle is added in this last map. It has been employed to underline that the optimization always choose the path with the minimum risk cost.
The map in the Figure 5.5 demonstrates the efficiency of the optimization of the risk cost. The black region is avoided and the computed total risk cost of the path is 5081.673 with a total distance of 334.923753 m rather than 288.507 m.

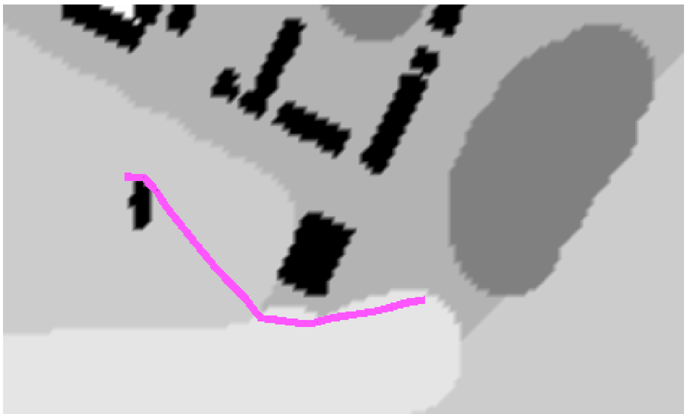


Figure 5.5: Path found with the modified map.

A particular case, interesting to analyze, is the one reported in the Figure 5.6.

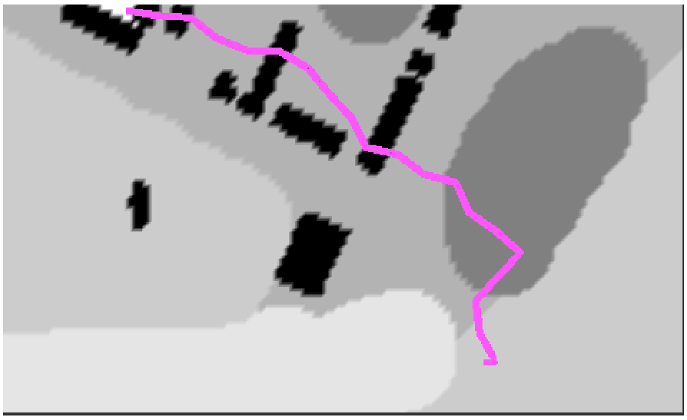


Figure 5.6: Map with no optimal path.

The start has been chosen in the area with zero risk, which is however surrounded by the area with the maximum value of risk. The Figure 5.6 explains that the optimization is not able to find an optimal path with minimum risk cost. In fact, the path illustrated in Figure 5.6 is invalid with an associated cost equal to infinity. This aspect can be better clarified by means of the function *publishGraph*, which prints the graph that the path planning algorithm *ompl* implements, in order to find the optimal path.

Before to apply this function in the studied map, it is reported, in Figure 5.7, a general map where it is easy to understand the behavior of this function.

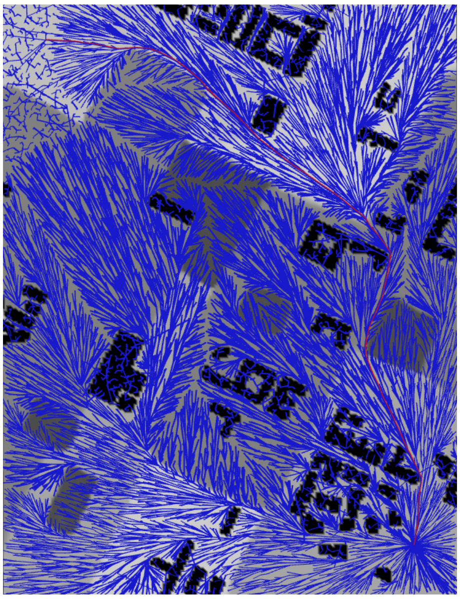


Figure 5.7: Graph implemented by the planning algorithm *ompl*.

The graph shown above represents a lines vector called tree, and so it is called tree graph. The algorithm analyzes all the possible ways to reach the goal using the *MotionCost* function, and arranges the graph in the low risk regions, while it does not care of the areas where the vertices are not promising, namely vertices that, if they are computed, imply a higher risk cost, so it is meaningless to tidy them up. Once the goal is reached, the graph is not more organized and the propagation stops.

This tree graph is organized in a circle centered on the start point and with a radius that is equal to the distance between start and goal. However, if in the optimizer is present a threshold that it is not fulfilled before having reached the goal, the radius of the circle will be arranged until that threshold is still satisfied.

Another interesting observation, that can be made by looking at the graph, is the presence of hollows inside it. These hollows, characterized from the absence of lines, determine the resultant optimal path. Indeed, if the goal is nearest the right side of the vale, the algorithm will choose that path, contrariwise, if it is nearest the left side, the algorithm will go through this other way.

Furthermore, the Figure 5.7 illustrates that, in the black areas, the path is not organized, because of the maximum value of risk. In fact, the algorithm avoids these regions due to the highest probability of failure.

Additionally, it is important to clarify that the path planning algorithm is associated with one of the implemented optimizers. Depending on which it is linked with, the graph will be arranged slightly different.

In the following simulations, it is expected to have a graph which is more or less organized until the goal is reached, due to the absence of the threshold in the optimizer.

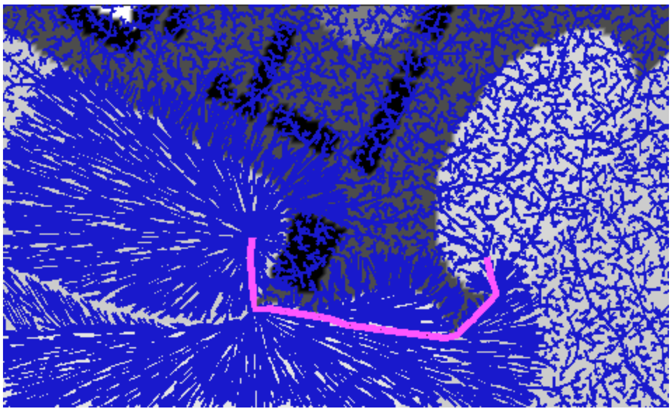


Figure 5.8: Tree graph of the optimal path found with the first simulation.

In the above Figure 5.8 the graph is organized as explained before. Centered in the start, the tree graph branches off to the goal, in an orderly manner in the light regions, where the risk has a low cost, and in a messy way in all the regions in which there is not minimization of the risk cost because of the presence of non-promising vertices. The Figure 5.9 better highlights these two regions by means of the red line that divide them.

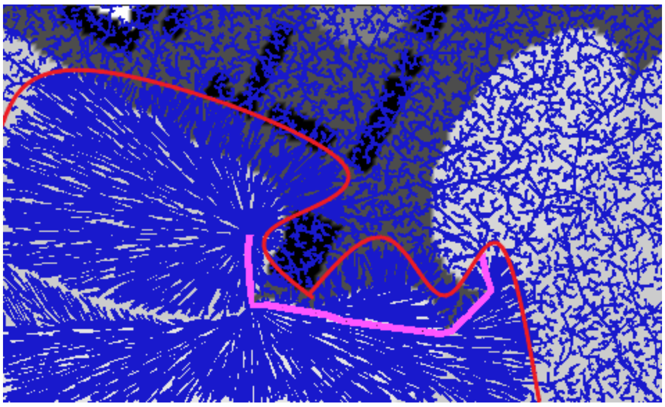


Figure 5.9: Highlighted region after which the graph of the optimal path is not more organized.

It has been demonstrated that, when the path planning algorithm *ompl* is connected with the minimization of the risk cost, it behaves arranging the tree graph in the regions where the minimization of the risk cost is optimal, and so the optimizer on the risk cost works correctly.
In the Figure 5.10 it is now understandable why the found path of the simulation shown in Figure 5.6, is not optimal.

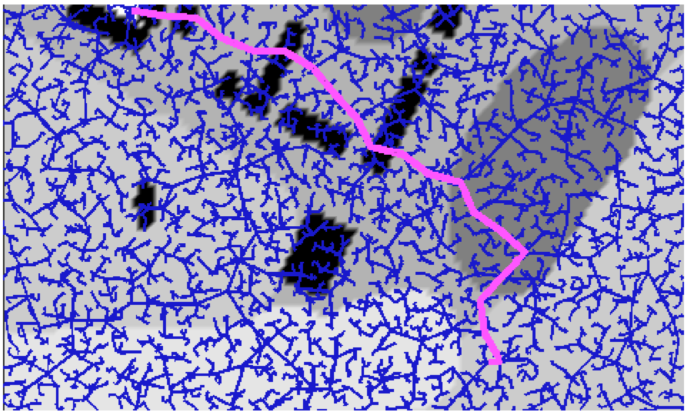


Figure 5.10: Tree graph of the no optimal path.

In the Figure 5.10, it is immediate that the propagation is discontinue, even if the goal is achieved. Indeed, the black area around the start makes impossible to compute a path that reaches the goal, due to the already explained action of the path planning algorithm. Since in the black area the vertices are not promising, the planning will not go through them, and so the propagation is suspended.
In conclusion, the path planning algorithm linked with the optimization of the risk cost, is always able to find the optimal path in terms of risk cost, providing that there is at list one promising vertex along the distance between start and goal.

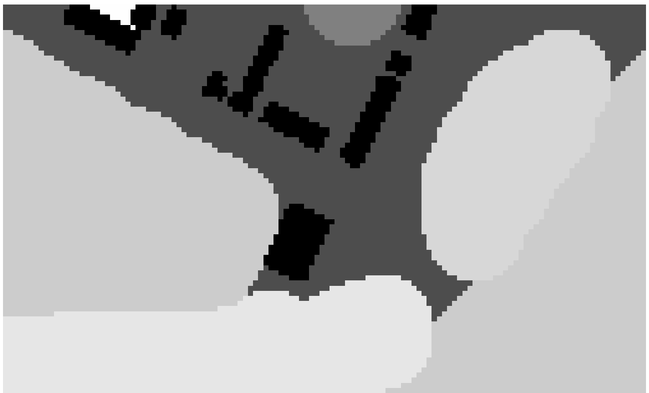
5.1.2 Tests with minimization Flight Time

The evaluation on the minimization of the Flight Time has been done considering the function *FlightTimeOptimizationObjective*, since the goal of this optimizer is to compute the optimal path that minimize the total flight time.
The flight time has been computed using the formula:

$$time\,[s] = \frac{distance\,[m]}{velocity\,[m/s]}$$

For these simulations the velocity has been imposed equal to 1, so the flight time has the same value of the distance between start and goal and it is easily to understand if the code works properly.

The same map of the first optimizer, with the same legend, has been used to evaluate the optimization of the Flight Time.



Colour	Risk value
□	0
░	10
▒	20
▓	50
█	70
■	100

Figure 5.11: Ideal map for the optimization of the flight time and its grayscale factor.

The Figure 5.12 shows a first simulation.

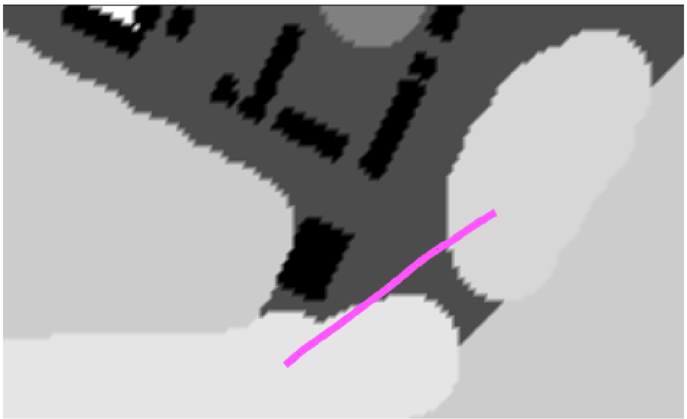


Figure 5.12: Optimal path found for the minimization of the flight time.

Differently from the previous optimizer, where the path passes through the regions with low risk level, now the path directly connects start and goal, crossing regions with high values of risk. The result is a total flight time of 236.301 s and a total risk path of 11645.87.

It seems obvious how this optimizer it is useful when the primary purpose is to delivery the medical pharmaceutics in the smallest time possible.

Even if this is the goal of this optimizer, the planning algorithm *ompl* avoid the black area of the map, i.e. where the risk level has its maximum value. The Figure 5.13 attests this important aspect.

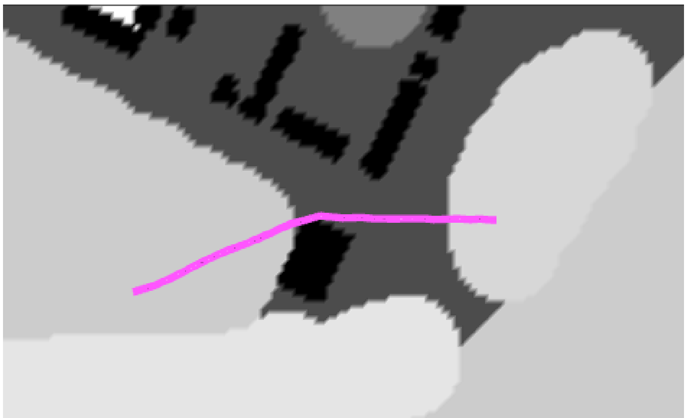
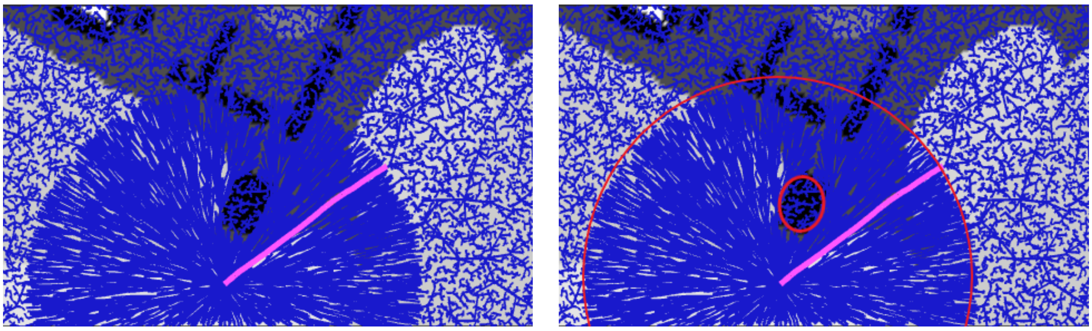


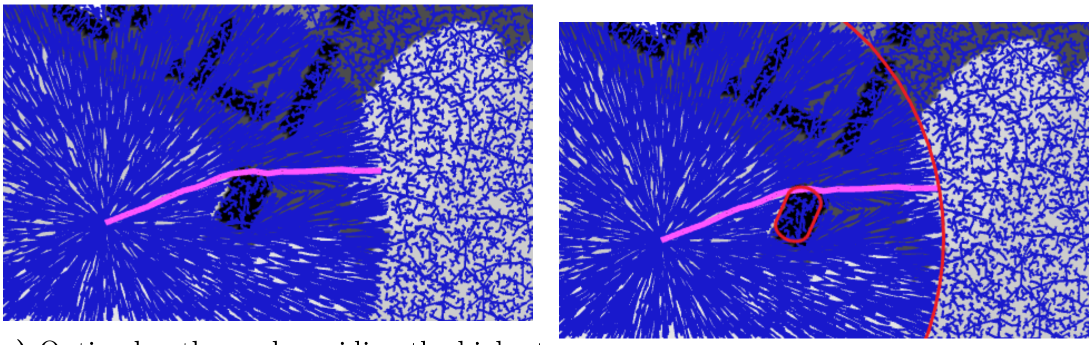
Figure 5.13: Optimal path for the minimization of the flight time avoiding the black area.

The planning algorithm *ompl* has found a path that minimizes the flight time between start and goal, however it has avoided the highest risk level, how described in the previous subsection where its behavior has been well described. The resulting total flight time is 336.291 s, against 319.721 s if it has been passed through the black area, and a total risk cost of the path infinity.
Analyzing the Figure 5.14a and 5.15a, in which the function *publishGraph* has been added, the behavior of the path planning algorithm *ompl* is explained.



(a) Graph of the optimal path. **(b)** Highlighted organization of the graph.

Figure 5.14: Tree graph of the optimal path that minimizes the total flight time.



(a) Optimal path graph avoiding the highest risk level.

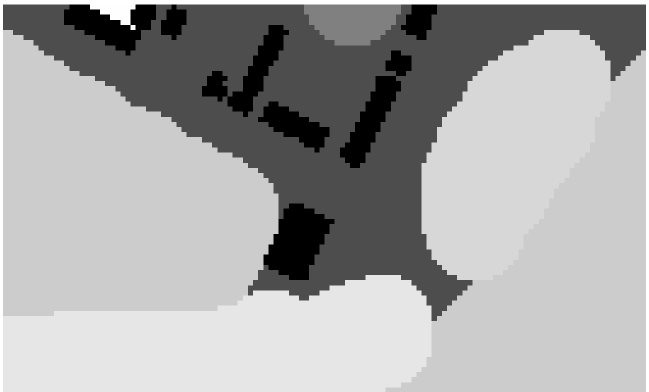
(b) Highlighted regions of the graph.

Figure 5.15: Tree graph of the optimal path that minimizes the total flight time that avoid the area with highest risk cost.

In both the above figures, the tree graph is organized in the circle centered on the start. The red line in Figure 5.14b and 5.15b has been added to have better view of the area after which the graph is no more organized. It is evident in both the figures that in the black area inside this circle, the graph has vertices that are not promising. For this reason, in the Figure 5.13 the found path is not a straight line, but has a deviation at the highest risk level.
In conclusion, it has been proved that the behavior of the optimization of the flight time is the one expected. Moreover the computed path avoids the dangerous area, so this optimizer can be applied when the delivery requires shortest time possible, without concerns of possible risks.

5.1.3 Tests with minimization risk and threshold on Flight Time

This test has been conduct to confirm that the planning algorithm *ompl* is able to visualize the path with minimum risk cost, also taking into account a threshold on the Flight Time.
As for the first studied optimizer, the function *RiskCostOptimizationObjective* has been applied.
In order to better visualize if the program works correctly, the velocity, for the computation of the Flight Time, has been imposed equal to 1. In this way the length of the computed path should be match with the threshold.
The same map (Figure 5.11) used for the simulation of the previous optimizers has been operate for the following simulations. For convenience it is again reported below.



Colour	Risk value
□	0
░	10
▒	20
▓	50
▓	70
█	100

Figure 5.16: Ideal map for the third optimization and its grayscale factor.

A first simulation has been launched imposing as threshold 500 s. It is expected an optimal path that move through the light gray regions, to minimize the risk cost, and with a total distance of the path less or equal to the threshold.

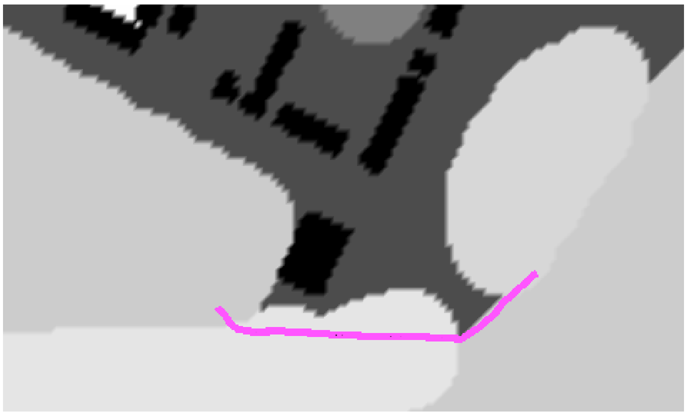


Figure 5.17: Optimal path found with the first simulation.

In the Figure 5.17 it is possible to observe the obtained optimal path. As predicted the line across the light gray regions and the total risk cost of the path is 4226.840, this result is achieved since the total distance of the path is 324.059 m, which is less than the threshold.

Let's know take a longer distance between start and goal, it can be predicted that the optimal path could not be found owing to the constraint on the threshold should not be respected anymore. Two cases could be achieved.
A first case is the one in Figure 5.18 where the found path could seem optimized. However the final solution cost is infinity and, in the low risk level region, the path

is not optimized since the length of the direct line from start and goal is almost equal to the threshold.

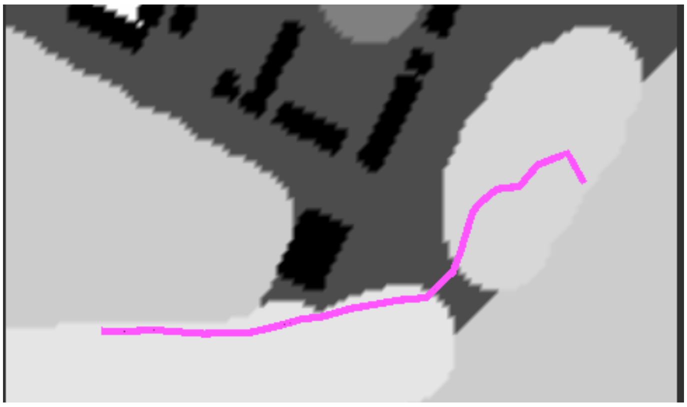


Figure 5.18: Optimal path found with the second simulation.

A second case is in Figure 5.19 below, where the start and goal are connected by a straight line.

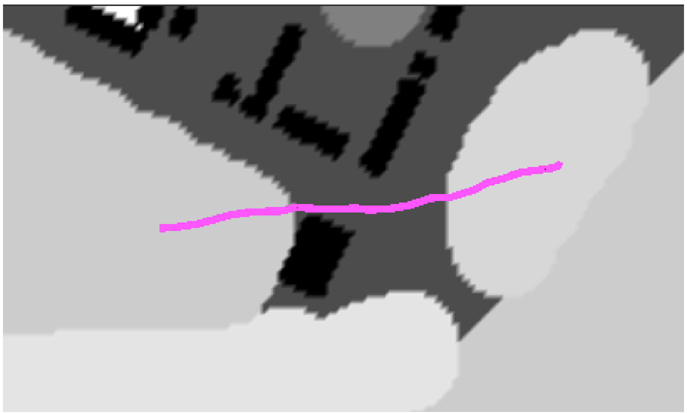
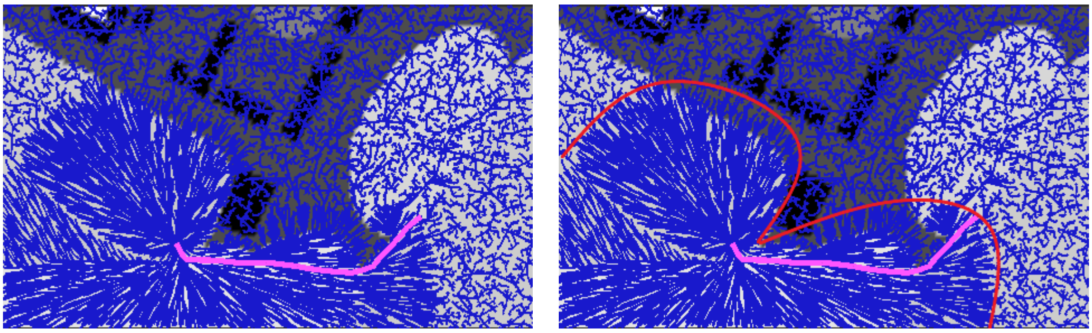


Figure 5.19: Optimal path found with the third simulation.

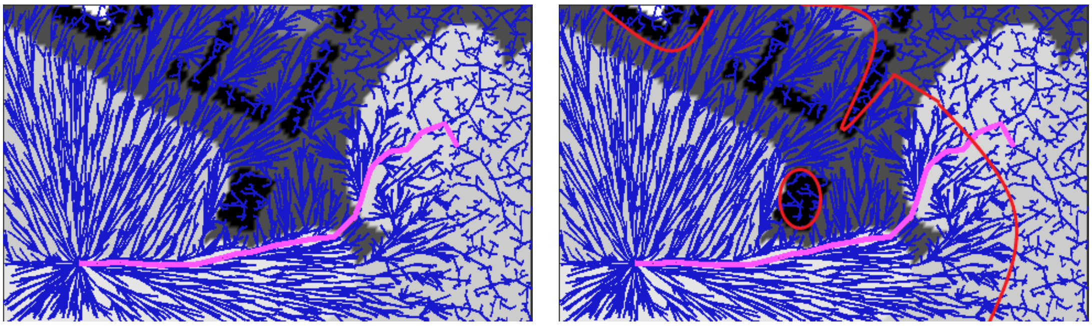
This Figure shows what already said. The found path is not the optimal one because the threshold is exceeded. To satisfy this requirement the planner *ompl* have found a path whose total distance is 378.447 m , although it passes through the region with high level of risk. As result the total risk cost of the path is 13925.375. The function *publishGraph* is helpful to visualize the behavior of the planning algorithm *ompl* in the presence of the threshold. The tree graph of the first simulation is illustrated in Figure 5.20.



(a) Optimal path graph. **(b)** Delimited area of the graph.

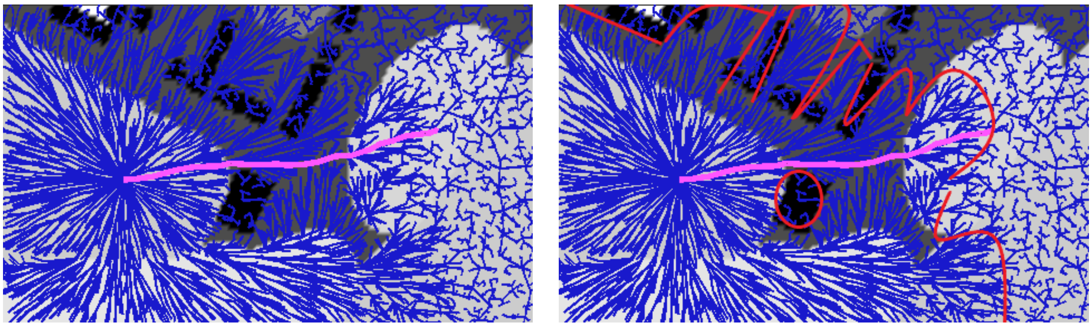
Figure 5.20: Tree graph for the optimal path found with the first simulation.

The obtained graph behaves exactly as the one achieved without a threshold. Centered on the start, a circle with a radius equal to the distance from start and goal is created. Inside it, the branches branch off optimally in low risk regions, whereas they are irregular in high risk areas.
Nevertheless, the presence of the threshold on the flight time implies a graph which is organized until this requirement if satisfied.
This aspect appears in the Figure 5.21a and Figure 5.22a, where the optimal path could not be found due to the overcoming of the threshold.



(a) Graph of the path found. **(b)** Delimited area of the graph.

Figure 5.21: Tree graph of the path found in the second simulation.



(a) Graph of the path found. **(b)** Delimited area of the graph.

Figure 5.22: Tree graph for the path found in the third simulation.

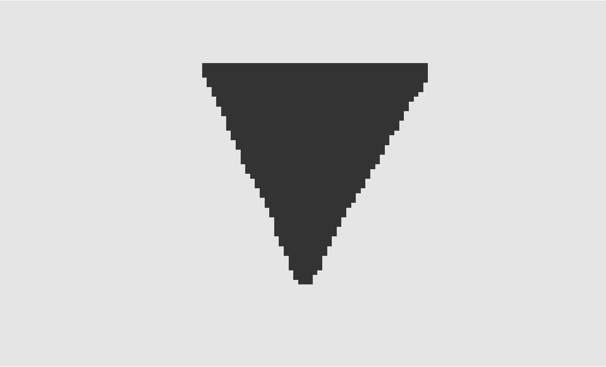
In the two figures the tree graph is organized until a point. As already explained, after that point, the graph is messed up since the constraint on the threshold is no more fulfilled. In Figure 5.21b and Figure 5.22b, the threshold corresponds to the red line. Indeed, In Figure 5.21a, the goal is reached after the red line, it means that the threshold is no more respected, thus the found path is not the optimal one.
In Figure 5.22a, the threshold is respected till the goal is reached but the minimum risk cost is not achieved. Thus it is possible to understand the choice of the planning algorithm *ompl* to prefer the path that pass trough the areas with high risk value.

5.1.4 Tests with minimization of Flight Time and threshold on risk

This test is the opposite of the one just studied. Indeed the path planning algorithm *ompl* is now linked with the optimizer that minimize the flight time of the total path, considering a threshold on the risk cost.
As for the first and third studied optimizer, the function *RiskCostOptimizationObjective* has been applied in order to respect the requirement on the threshold, while the flight time has been computed by means of the formula:

$$time\,[s] = \frac{distance\,[m]}{velocity\,[m/s]}$$

where the velocity has been imposed equal to 1, thus the obtained flight time corresponds to the distance between start and goal.
For this simulation a different map has been utilized, shown in the Figure 5.23.



Colour	Risk value
⬜	10
⬛	80

Figure 5.23: Ideal map for the optimization of the Flight time with threshold on the risk cost and its grayscale factor.

The considered map has a triangular area with high value of risk, placed at the center of a lower risk area. This different map has been employed to underline that, since a threshold on the risk has been imposed, an arched path from the start and the goal should be achieved in order to respect this constraint.
The simulation has been conducted imposing a threshold on the risk cost equal to 5000. A first result is the one in Figure 5.24.

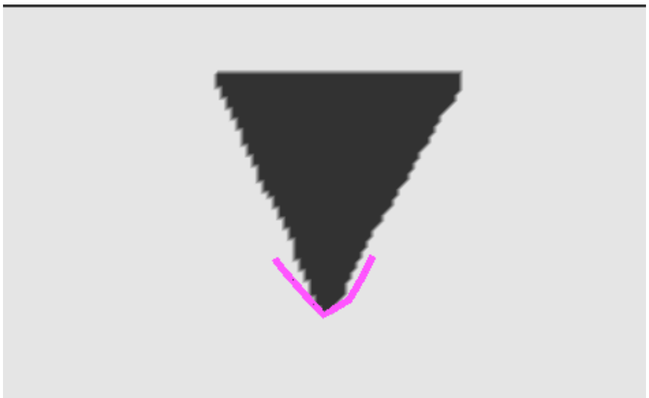


Figure 5.24: Optimal path found for the minimization of the Flight time, considering the threshold on the risk.

The optimal path computed by *ompl* has the predicted arched shape with total risk cost equal to 4555.031 and total distance of the path equal to 148.064 m. Having impose the velocity equal to 1, the optimal flight time is 148.064 s.
The test has proved the correct behavior of the optimizer when the constraint on the risk cost is respected.
The second step has been to demonstrate that, when the threshold cannot be

satisfied, the planning algorithm *ompl* will connect start and goal, by means of an almost straight line.

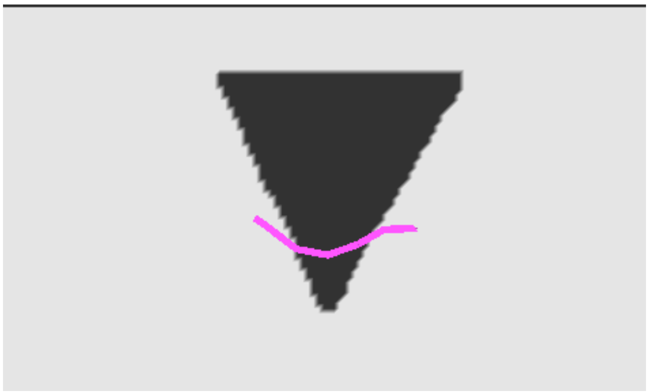
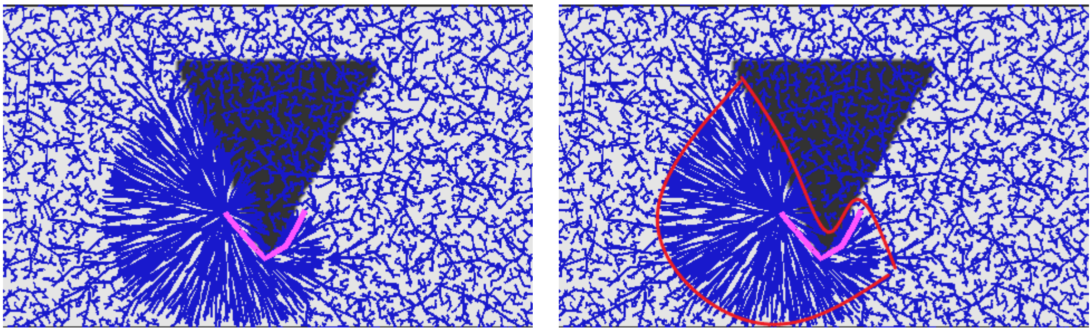


Figure 5.25: Computed path in the presence of exceed threshold.

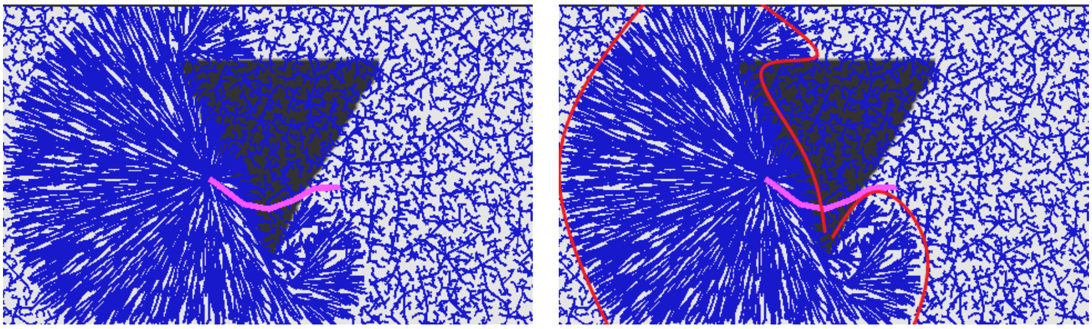
The path of the Figure 5.25 is important to underline how the optimizer works. The requirement on the risk could not be achieved so the planning algorithm has found the path with minimum flight time. The total risk cost is, indeed, 14753.19, almost three times the value of the threshold, while the total distance, and so the total flight time is 179.08 s. However the final solution cost is infinity because the threshold has been exceeded.

As done for the previous optimizers, the function *publishGraph* is again adopted to a full comprehension of this optimizer.



(a) Optimal path graph. **(b)** Delimited area of the graph.

Figure 5.26: Tree graph with the optimal path in the presence of the risk's threshold.



(a) Graph of the path found.

(b) Delimited area of the graph.

Figure 5.27: Tree graph of the not optimize path for the not satisfied threshold.

Around the start point a circle is created for the Euclidean distance and its radius is equal to the threshold. It is now understandable that, when in this circle the tree graph is not organized, it means that the threshold is not respected anymore. As shown in the Figure 5.26 and 5.27, in the area with high value of risk, the threshold is not fulfilled and so the graph is untidy. So, while in Figure 5.26 the goal is achieved avoiding that area, so that the flight time is minimized and the threshold on the risk cost is satisfied, in Figure 5.27 the goal is reached with a path that is not optimized. Indeed, in this last scenario, the propagation ends not because the path arrived at the goal, but due to the overcome of the threshold, as highlighted in Figure 5.27b.
In conclusion, it can been assert that the code of the optimizer works correctly.

5.2 Tests on the real map

Once it has been proved that the implemented optimization strategies return a correct behavior, a realistic scenario of life-saving pharmaceutics delivery has been simulated in the city of Turin, focusing on the area highlighted in Figure 5.28.

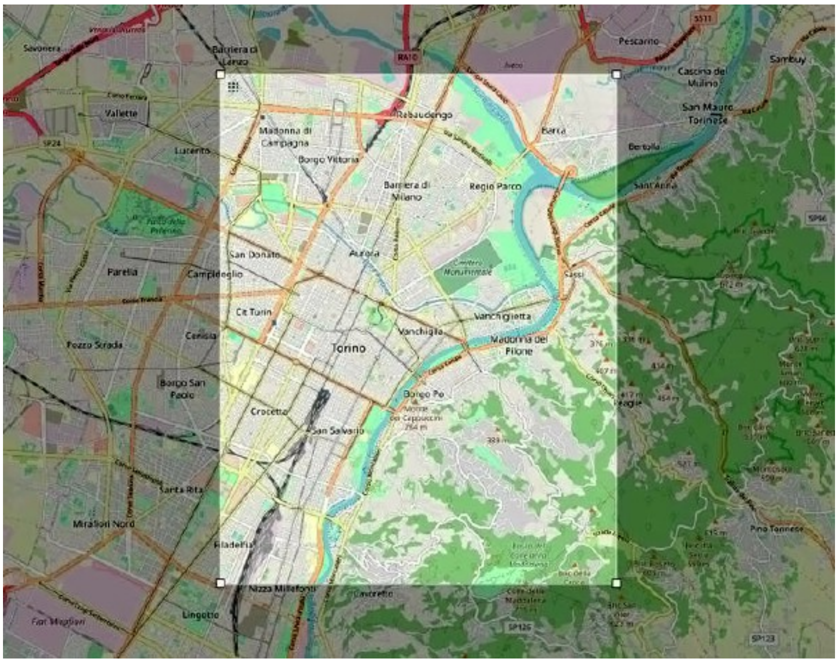


Figure 5.28: Simulation section of Turin map.

The real scenario has been tested considering the information provided in the thesis of Cavallo [7] on the Mavic 2, Matrice 200 and Matrice 600. From the feature of mass, maximum payload and cruise speed of these drones the maps have been generated, taking into account as flight altitude $50\,m$. These features are reported in the table 5.1.

Drone	**Weight (kg)**	**Payload max weight (kg)**	**Cruise speed (m/s)**
Mavic 2	0.90	0.20	20
Matrice 200	4.69	1.45	17
Matrice 600	9.60	6.00	18

Table 5.1: Features of the considered drones.

For each of the considered drone, a risk map has been computed in a grayscale factor risk, which goes from 0 to 100. However, to get the effective risk value for the computed trajectory, a range risk has been computed, comparing the effective maximum and minimum risk of each considered drone, which are reported in the table 5.2.

Drone	Maximum risk ($1/h$)	Minimum risk ($1/h$)
Mavic 2	$3.08 \cdot 10^{-5}$	$2.11 \cdot 10^{-8}$
Matrice 200	$1.94 \cdot 10^{-4}$	$1.01 \cdot 10^{-7}$
Matrice 600	$7.58 \cdot 10^{-4}$	$4.72 \cdot 10^{-7}$

Table 5.2: Maximum and minimum risk of the considered drones.

The chosen range risk is computed evaluating the logarithm of these risk values, thus it is [-9 ; -3]. Starting from this range it has been possible to calculate the effective total risk cost of the optimal path in the following way:

$$risk\_cost = 10^{minimum\_risk + (maximum\_risk - minimum\_risk)\frac{temp\_cost}{100}}$$

where:

- **minimum_risk** and **maximum_risk** are, respectively, the minimum and the maximum value of the chosen range;
- **temp_cost** is the obtained value in the grayscale factor;
- **100** since temp_cost goes from 0 to 100, dividing for 100 returns the risk cost.

So, for example:

- if $temp\_cost = 0$ the minimum risk cost is obtained $risk\_cost = 10^{-9}$;
- if $temp\_cost = 100$ the maximum risk cost is obtained $risk\_cost = 10^{-9+(-3+9)*\frac{100}{100}} = 10^{-3}$.

The resultant maps are reported in Figure 5.29, which shows the risk map of the Mavic 2, Figure 5.30, that illustrate the risk map computed on the Matrice 200 and Figure 5.31, where the risk map on the basis of the features of Matrice 600 is reported.

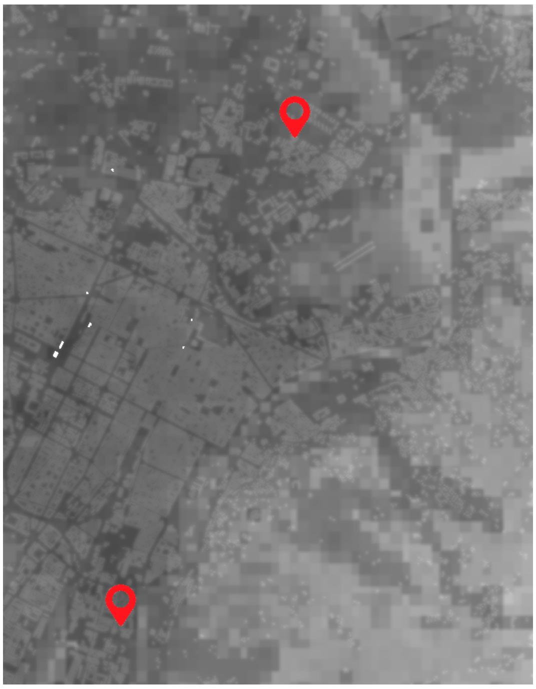


Figure 5.29: Turin city risk map for Mavic 2.

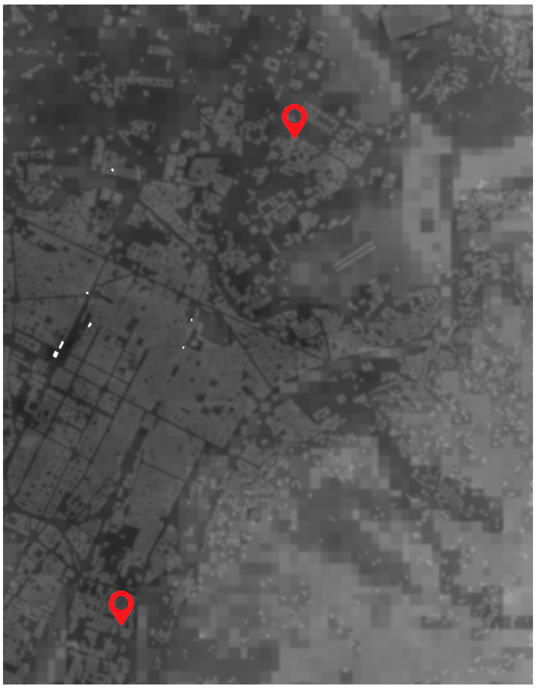


Figure 5.30: Turin city risk map for Matrice 200.

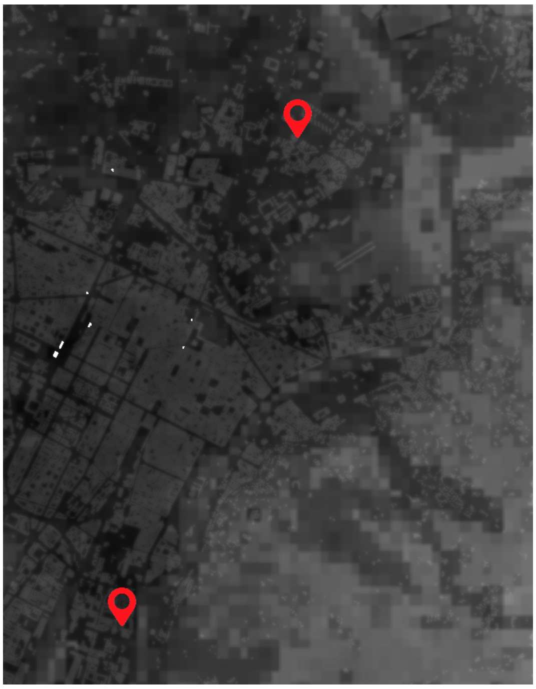


Figure 5.31: Turin city risk map for Matrice 600.

In these maps the white areas correspond to obstacles which cost is 0, indeed the white area has not been considered in the computation of the optimal path since the risk is multiplicative.

The three maps are exactly the same for what regards the considered area, but different for the grayscale intensity. The map in Figure 5.29 is the clearest due to the features of the drone from which it has been created, indeed the Mavic 2 has a small mass and can overfly an urban area with a lower risk of fatality. On the other hand, the map shown in Figure 5.31 is the darkest one since it is referred to the drone Matrice 600, which is the heaviest. In fact, its risk cost is high over an urban area, so the less risky path could only be the one that avoid the city.
As result, minimizing the risk cost, it is expected a fairly straight path with the Mavic 2, while a round wider it is looked for with Matrice 600. A middle behavior can be predicted with Matrice 200, highlighted by the average intensity of its risk map.

In the maps, the start is the *Ospedale Molinette* from *Città della Salute e della Scienza di Torino* and the goal is *Ospedale San Giovanni Bosco*. These two points are the ones highlighted in the maps and correspond to the GPS coordinates: 45°02′19.1″ N 7°40′26.4″ E for the start and 45°05′47.3″ N 7°42′07.5″ E for the goal. The start point is located in the southern part of the city of Turin, in a high populated area without any parks in the surroundings. For this reason the right place for the takeoff and landing of the drone has been found inside the structure, where a large, mostly unused terrace has been selected as the starting point of the simulation and UAVs flight. Similarly, the ending point is situated in a high populated area, but in the North of the city where more parks are nearby the building, so a large unused garden inside the structure has been selected as the ending point of the simulation and UAVs flight.

The implemented optimizers have been simulated in each designed map and the different behaviors have been analyzed and compared to have a realistic scenario of medical delivery.
The behavior of the different optimization strategies is expected to be the same of the one examined in the tests of the ideal maps, namely, in the minimization of the risk cost, an optimal path that crosses the lower area is claimed, whereas, in the minimization of the flight time, an almost straight line is prospected to collect the starting and ending point. Similarly, an intermediate behavior is wanted when considering a threshold value in the minimization. Moreover, the function *PublishGraph* has applied the same mechanism to the computation of the optimal path and the just analyzed trend it is assumed to be achieved.

For each simulation it has been considered 50000 sampling states and, to compare the difference of each optimizer, it has been analyzed the time that the algorithm needs to find the optimal path. Moreover, the solve time has been imposed to a very high value (120 s) because of the large map and, to be sure that the solution is the optimal one, since the algorithm converges to the finest solution with the increasing of the states. Furthermore, the planner range is imposed to 1000 m which correspond to 200 pixel. The planner range is the r ball radius that starts from the defined value and, with the increasing size of the graph, it decreases its dimension to avoid that each state has a high number of neighbors that causes an useless slowdown.

In every simulation, the obtained risk cost is the effective risk of the specific mission. To assert if that mission is safe or not, it is necessary to evaluate the average risk, which is rated as:

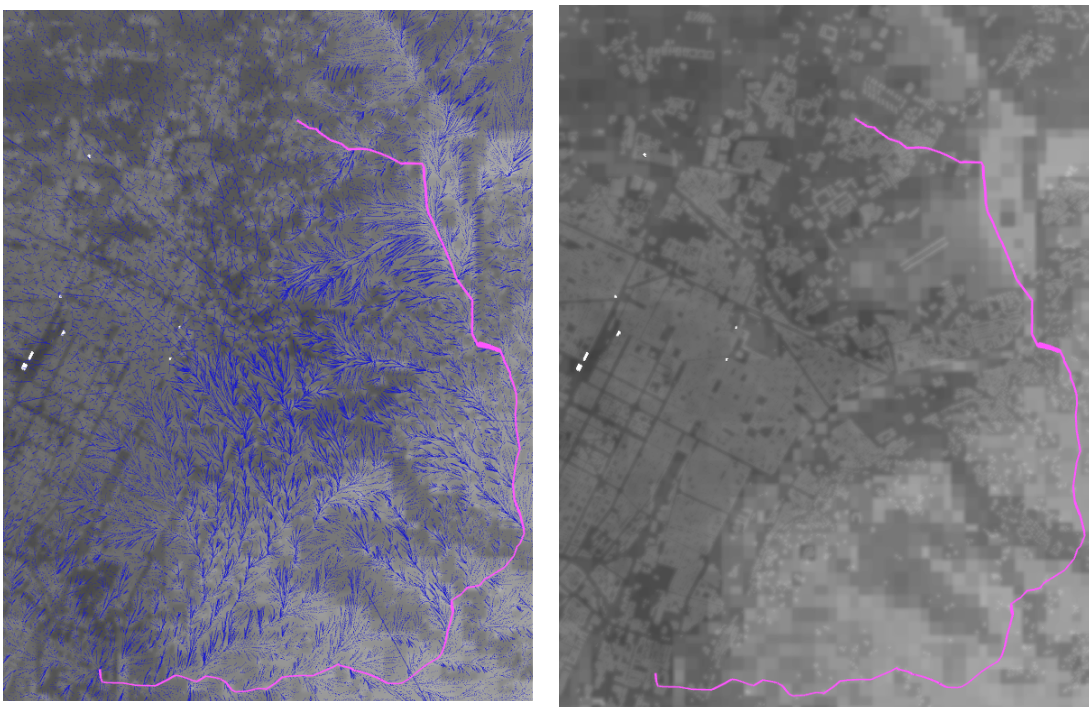
$$average\_risk = \frac{risk\_cost * velocity}{path\_length} = \frac{risk\_cost}{flight\_time}$$

Furthermore, in all the simulations, the total risk cost has been computed with the function *RiskCostOptimizationObjective*, that analyzes the risk map and creates a graph in the regions with low risk cost, whereas the total flight time has been reckoned dividing the total distance of the optimal path found by the constant velocity of the specific drone.

5.2.1 Simulations on the risk map of Mavic 2

Minimization of the risk cost

On the risk map of Mavic 2, the first simulation has been done considering the algorithm that minimizes the risk cost. In Figure 5.32 it has been reported the computed optimal path from the departure to the arrival Hospital. Moreover, in Figure 5.32a it is visualized how the graph is organized to reach the ending point minimizing the risk cost of the path, and in Figure 5.32b it is shown only the optimal path found, in order to better understand which area the drone can overfly.



(a) Graph of the optimal solution. **(b)** Optimal path.

Figure 5.32: Optimal path found for the minimization of the risk cost in the Mavic 2 map.

The simulation, for 50000 states, has found the optimal solution in 74.869 s. The optimal path found has:

- total risk cost of $1.894 \cdot 10^{-4}$;

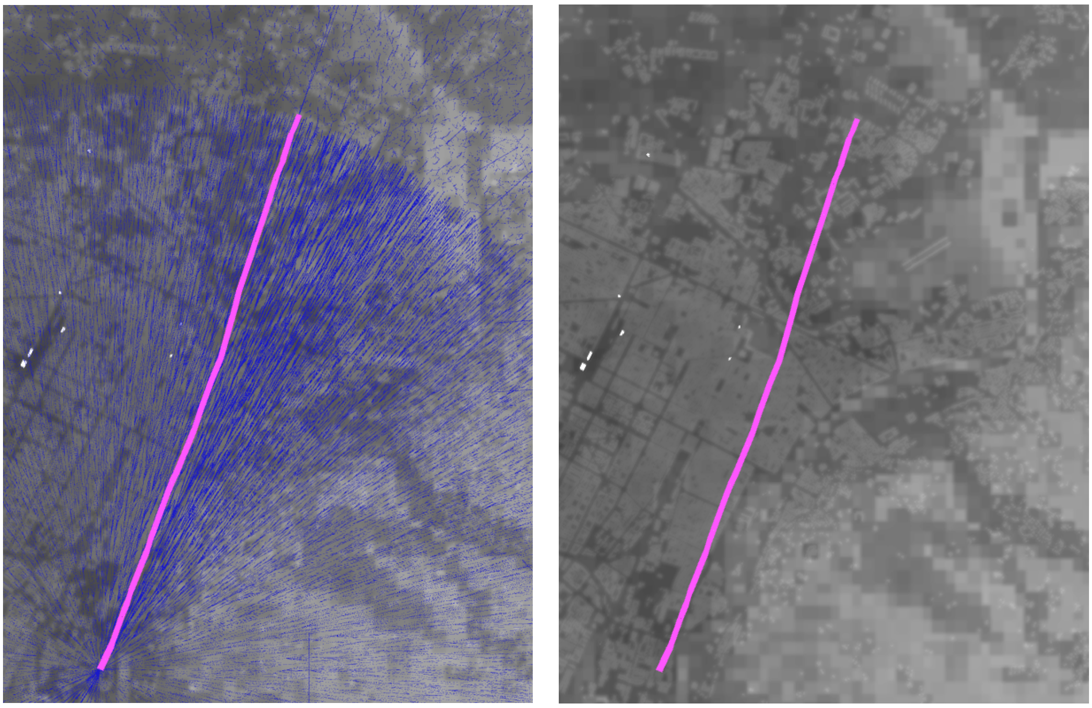
- total flight time of 308.786 s, which is consistent due to the total distance of the path and cruise speed of $20\,m/s$ of the Mavic 2;

- total distance of 6175.728 m;

- average risk cost of $6.134 \cdot 10^{-7}\ h^{-1}$, that is lower than the *Equivalent Level of Safety* (ELOS) threshold $(1 \cdot 10^{-6})$ that should be guaranteed to conduct a safe flight mission. As explained in a previous chapter, this threshold value determines the safety of the computed path..

Minimization of the flight time

As a second simulation on the Mavic 2 risk map, the algorithm that minimizes the flight time has been considered. As for the simulation on ideal maps, the function *FlightTimeOptimizationObjective* has been considered to achieve the goal of this optimizer. The total flight time has been computed as the ratio of the total distance of the path and the velocity of the drone, which is constant over the time. This means that the compute flight time corresponds to the actual flight time.



(a) Graph of the optimal solution. **(b)** Optimal path.

Figure 5.33: Optimal path found for the minimization of the flight time in the Mavic 2 map.

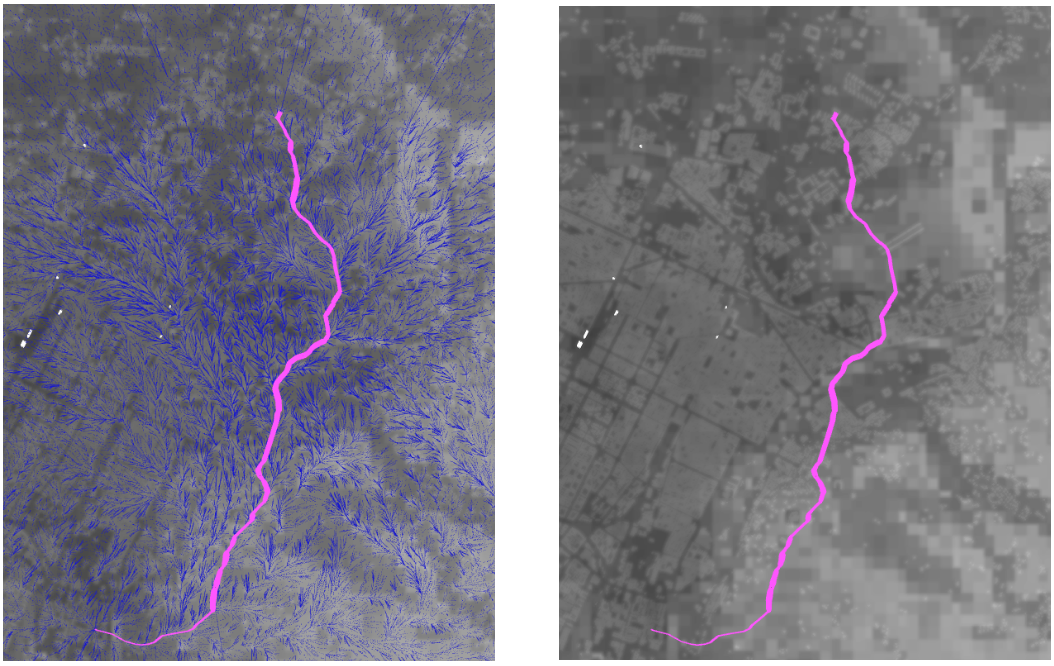
In Figure 5.33 it is shown the optimal path found by the algorithm in 7.796 s across 50000 states. In Figure 5.33a it is reported how the algorithm creates the graph, which is the same obtained for the simulation on the ideal map, so it is the one expected. Besides, in Figure 5.33b the only path is displayed to observe that the only flight time is minimized as the found path across area with high risk value. In this case, the optimal path has:

- total risk cost of $7.785 \cdot 10^{-4}$;
- total flight time of 166.040 s;
- total distance of 3320.808 m;
- average risk cost of $4.689 \cdot 10^{-6}\ h^{-1}$.

The total flight time is the expected one, knowing the total distance of the path and the cruise speed of the drone under study. Furthermore the average risk cost is higher than the ELOS threshold, so the optimal path found is not safe and cannot be crossed.

Minimization of the risk and constraint on the flight time

The third simulation regards the minimization of the risk cost while a threshold is imposed on the flight time.
As threshold has been chosen an average value between the flight time achieved in the minimization of the risk cost and the one obtained in the minimization of the flight time, i.e. 220 s is threshold value. The solution found in 99.171 s is reported in Figure 5.34, in the presence and absence of the graph, respectively in Figure 5.34a and 5.34b.



(a) Graph of the optimal solution. **(b)** Optimal path.

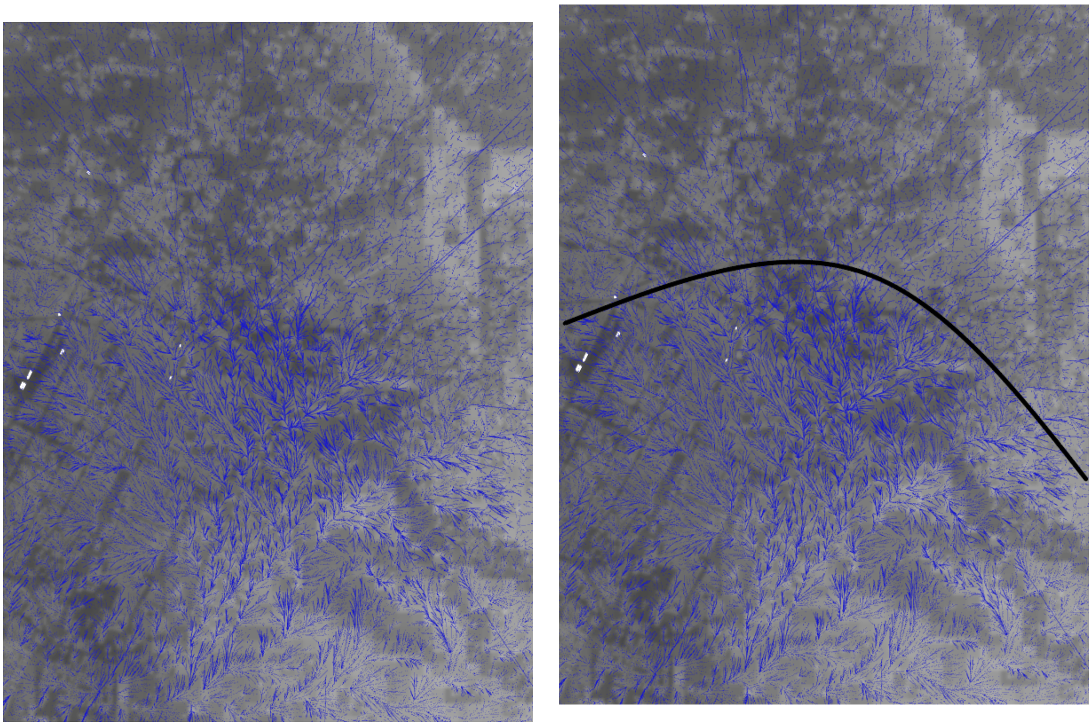
Figure 5.34: Optimal path found for the minimization of the risk cost while forcing a constraint on the flight time on the Mavic 2 map.

The optimal found path has:

- total risk cost of $3.109 \cdot 10^{-4}$;
- total flight time of 213.160 s, which is below the imposed threshold;
- total distance of 4263.200 m;
- average risk cost of $1.459 \cdot 10^{-6}$ h^{-1}, not a safe value since it is higher than the ELOS threshold.

From these obtained value the optimal path found minimize the total risk cost, while respects the flight time threshold. As it is shown on the map, the optimal path crosses an inner area of the map with respect to the one of the first simulation. Moreover, both on the Figure and the obtained values, it is evident that this optimizer is able to find an average behavior between the first two simulations. This aspect is highlight in the comparison done in the table 5.3.
The distance of the total path is an average value achieved with the previous simulations, further proof that the optimal found path is a mean one among the previous simulations.

Another test has been done with a lower value of threshold, i.e. 170 s. The chosen threshold value is too small since the optimization has returned an infinite solution, indication that the optimizer has not been able to reach the ending point minimizing the risk cost while respecting the constraint on the flight time. In the Figure 5.35 it is shown the graph that the algorithm has created until the threshold was satisfied, when it was no more able to fulfill it the graph has shown a messy behavior. In the Figure 5.35b it is highlighted, by means of a black line, the moment after which the threshold on the flight time is not satisfied and the graph is no more organized.



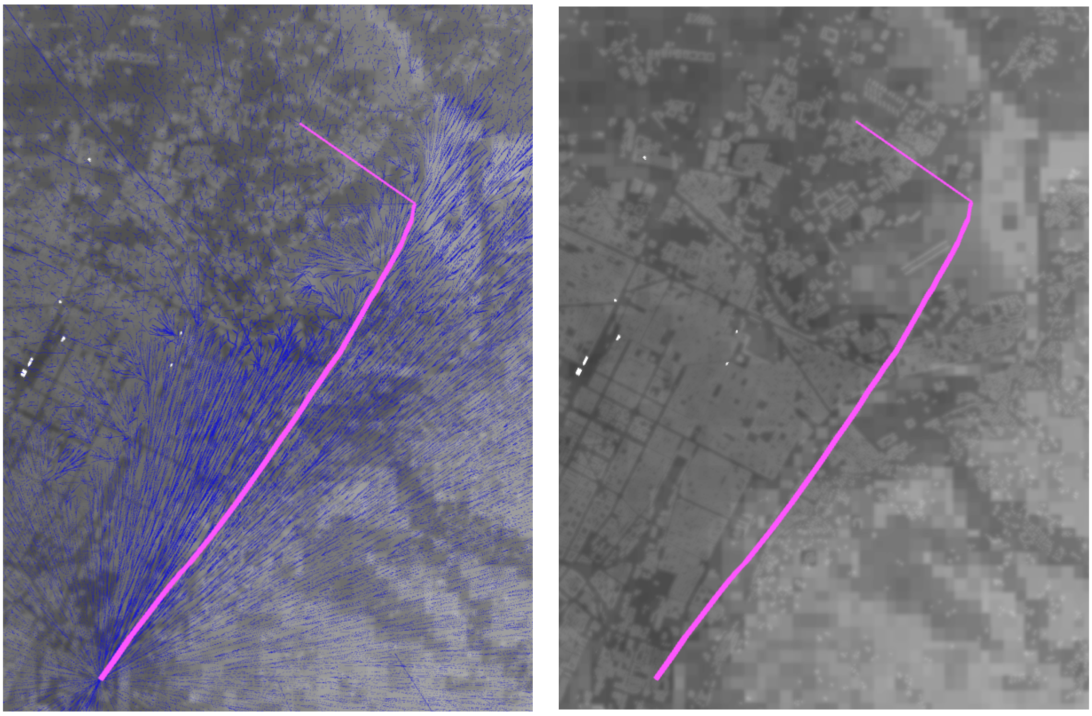
(a) Graph of the infinite solution.

(b) Highlighted threshold after which the graph is no more organized.

Figure 5.35: Trend of the graph that returns an infinite solution for the minimization of the risk cost while forcing a constraint on the flight time on the Mavic 2 map.

Minimization of the flight time and constraint on the risk

The fourth simulation minimizes the total flight time, by the total distance of the path and the constant velocity of the drone, while forcing a constraint on the risk cost, by means of the function *RiskCostOptimizationObjective*.
Likewise the threshold on the flight time, the constraint on the risk cost has been chosen considering the data obtained from the first two simulations and evaluating an average value, thus the taken risk cost threshold is $4.0 \cdot 10^{-4}$.
The optimal path has been found in 17.765 s and it is shown in Figure 5.36, where in Figure 5.34a it is printed the graph computed by the algorithm, while in Figure 5.34b, only the path is shown to better understand the areas crossed by the optimal path.



(a) Graph of the optimal solution. **(b)** Optimal path.

Figure 5.36: Optimal path found for the minimization of the flight time while forcing a constraint on the risk cost on the Mavic 2 map.

From the Figure 5.36a it is possible to visualize how the graph is organized in the clearest area. This lead to the understanding of how the algorithm has tried to minimize the flight time and to respect the risk cost constraint on the same time. Indeed in the darker regions the graph is not organized because the threshold is reached immediately. Moreover, near the ending point, the graph seems to be not ordered. This is due to the impossibility to satisfy the constraint, so the algorithm has considered a trade off between the minimum flight time and the minimum risk and has reached anyway the goal. Thus, the obtained results are:

- total risk cost of $4.569 \cdot 10^{-4}$, slightly higher than the threshold, but still reasonable considering the previous simulations;
- total flight time of 200.323 s, good result considering the values of the previous simulations;
- total distance of 4006.466 m;
- average risk cost of $2.281 \cdot 10^{-6}$ h^{-1}, above the ELOS threshold, thus it is not a safe value.

Evaluation of the results

The data obtained from all the simulations are summarized in the table 5.3.

	Time to find the optimal solution [s]	**Total risk cost**	**Total flight time** [s]	**Total distance** [m]	**Average risk cost** $[h^{-1}]$
Minimiz. risk cost	74.869	$1.894 \cdot 10^{-4}$	308.786	6175.728	$6.134 \cdot 10^{-7}$
Minimiz. flight time	7.796	$7.785 \cdot 10^{-4}$	166.040	3320.808	$4.689 \cdot 10^{-6}$
Minimiz. risk cost threshold flight time	99.171	$3.109 \cdot 10^{-4}$	213.160	4263.200	$1.459 \cdot 10^{-6}$
Minimiz. flight time threshold risk cost	17.765	$4.569 \cdot 10^{-4}$	200.323	4006.466	$2.281 \cdot 10^{-6}$

Table 5.3: Results of the simulations on the Mavic 2 risk map.

Evaluating the time needed to find the optimal solution, considering 5000 states, the shortest time is reached in the minimization of the flight time. This happens since the minimization is computed by means of the formula, so it is faster than the minimization of the risk cost, where it is necessary to analyze the overall risk by reading risk costs of the risk map.

The smallest risk cost, has predicted, is achieved in the minimization of the risk cost, while the highest values is obtained in the minimization of the flight time. An average value is got in the minimization on the risk cost with the threshold on the flight time. This was expected since was also considered an intermediate value for the total flight time. However, it has to be highlighted that the obtained average value are independent from the chosen value of threshold. Regarding the value of the flight time, the highest value is achieved in the first minimization while the lowest in the second minimization.

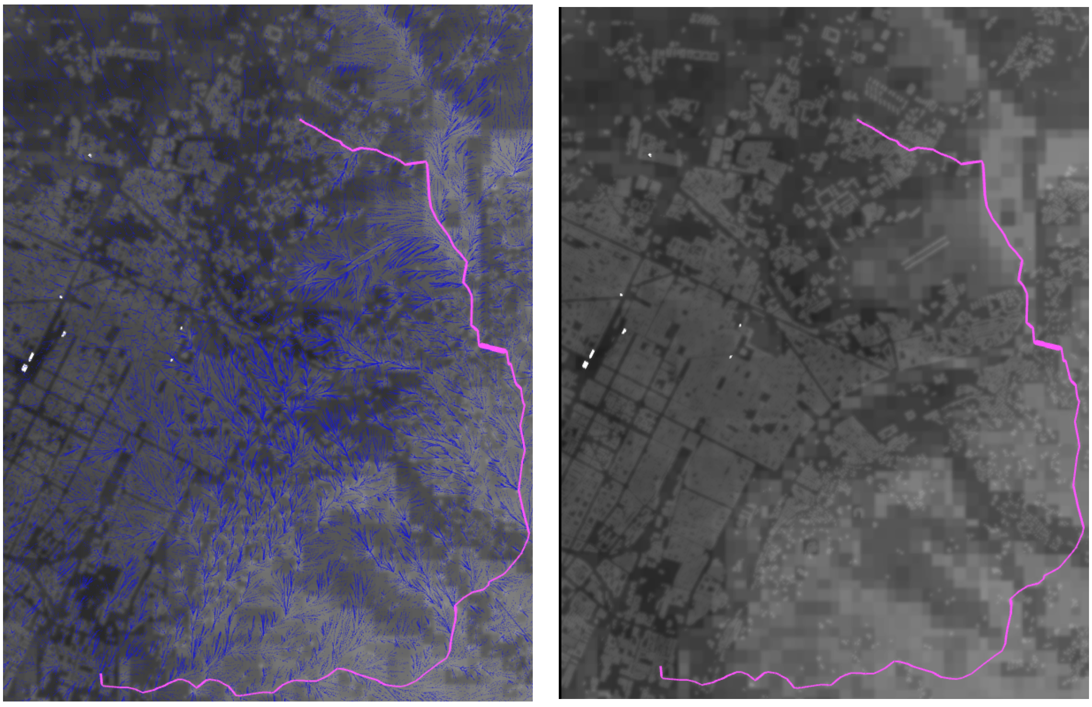
The total distance is more or less the same for all the simulation, an higher value it is expected when minimizing the risk cost, whereas a lower value in the minimization of the flight time, since start and goal are connected through an almost straight line.

The average risk cost of the path is under the ELOS threshold only in the first simulation, thus this implemented optimizer is safe and it can be applied with the drone Mavic 2. On the other hand, this drone cannot be associated with the other optimizers for the requirements imposed by the National Aviation Authorities.

5.2.2 Simulations on the risk map of Matrice 200

Minimization of the risk cost

The first simulation has been done minimizing the total risk cost of the risk map Matrice 200. From what observed in the simulation of the Mavic 2, the total risk cost should be the smallest of all the optimizations while the flight time the highest. In Figure 5.37 it is shown that the path minimizing the total risk follows the same trajectory of the one found for the Mavic 2. Comparing Figure 5.37a and Figure 5.37b the graph seems organized in the lightest areas.



(a) Graph of the optimal solution. **(b)** Optimal path.

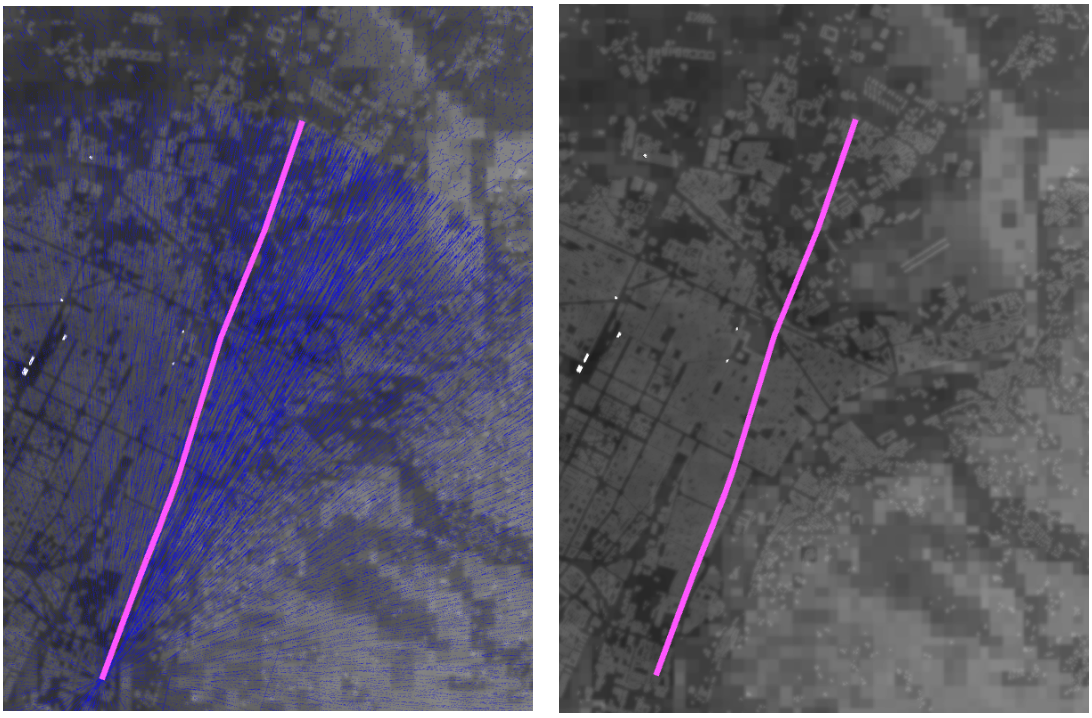
Figure 5.37: Optimal path found for the minimization of the risk cost in the Matrice 200 map.

The solution has been found in 89.301 s which has:

- total risk cost of $1.272 \cdot 10^{-3}$;
- total flight time of 368.406 s;
- total distance of 6262.896 m, which divided by the constant velocity of 17 m/s returns the total flight time obtained;
- average risk cost of $3.454 \cdot 10^{-6}$ h^{-1}, higher than the ELOS threshold, thus for the aviation regulation it cannot overfly an urban environment.

Minimization of the flight time

The solution found for the minimization of the total flight time is shown in Figure 5.38. The optimal path found has the wanted behavior since connects the starting and ending hospital with an almost straight line. Moreover, in Figure 5.38a the circle created by Euclidean distance is well noted.



(a) Graph of the optimal solution. **(b)** Optimal path.

Figure 5.38: Optimal path found for the minimization of the flight time in the Matrice 200 map.

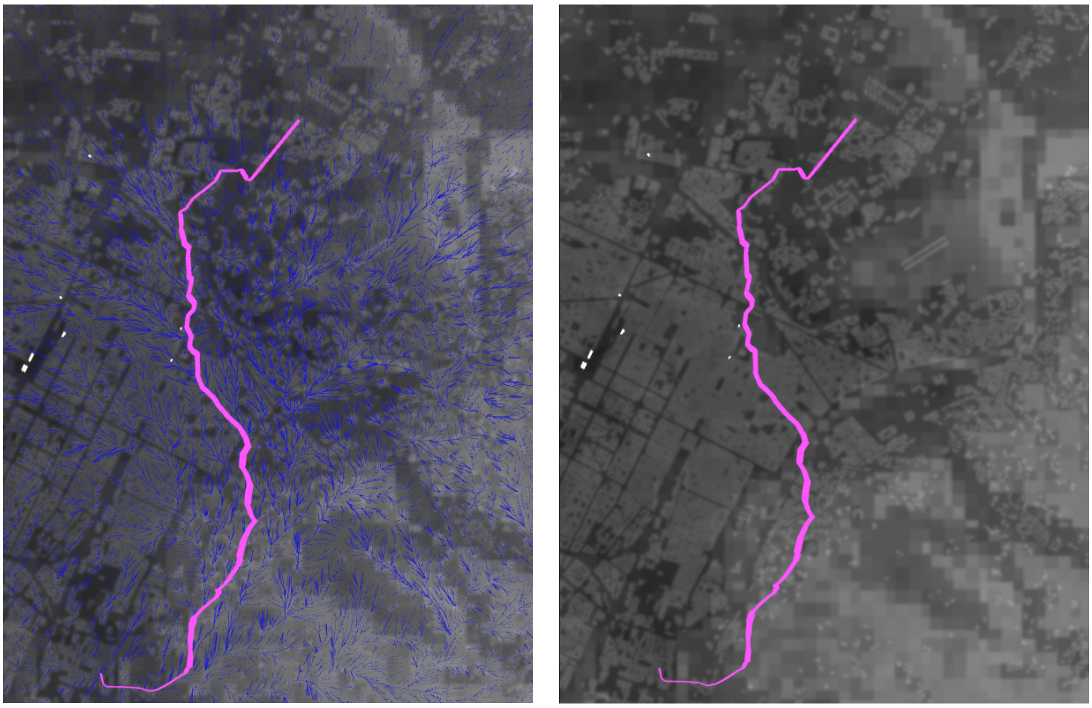
The total flight time of the computed path is 195.215 s that has been achieved in 8.111 s. The total risk cost is $4.835 \cdot 10^{-3}$ which is quite high, moreover the average risk ($2.477 \cdot 10^{-5}$ h^{-1}) cost provide that the obtained path is not safe, since the ELOS threshold is exceed. This means that, for the requirements imposed by the National Aviation Authorities, this drone is not suitable to execute this path. Summing up, the obtained results are:

- total risk cost of $4.835 \cdot 10^{-3}$;
- total flight time of 195.215 s;
- total distance of 3318.661 m;
- average risk cost of $2.477 \cdot 10^{-5}$ h^{-1}.

Minimization of the risk and constraint on the flight time

In this simulation a threshold of 280 s on the flight time has been imposed for the minimization of the total risk cost, that must not be overcomed. This threshold

value has been chosen as the average between the flight time achieved in the first simulation and the one in the second simulation.
In Figure 5.39 the optimal path follows an inner area of the map that has a higher risk cost than the outside zone since it is aimed at respecting the constraint on the flight time.



(a) Graph of the optimal solution. **(b)** Optimal path.

Figure 5.39: Optimal path found for the minimization of the risk cost while forcing a constraint on the flight time on the Matrice 200 map.

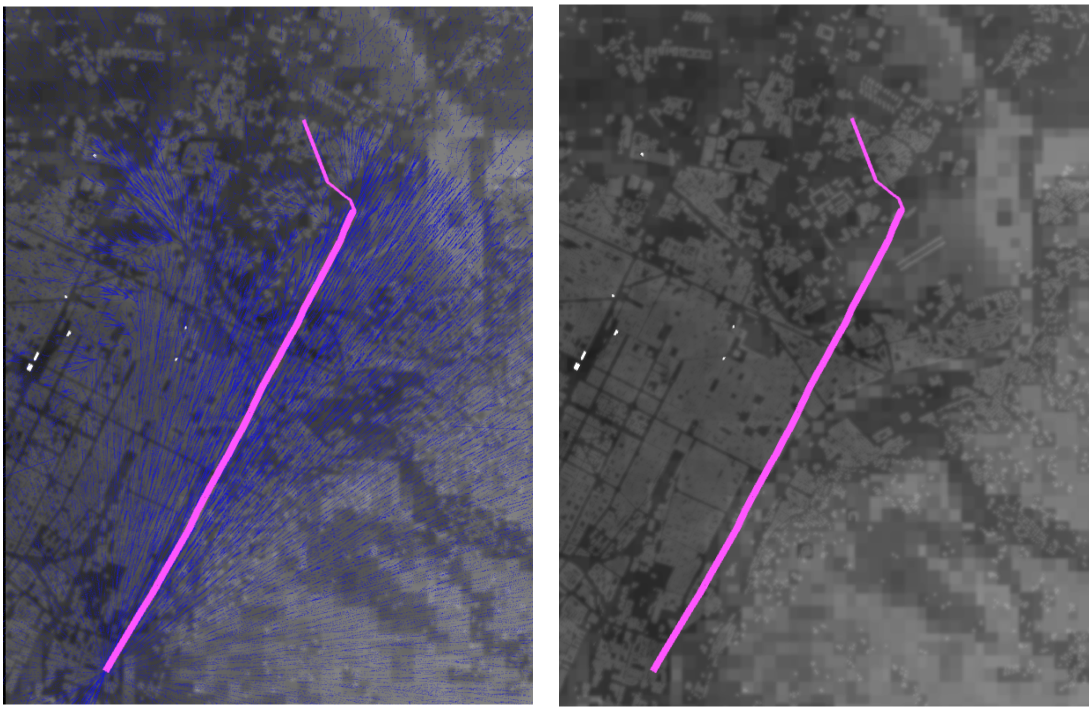
Indeed the results of the simulation indicate that:

- total risk cost of $2.924 \cdot 10^{-3}$;
- total flight time of 262.349 s, lower than the threshold;
- total distance of 4459.924 m;
- average risk cost of $1.115 \cdot 10^{-5}\ h^{-1}$, overcome the ELOS threshold and, consequentially, also this optimizer is not applicable for this type of drone.

However, since the solution has been found in 120.115 s, that is the imposed solve time, a better solution could be achieved enlarging it, even if the average risk cost is too high to expect a value that will respect the ELOS threshold.

Minimization of the flight time and constraint on the risk

The simulation aimed at minimizing the flight time considers now a threshold of $3.5 \cdot 10^{-3}$ for the risk cost. Likewise the threshold on the flight time, it has been obtained as the middle value between the first two simulations on the Matrice 200 map.
In Figure 5.40 the optimal path is a straight line until the threshold is satisfied, while the algorithm tries to minimize the flight time and also not exceed to much the threshold value in the ending part. As result a deviation on the path is performed to reach the goal, even if the graph seems no more organized in the latest part of the path.



(a) Graph of the optimal solution. **(b)** Optimal path.

Figure 5.40: Optimal path found for the minimization of the flight time while forcing a constraint on the risk cost on the Matrice 200 map.

The solution has been found in 24.820 s with:

- total risk cost of $3.782 \cdot 10^{-3}$, that slightly overcomes the threshold;
- total flight time of 210.042 s;
- total distance of 3570.719 m;

- average risk cost of $1.801 \cdot 10^{-5}\ h^{-1}$, still the ELOS threshold is exceeded and even this optimizer cannot be used for the Matrice 200.

Evaluation of the results

The data obtained from all the simulations are summarized in the table 5.4.

	Time to find the optimal solution [s]	**Total risk cost**	**Total flight time** [s]	**Total distance** [m]	**Average risk cost** [h^{-1}]
Minimiz. risk cost	89.301	$1.272 \cdot 10^{-3}$	368.406	6262.896	$3.454 \cdot 10^{-6}$
Minimiz. flight time	8.111	$4.835 \cdot 10^{-3}$	195.215	3318.661	$2.477 \cdot 10^{-5}$
Minimiz. risk cost threshold flight time	120.115	$2.924 \cdot 10^{-3}$	262.349	4459.924	$1.115 \cdot 10^{-5}$
Minimiz. flight time threshold risk cost	24.820	$3.782 \cdot 10^{-3}$	210.042	3570.719	$1.801 \cdot 10^{-5}$

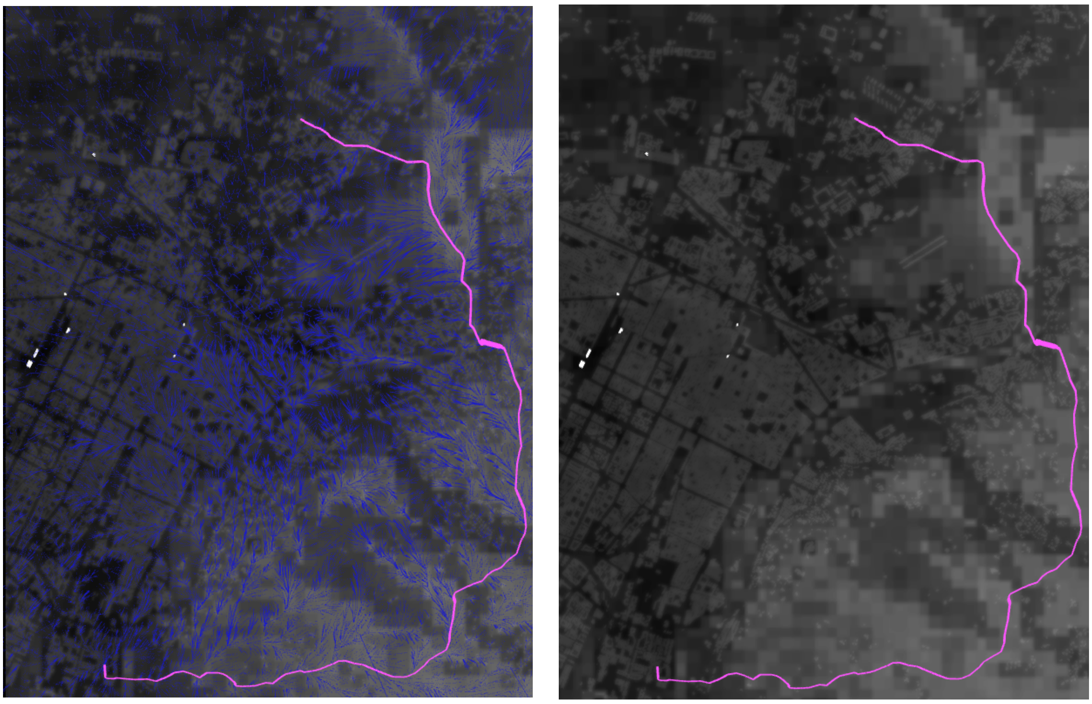
Table 5.4: Results of the simulations on the Matrice 200 risk map.

Comparing these values it is noticeable that the fastest optimizer to find the optimal solution is the one that minimize the flight time, while the slowest is the one that has to minimize the total risk cost and also consider the constraint on the flight time. For what regards the total risk cost the better solution is achieved by the optimizers that have to minimize it, while the better total flight time is reached by those who have to minimize it. As consequence the total distance is higher according to the value of the flight time. In fact, must be reminded that the flight time is computed from the distance. By looking at the last column of the table 5.4, no optimizer has an average value that satisfy the ELOS constraint. Accordingly to the National Aviation Authorities, this drone is not suitable for these missions, unless the risk is reduced by means of a more reliable drone.

5.2.3 Simulations on the risk map of Matrice 600

Minimization of the risk cost

The simulations have also be done considering the Matrice 600 risk map. Since this map is the darkest one, thus the most dangerous in terms of risk cost, it is expected a worst behavior than the simulations with the other two maps.
The first simulation aim at minimizing the total risk cost of the risk map Matrice 200. Likewise the simulation for the Mavic 2 and the Matrice 200, the optimal path is the one that avoids the city area, as illustrated in Figure 5.41.



(a) Graph of the optimal solution. **(b)** Optimal path.

Figure 5.41: Optimal path found for the minimization of the risk cost in the Matrice 600 map.

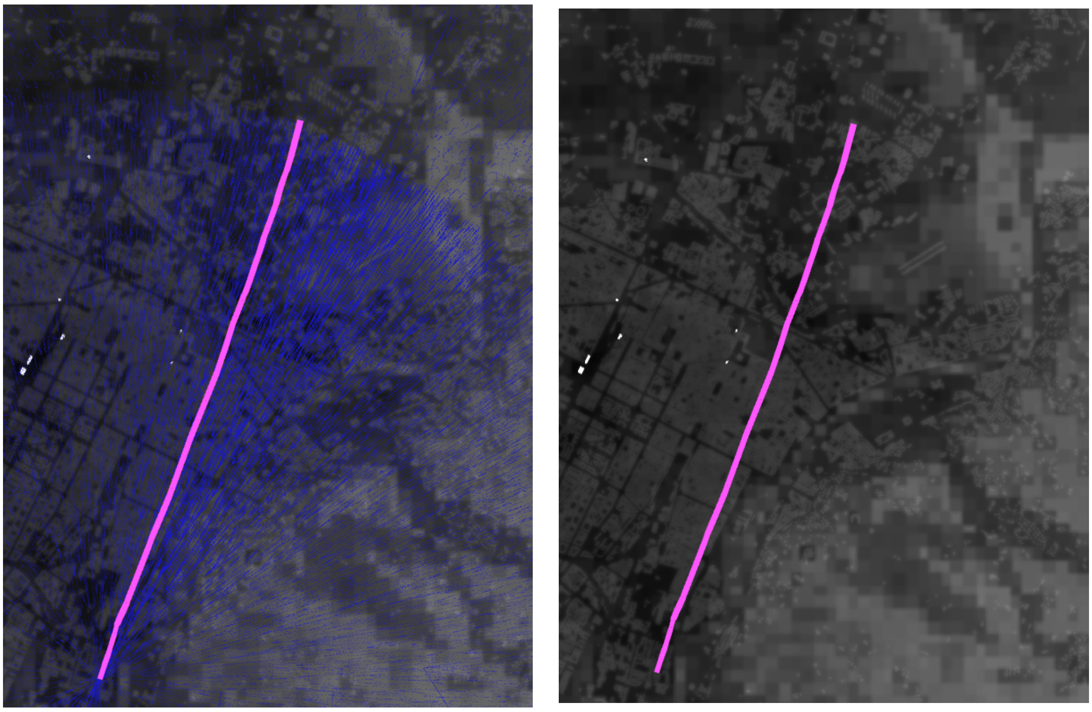
The solution has been found in 95.473 s and the total risk cost of the path is $5.289 \cdot 10^{-3}$. It is an high value of risk, especially if the average risk cost is $(1.531 \cdot 10^{-5}\ h^{-1})$ higher than the ELOS value. Therefore this optimizer cannot be used for the Matrice 600 and, since this optimizes the total risk cost of the path, it is expected that neither the other optimizers should be safe for a drone so heave. The results of this simulation are:

- total risk cost of $5.289 \cdot 10^{-3}$;

- total flight time of 345.488 s;
- total distance of 6218.776 m;
- average risk cost of $1.531 \cdot 10^{-5}$ h^{-1}.

Minimization of the flight time

The minimization of the flight time leads to a straight path that connects the starting and the ending Hospital, like the simulation for the Mavic 2 and the Matrice 200. This conduct is shown is Figure 5.42 and due to the high risk cost of the map, higher values, than the ones of the other drones, are expected.



(a) Graph of the optimal solution. **(b)** Optimal path.

Figure 5.42: Optimal path found for the minimization of the flight time in the Matrice 600 map.

The optimal path found has:

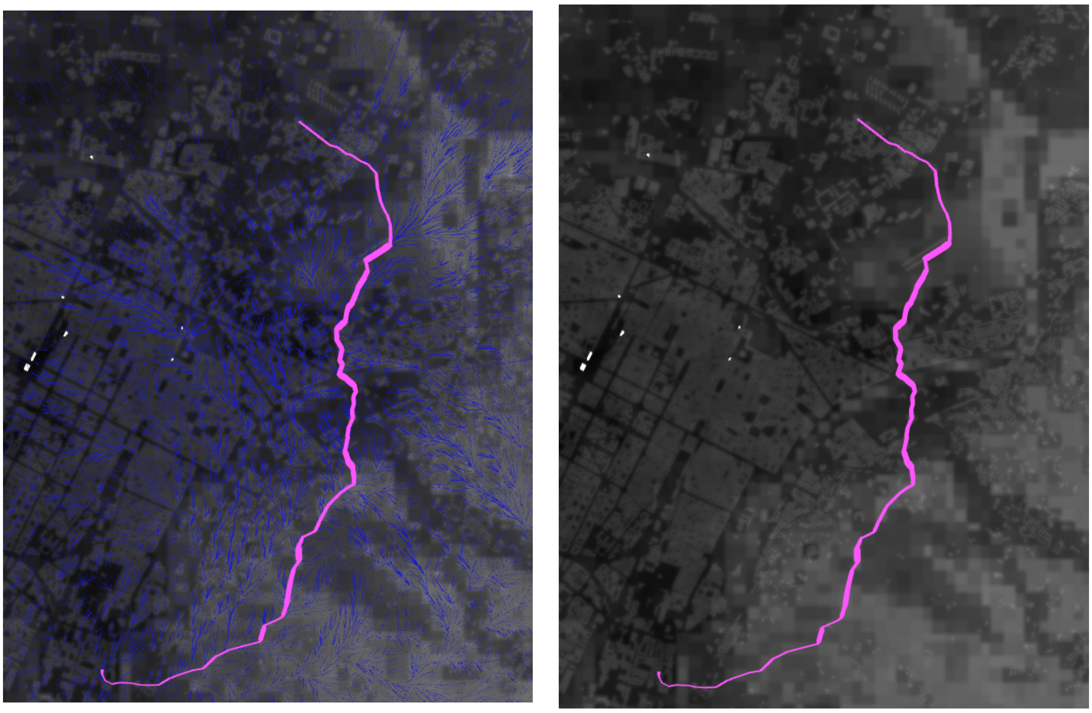
- total risk cost of $1.962 \cdot 10^{-2}$;
- total flight time of 184.458 s;
- total distance of 3320.237 m;

- average risk cost of $1.064 \cdot 10^{-4}\ h^{-1}$ does not respect the ELOS constraint, so the optimal path is not secure and this optimizer cannot be used for Matrice 600.

These results have been found in 7.761 s, almost the same computation time of the Mavic 2 and Matrice 200.

Minimization of the risk and constraint on the flight time

In Figure 5.43 is shown the path that tries to minimize the risk cost, while satisfying the threshold of 280 s on the flight time.



(a) Graph of the optimal solution.

(b) Optimal path.

Figure 5.43: Optimal path found for the minimization of the risk cost while forcing a constraint on the flight time on the Matrice 600 map.

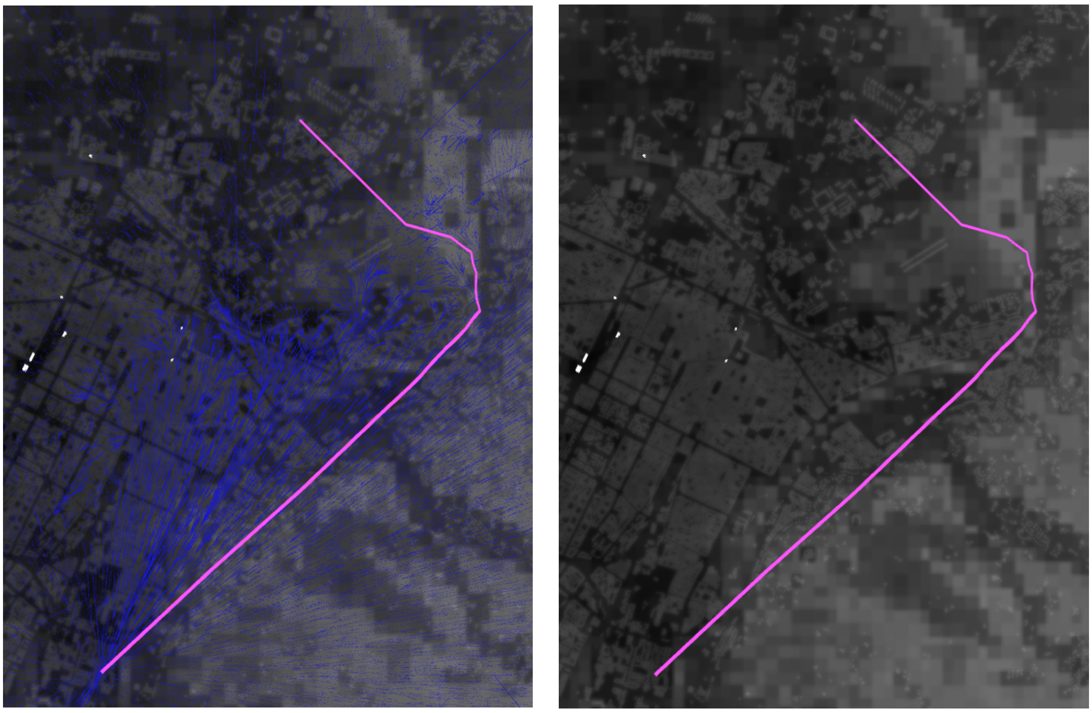
The optimal path has respected the imposed threshold, since:

- total risk cost of $9.056 \cdot 10^{-3}$;
- total flight time of 259.681 s;
- total distance of 4674.252 m;

- average risk cost of $3.487 \cdot 10^{-5}$ h^{-1} overcomes the ELOS threshold, determining that also this optimizer is not applicable to the Matrice 600.

Minimization of the flight time and constraint on the risk

The imposed risk cost threshold is $1.0 \cdot 10^{-2}$, the average value of the total risk cost obtained from the first two simulations. The Figure 5.44 highlights that the algorithm reaches the goal even if it does not succeed in respecting the threshold near it. So, in Figure 5.44a, the graph is organized until the threshold is satisfied, and then only the quickest way is searched.



(a) Graph of the optimal solution.   **(b)** Optimal path.

Figure 5.44: Optimal path found for the minimization of the flight time while forcing a constraint on the risk cost on the Matrice 600 map.

The best solution is found after 17.658 s, with:

- total risk cost of $1.269 \cdot 10^{-2}$ does not exceed too much the imposed threshold;
- total flight time of 252.707 s;
- total distance of 4548.724 m;

- average risk cost of $5.020 \cdot 10^{-5}$ h^{-1} does not fit the ELOS threshold, therefore also this optimizer does not produce a safe pah for an heavy drone.

Evaluation of the results

The results of all the simulations regarding the Matrice 600 are reported in table 5.5. They are the expected one and similar to the ones of the Mavic 2 and Matrice 200.

	Time to find the optimal solution $[s]$	**Total risk cost**	**Total flight time** $[s]$	**Total distance** $[m]$	**Average risk cost** $[h^{-1}]$
Minimiz. risk cost	95.473	$5.289 \cdot 10^{-3}$	345.488	6218.776	$1.531 \cdot 10^{-5}$
Minimiz. flight time	7.761	$1.962 \cdot 10^{-2}$	184.458	3320.237	$1.064 \cdot 10^{-4}$
Minimiz. risk cost threshold flight time	120.051	$9.056 \cdot 10^{-3}$	259.680	4674.252	$3.487 \cdot 10^{-5}$
Minimiz. flight time threshold risk cost	17.658	$1.268 \cdot 10^{-2}$	252.707	4548.724	$5.020 \cdot 10^{-5}$

Table 5.5: Results of the simulations on the Matrice 600 risk map.

Even with this risk map the total risk cost is smaller in the minimization of the risk cost while higher in the minimization of the flight time, the reverse is verified for the minimization of the flight time, which results to be higher in the minimization of the risk cost wheres smaller in the minimization of the risk cost. The trade-off is achieved for the minimization regarding also the threshold.
The time needed to find the optimal solution has the same trend of the simulations on Matrice 200 risk map. Indeed the solve time seems small for the minimization of the risk cost considering the threshold on the flight time, due the optimal solution found at the end of the execution of the algorithm. Thus a better behavior could be obtained if an higher value of solve time will be considered.

By focusing on the average risk cost, no optimizations pass the ELOS condition to be lower than $1.0 \cdot 10^{-6}$, thus the Matrice 600 cannot be considered for the life-saving pharmaceuticals delivery. However, the use of mitigations (e.g. a parachute) to improve the reliability of the drone, may reduce the effective risk.

Chapter 6

Conclusions

In this master's thesis, the study of different strategies of path planning for the transportation of life-saving pharmaceuticals using unmanned aerial vehicle in urban environments have been focused. The study started with the implementation, in C++, of four different optimizers, that aim at: (i) minimization of the total risk cost of the path; (ii) minimization of the total flight time; (iii) minimization of the total risk cost, while evaluating a constraint on the flight time; (iv) minimization of the total flight time, whilst considering a threshold on the risk cost.

The algorithm with which these optimizers are associated is the well-known Optimal Rapidly-exploring Random Tree (*RRT**), a sample-based algorithm, that rapidly explore the state space with an incremental tree and returns a near optimal path. The optimum path is quantified by the risk-map, which defines the risk level of each area of a map and identifies the zones where the drone is not allowed to fly due to presence of obstacles at flight altitude or no-fly zones forced by the aviation rules.

The path planning algorithm is developed using the the Open Motion Planning Library OMPL, whose functions provide the components for solving a path planning query. Among all the OMPL classes, the modified one has been the *OptimizationObjective*, in which the function *motionCost* has been appropriately adapted to the goal of each optimizer.

Once the optimizers have been implemented, they have been tested on ideal map, to analyze the correct behavior. As result, all the optimizers return an optimal path until the constraint (if present) is respected; once it is no more fulfilled, the algorithm restores a path that has infinite cost, i.e. it is no optimal.

Lastly, three types of drone have been chosen to be aware of all the possible scenario in which the algorithm can be applied. For this reason the drones have

three significantly different mass as the payload maximum that they can carry. For each of them a risk map has been computed, whose risk level is diverse among them.
The four different optimizers have been simulated on the risk map representing the city of Turin.
It has been demonstrated that the best outcome in terms of risk cost is achieved by the optimizer that minimizes the risk cost but it produces the worst flight time; on the other hand the smallest flight time is obtained by the optimizer that minimize the only fly time, which returns the worst risk cost of the path. The trade-off behavior is procured by considering a threshold on the risk cost or the flight time.
Concerning the drone adopted, it has been shown that all the optimizers produce the best result on the risk map of the lightest drone, since it has a low risk cost map. On the contrary the worst path, for all the optimizers, is achieved by the risk map of the heaviest drone, due to its high risk cost map.
Moreover, for the drone having an average weight, it has been got that no optimizer computes a path that has an average risk cost under the value imposed by the Equivalent Level Of Safety (ELOS). This means that the optimizers found an optimal path which is not safe considering the current regulations, from a risk point of view, thus they can be not applied to drone whose mass is around $4.7\,kg$.
Furthermore, the risk map associated to the drone owing a high mass, returns a path that is not safe for each implemented optimizer, since each average risk cost is higher than $1 \cdot 10^{-6}\ h^{-1}$. Therefore the heave drone, as it is, cannot be considered for medical delivery in an urban environment, unless the reliability of the drone is improved. In fact, the use of risk mitigations, such as the equipment of a parachute, reduces the effective risk making possible the use of heavier drones.

This work could be further improved by considering, in the minimization of the flight time while forcing a constraint on the risk cost, a threshold on the average risk cost instead of the total risk cost. This would lead to the computation of a path who is certainly safe (using the ELOS as threshold) and thus all the optimizers could be used for every type of drone.

Bibliography

[1] «Urban logistics and mobility concepts». In: (). DOI: https://uam.fev.com/ (cit. on p. 1).

[2] «Urban logistics and mobility concepts». In: (). DOI: https://hypermotion-frankfurt.messefrankfurt.com/frankfurt/en/themes/urban-logistics-mobility-concepts.html (cit. on p. 1).

[3] «Storia ed evoluione dei droni: gli apparecchi SAPR dalla nascita al futuro.» In: (). DOI: https://www.egm96.it/storia-ed-evoluzione-dei-droni-gli-apparecchi-sapr-dalla-nascita-al-futuro/ (cit. on p. 1).

[4] «Urban Air Mobility (UAM)». In: (). DOI: https://www.easa.europa.eu/domains/urban-air-mobility-uam (cit. on p. 1).

[5] Cestino E. and Romeo G. «Innovative Unmanned Aircrafts: role and constraints for GMES applications». In: *3rd CEAS Air Space Conference* (2011) (cit. on p. 1).

[6] Antonio Amoroso et al. *INDOOR DRONI E TRAPIANTI. Visioni nel futuro del trasporto di materiale sanitario, con particolare attenzione alla medicina dei trapianti.* Brochure. Univ. degli Studi di Torino et al. (cit. on p. 2).

[7] Francesco Cavallo. «Innovative ways of distribution for life-saving medicines during emergency situations». MA thesis. Politecnico di Torinp, 2020 (cit. on pp. 2, 12, 56).

[8] Cornelius A. Thiels, Johnathon M. Aho, Scott P. Zietlow, and Donald H. Jenkins. «Use of Unmanned Aerial Vehicles for Medical Product Transport». In: *Journal of Air Medical Transport* (2015). URL: https://www.researchgate.net/publication/272946746 (cit. on p. 3).

[9] Dr Paul G Royall and Patrick Courtney. «Medicine delivery by drone – implications for safety and quality». In: (Nov. 2019). DOI: https://www.europeanpharmaceuticalreview.com/article/103799/medicine-delivery-drone-safety-and-quality/ (cit. on pp. 3, 10).

[10] Michelle Sing Yee Hii, Patrick Courtney, and Paul G. Royall. «An Evaluation of the Delivery of Medicines Using Drones». In: *Drones* (June 2019) (cit. on pp. 4, 11).

[11] Timothy Amukele, Paul M. Ness, Aaron A.R. Tobian, Joan Boyd, and Jeff Street. «Drone transportation of blood products». In: *Transfusion* (November 2016). DOI: `https://www.researchgate.net/publication/310474300_Drone_transportation_of_blood_products` (cit. on p. 5).

[12] Margaret Eichleay, Emily Evens, Kayla Stankevitz, and Caleb Parker. «Using the Unmanned Aerial Vehicle Delivery Decision Tool to Consider Transporting Medical Supplies via Drone». In: *Global Health: Science and Practice* (2019). DOI: `https://www.ghspjournal.org/content/7/4/500` (cit. on p. 5).

[13] Anna Konert, Jacek Smereka, and Lukasz Szarpak. «The Use of Drones in Emergency Medicine: Practical and Legal Aspects». In: *Hindawi: Emergency Medicine International* (2019). DOI: `https://www.hindawi.com/journals/emi/2019/3589792/` (cit. on p. 6).

[14] «Regolamento ENAC, Mezzi Aerei a Pilotaggio Remoto, Ed.3, Emendamento 1 del 14 luglio 2020». In: () (cit. on p. 6).

[15] *D-Flight - Enabling Autonomous Flight.* D-Flight. URL: `https://www.d-flight.it/portal/` (cit. on pp. 7, 9).

[16] «Circolare ENAC ATM-09, Aeromobili a pilotaggio remoto - Criteri di utilizzo dello spazio aereo del 24 maggio 2019.» In: () (cit. on pp. 7, 8).

[17] Gaurav Singhal, Babankumar Bansod, and Lini Mathew. «Unmanned Aerial Vehicle classification, Applications and challenges: A Review». In: *Preprints* (Nov. 2018). DOI: `https://www.preprints.org/manuscript/201811.0601/v1` (cit. on p. 10).

[18] «Drone types: multi-rotor vs fixed-wing vs single rotor vs hybrid VTOL». In: *Australian DRONE magazine, issue 3* (June 2016). DOI: `auav.com.au/articles/drone-types/` (cit. on p. 11).

[19] Allahyar Montazeri, Aydin Can, and Imil Hamda Imran. «Unmanned aerial systems: autonomy, cognition, and control». In: *Unmanned Aerial Systems. Theoretical Foundation and Applications. Advances in Nonlinear Dynamics and Chaos (ANDC).* 2021, pp. 47–80. URL: `https://doi.org/10.1016/B978-0-12-820276-0.00003-0` (cit. on p. 14).

[20] Hui Liu. «Introduction». In: *Robot Systems for Rail Transit Applications.* School of Traffic and Transportation Engineering, Central South University, Changsha, China, 2020, pp. 1–36 (cit. on p. 15).

[21] Sertac Karaman and Emilio Frazzoli. «Sampling-based Algorithms for Optimal Motion Planning». In: (May 2011) (cit. on pp. 15–21, 23).

[22] Xinting Hu, Bizhao Pang, Fuqing Dai, and Kin Huat Low. «Risk Assessment Model for UAV Cost-Effective Path Planning in Urban Environments». In: *IEEE Access* (Aug. 2020) (cit. on p. 15).

[23] Kavraki Lab Rice University. «Open Motion Planning Library: A Primer». In: *RICE* (Jan. 2021). DOI: `https://ompl.kavrakilab.org/OMPL_Primer.pdf` (cit. on pp. 16, 18, 19, 24–26).

[24] Alessandro Rizzo, Giorgio Guglieri, Luca Spanò Cuomo, and Stefano Primatesta. «An Innovative Algorithm to Estimate Risk Optimum Path for Unmanned Aerial Vehicles in Urban Environments». In: *Transportation Research Procedia* (2018) (cit. on pp. 24, 26, 31–33).

[25] Stefano Primatesta, Alessandro Rizzo, and Anders la Cour-Harbo. «Ground Risk Map for Unmanned Aircraft in Urban Environments». In: *Journal of Intelligent Robotic Systems* (Nov. 2018), pp. 489–509. DOI: `https://doi.org/10.1007/s10846-019-01015-z` (cit. on pp. 27, 28).