POLITECNICO DI TORINO
Master degree course in Mechatronic Engineering

Master Degree Thesis

Cloud Based UASs Traffic Management: a Risk-Aware Map Manager

Supervisors
prof. Alessandro Rizzo
prof. Giorgio Guglieri

Candidate
Andrea Lorenzini

Internship Tutor
dott. ing. Stefano Primatesta

Anno accademico 2017-2018

Abstract

Every day the usage of Unmanned Aerial Systems (UASs), also for civil application, increases. The features that most characterize this new kind of flights are: low experience and capabilities of the pilot/owner, complex urban scenario in which the UAS moves, low quality in the manufacturing of the aircrafts and a lack of a common regulation, not only between different states, but also inside the same national airspace. The result of such situation is, beside a huge quantity of confusion, the impossibility of applying safety criteria and performing a coherent risk analysis of the drones' flight.
In this context, a new research project at TIM Joint Open Lab in Politecnico di Torino began to analyse the potentiality of cloud robotics, in order to build a completely new Traffic Manager capable to handle the complexity of this environment and finally ensuring safety for people on ground. The results, actually very encouraging, brought to the first implementation of a Cloud-Based UASs Traffic Manager (CBUTM) in a ROS environment. Many different aspects (and issues) came out while developing the whole system, and only a great team work could allow to reach a complete knowledge of the traffic manager from every point of view. Thanks to the capability of the cloud computing, it is possible to remove the men from the loop: drones are controlled by the CBUTM from the planning of the flight till the landing, without the need of human intervention in any of the phases.
In this thesis, the problem of risk assessment will be analyse in deep and the results will be used for the construction of a completely new Risk-Aware Map Manager.
The first step to accomplish in this sense is to have a complete, coherent and standardized risk metric. The state of the art will be so exposed, highlighting its pro and its lacks. Upon this, a new and complete discussion about risk will be performed, in order to provide all the instruments to calculate the risk, in a urban scenario but not only, as victims per hour of flight. Beside, an analysis of the actually known legislations will allow us to find an upper-bound for the risk (or victims' rate) that can be easily adopted by all the Flight National Authorities.
After the definition of the metric, it's possible to start the construction of a Map of the Risk (risk map) of the environment in which the UAS flies. In order to accomplish this step, not only a well-defined risk assessment procedure has to be defined, but also a set of techniques to correctly find, acquire and analyse a 3D model of the operative scenario (Environmental Modelling) are needed.
The capabilities of the cloud, as for example its great computational power and the real time update of information through Internet connection are crucial for this part of the process, since it can produce a dynamic risk map, function of space but also of time. This aspect of the work is really innovative, and has been merged also with some planar robotics' knowledge to define completely the rules and settings of the risk maps.
Moving a moment apart from the risk field, the Map Manager has been developed to include also other aspects of an Unmanned Mission: the study of this new kind of flight has shown many different elements crucial to consider a mission successfully completed. To accomplish this need, a framework capable to consider each requirement as a "layer", and merge it into a final Cost Layer

(or cost map) according to each priority has been built.
Then, two last parts complete the work of the Map Manager. First, the Path Planner, which works upon the Cost Layer to find the best trajectory for the drone to perform, and then the Path Validator, which takes as input the trajectory and gives as output the authorization (or not, if safety standard are not satisfied) to fly. Once again, a validation metric has been developed from scratch.
What we are going to present in this thesis is just one of the aspect of the CBUTM, focused on mapping the environment and on risk assessment, while the rest of the Traffic Management will be treated by other thesis works. Finally, the Risk-Aware Map Manager is a modular element of the traffic management, and can be easily extended in future in order to make it capable of accomplish new tasks (or make it better), if needed.

Acknowledgements

I want to thank all those who shared with me all, a good part, or just a piece of the road that has brought me here. You are all equally important, because what we are in this moment is child of the social, working, affective relationships, which we entertain throughout this long journey.
We are continuous exchanges, in life as in the thesis proceeding alone is only to put a stop to our potential.
For this reason I want to thank sincerely my companion of life, it doesn't matter how close, not only for help and support but above all for allowing me to become what I am now.

Contents

List of Tables

List of Figures

Acronyms

UAV Unmanned Aerial Vehicle

UAS Unmanned Aerial System

GSM Global System for Mobile communication

GPS Global Positioning System

NCS Networked Control System

UTM UAVs Traffic Management

CBUTM Cloud-Based UASs Traffic Management

CCS Cloud Control Station

PTP Predictive Trajectory Planner

RAMM Risk-Aware Map Manager

IMU Inertial Measurements Unit

SITL Software In The Loop

Chapter 1

Introduction

1.1 Context Definition

In the last ten years the usage of Unmanned Aerial Vehicles (also known as Drones) enormously increased: from civil application to the most difficult military operation, it seems that UAVs can be useful in almost every fields and situations. According to [1] this growth is not going to stop, especially for what concerns the civil environment: by the end of 2021 commercial drones shipments will reach 805,000, with a compound annual growth rate (CAGR) equal to 50%.
In parallel with this phenomena, a lack in a coherent legislation development has brought to a situation in which every single state has a different policy on drones flight, very often not appropriate since they are commonly too restrictive or too relaxed.
In order to better understand the environment in which this work is going to move, some central concepts have to be presented, here and in the rest of this chapter.
First of all, it's important to define formally the concept of Unmanned System. According to [2]:

Definition 1. An Unmanned System (US) is an electro-mechanical system, with no human operator aboard, that is able to exert its power to perform designed missions. It may be mobile or stationary. Includes categories of unmanned ground vehicles (UGV), unmanned aerial vehicles (UAV), unmanned underwater vehicles (UUV), unmanned surface vehicles (USV), unattended munitions (UM), and unattended ground sensors (UGS).

In particular, this thesis work will focus on Unmanned Aerial Systems (UASs).
The crucial point that characterizes each of this items is the non-presence of personal on board. This is the key features of USs, since without the need of human operator on board they can be smaller, lighter but most of all they can perform very dangerous tasks without any risks for pilots. Furthermore, it's interesting to notice that UAVs are very often used in boring or repetitive missions, in order to not stress human personal.
But, if no human is needed on board to guide it, how an unmanned system can accomplish a specific task? Let's take a look to one of the most common classification for UAVs, based on their level of autonomy (LOA), which influences the mode of operation of a mission.
The mode of operation of the Unmanned Systems can be [2]:

Remote Control A mode of operation wherein the human operator, without benefit of video or other sensory feedback, directly controls the actuators of the US on a continuous basis, from off the vehicle and via a tethered or radio linked control device using visual line-of-sight.

Teleoperator A mode of operation wherein the human operator, using video feedback and/or other sensory feedback, either directly controls the actuators or assigns incremental goals,

waypoints in mobility situations, on a continuous basis, from off the vehicle and via a tethered or radio linked control device.

Semi-Autonomous A mode of operation where the human operator and/or the US itself plans and conducts a mission.

Fully Autonomous A mode of operation wherein the US is expected to accomplish its mission, within a defined scope, without human intervention. Note that a team of USs may be fully autonomous while the individual team members may not be due to the needs to coordinate during the execution of team missions.

Unmanned Aerial Vehicles (UAV) and more in general all the Unmanned Systems (US) are one of the more interesting technologies developed in the last years. The reason behind the growth in their usage becomes evident if we look to the typical scenarios where this systems are involved. An unmanned mission is commonly defined as:

Definition 2. An unmanned mission consists in a trajectory travelled by an Unmanned Systems, that has both a starting and an ending geographical point and a particular task to perform.

Moreover, the state of the art of technologies allows to suppose that in future missions will have to be completed in a fully autonomous mode, that implies no human in the loop neither driving nor supervising. This condition obviously introduces a lot of issues, since the same safety level as the one with a human operator must be guaranteed.
The idea of unmanned mission is not new, since the first attempt to build such kind of vehicle was performed by Austria to attack Venice with bomb-filled balloons, in 1849 [29]. If we try to compare manned aviation with USs, is easy to understand that the latters are incredibly more performing in all the missions that are too dull, dirty and dangerous for humans. Although they were originally used almost only in military operations, the rapid increase of new technologies opened the use of unmanned systems also to civil application. Actually, the most common are:

- Cargo transport
- Agriculture
- Forest fire detection and monitoring
- Scientific research
- Recreational use

In all this kind of missions the usage of drones is crucial, since they can perform tasks repetitively, fast and accurately [30]. Furthermore, if we look to the future, in the next ten years a massively increase in the usage of Internet of Things (IoT) is expected: in this scenario, a fleet of "on demand sensors" in the sky will have a fundamental role.
The analysis of the Unmanned Systems' level of autonomy combined with their numerical and technological growth shows why the actual legislation is completely inadequate: the new technologies allow us to imagine in a very near future a world in which a huge amount of drones will daily flight over the cities, without any kind of human pilot (nor on board neither in a Ground Control Station) at a quite low altitude and with complex tasks to accomplish. In a so confused environment, it seems completely useless still imposing a sort of driving license for drones' pilots (like Italian laws do, [3]) while the need of a new kind of flight controller arises. This thesis will address this problems almost from all its aspects, as will be shown in the following section and in the rest of the work.

1.2 Thesis Structure and Contribution

If we imagine a future urban scenario, the growth in the use of small Unmanned Aerial Vehicle seems on one hand unstoppable, both for the cargo usage by delivery company and for the increasing recreational use, while on the other it arises safety problem actually not regulated coherently in any legislation.
Starting from this new environmental, social and technological background, a research project has born inside the TIM's Joint Open Lab at Politecnico di Torino, and this master thesis is going to shows just a single part of it. In this section, objectives, assumptions and structure of the work will be presented, highlighting the results of the research group in parallel with a detailed description of the main themes that instead will be treated in this particular thesis.
The main aim of the overall work is to guarantee safety during the flight of fully autonomous systems operating in a urban environment. It's crucial to understand that when talking about safety, we are referring to the safety of people on ground that can be involved in an UAV crash. This concept is fundamental, since it underlines that the center of this work are the people, not drones.
In order to start facing this problems, a completely new Traffic Management system (which will be formally defined in chapter 3) has to be developed. In particular, it will have to perform two different functions:

1. Developing all the techniques needed to have an autonomous flight, as path planning, flight control systems, collision avoidance systems, ecc.

2. To guarantee that the flights will respect the safety standards imposed by the local flight authority.

So, in parallel with the development of the flight's control techniques, a new metric has to be introduced in order to correctly evaluate the safety (or the risk) of an unmanned mission. This will be the core of the next chapters and of this specific thesis work.
It's important to underline that we are facing a completely new situation in the field of air transport: for the first time quite cheap air-vehicle are going to fly over cities without any human supervision on it. Since this systems are low budget, and their price is going to decrease in the next years, they will be both very easy to be bought (also by common people) and built with low quality standards. Moreover, the operational field in which they are going to move is characterized by a real high population density, that means an high probability of injuries for people if a crash happens. This is not acceptable.
The aeronautic industry has always been very restrictive in evaluating the safety of an aircraft, and thanks to this it's actually capable to produce items with a failure rate equal to $\frac{1}{10^9}h$. That's said, in order to receive the flight's authorization from the proper national agency,it seems crucial the introduction of a control system capable to measure and guarantee the correct level of safety. The main features that this system has to implement are:

- Statistical and historical based risk assessment. It has to be capable to evaluate a priori the risk of flight over a zone relying on the information it has on it.

- Real time risk assessment. The control system must update its risk analysis if the available data are changed during the mission execution.

- Registration and Identification services. The system has to provide a "registration service", for all those aircrafts that wants to join it. Beside, it has also to constantly monitoring the activities and the status of its drones, in order to be aware of eventual critical situations.

- A priori and real time path planner. Starting from the risk analysis, a path planner have to be dsegned in order to choose the safest path among all the possible ones. Moreover, it has to be capable of changing in real time the established waypoints if something happens during the flight (for example, the risk of an unpredicted collision).
- Trajectory follower, in order to follow the waypoints provided by the path planner. This controller must consider the drone's dynamic and provide the correct control input. Furthermore, a robust controller should be capable to detect failures in the flight system and apply a correction.
- Collisions avoidance algorithms, to ensure that each drone is able to avoid obstacles during the flight

All this characteristics have to be developed always focusing the work on safety, and can be summarized in the construction of an Unmanned Traffic Manager (UTM), which in the last year has been the central project of many different aviation agencies around the world (see chapter 3 for detail on state of the art).
Thanks to the Joint Open Lab's experience in this fields, we decided to design a brand new UTM, that differently from the others will rely and fully exploit the capabilities of cloud computing, which also grew enormously in recent years. In chapter 3, a detailed explanation of the reasons why we choose a cloud based framework will be provided. Introducing a Cloud Based UASs Traffic Management (CBUTM) we will be able on one side to evaluate and consequently mitigate the risk, on the other to handle all the critical situations (as collisions or internal failures) that can happen in a crowd flight scenario.
Actually, the Cloud-Based UASs Traffic Management has been realized in almost all its parts, and it is capable of providing all the features exposed before. In particular, this thesis works will focus on the development and integration in the overall framework of a Risk-Aware Map Manager.
It's role, in addition to being complex, is also at the base of the traffic managing system itself. It has to:

- Model the environment in which the mission happens. It has to produce a map, consistent with the geographical one, where eventual critical situations (as buildings, for example) are marked as "no-fly zone".
- Make a risk assessment procedure, according with the most innovative techniques in this field. The risk evaluation has to be function of the environmental map already seen, of the constructive parameters (i.e. dimensions) of the Unmanned Vehicle and of the vulnerability of the human body. The results will constitute the so called risk map, which associates a risk value to each point of the map (see chapter 3).
- Update itself in real-time. In order to really ensure safety, it must be always consistent with the real operative scenario, updating its data if something changes. In a crowded area, with a quite small flight's height, it must know everything about the static and dynamic objects of the environment, otherwise the fatal event will be unavoidable. The idea of a real time map capable to update itself is crucial to have a safe managing of the airspace.
- Establish a metric of path's risk evaluation, being capable of comparing different trajectories, between them and with a safety standard imposed by the national flight authority.

To understand the importance and the function of this platform, it is crucial to show some criticism related to the operational environment we are considering.
The first issue in ensuring safety in an urban scenario is due to the huge number of people that can be involved in a crash. As we will discuss in chapter 2, the risk is (obviously) strictly related to

the population density, which in the cities, and especially in the bigger ones, reaches values in the range $[5000\frac{ab}{km^2}, 8000\frac{ab}{km^2}]$. Furthermore, cities are often involved in big events, that can gather huge crowd of people in a relatively small area. In order to reduce the risk, the traffic manager should be aware of this and taking into account when and where are the most populated areas.
The second problem to face is the collision avoidance, both with static and dynamic objects. In the first case, the urban environment implies the presence of buildings, very often with different heights and structures. Since we are in a urban scenario, in order to reduce the risk the UAV that we are going to consider will be very small and light. However, this implies a quite small flight's height, that will be around $25m$, and consequently high risk of collision.
That said, the need for a Risk-Aware Map Manager becomes evident, since it has to guarantee safety for people before the beginning of an Unmanned Mission (through a coherent risk analysis) and during its progress, constantly updating its values and changing the drone trajectory if something dangerous happens.
Furthermore, the importance of a cloud infrastructure, which main characteristics will be shown in chapter 3, have to be underlined, since it allows to suppose a huge computational power, as well as a real-time update of information from the Internet and a knowledge sharing between all the elements that joins the Traffic Manager.
In this thesis the construction of this kind of system, with all its characteristics and issues will be carried on. In particular, in chapter 2, the state of the art about risk assessment will be discussed, and when possible improved. Then, in chapter 3 the overall structure of the Cloud-Based UASs Traffic Management will be presented, firstly with an overview of the most important sub-systems that constitute it, and then focusing in particular on the Risk-Aware Map Manager structure. In chapters 4 and 5 the implementation of the CBUTM, in a ROS environment, will be discussed and then a set of simulation, to validate our approach, will end the work. In particular, chapter 5 will show the behaviour of the whole system, considering also the contribution given by my colleagues. Finally in chapter 6 the most important results obtained will be resumed, and a brief set of future works that should be done to improve the CBUTM will be presented.

Chapter 2

Ground Impact Risk Modelling

2.1 Unmanned Missions Risk: Introduction

2.1.1 Main Concepts and Risk-Aware Map Manager

Following what has been exposed in previous chapter, this part of the work is going to analyze the concept itself of risk for an unmanned mission, proposing some definitions and analysis methods. Furthermore, the idea of a Risk-Aware Map Manager (which is the core of this thesis project) will be introduced, in order to provide an initial presentation of it, to better understand the risk assessment equations that will be shown in the rest of this chapter.
First of all, when discussing about risk we are referring to the time frequency in which the drone causes deadly (or very serious) injuries to one ore more people on ground [6]. It's important to focuses that the risk is not to be intended for the drone (for example a crash or an in-flight collision) but for the people on ground. Our analysis aims to avoid hazards for all human beings that can be involved in an UAV's accident, not to avoid the accident itself. Obviously, as it will emerge in the following chapters (and thesis' works), avoiding drones' crashes is one way (but not the only one) to prevent fatal injuries. Since the avionic industry worked on safety for many years, actually there are many ways to model the risk, each one with a specific level of accuracy [6].
Although different, each of them has been built taking into account four main standard criteria that cannot be forgotten during the creation of a completely new model [9]:

- Transparency.
- Consistency.
- Clarity.
- Reasonability

Furthermore, it's important to be very careful to not underestimate the risk, always preferring a more conservative model respect to one that does not take into account all the available elements. The main question we must answer at the end of a risk analysis is if a given Unmanned Aerial Vehicle (UAV) can guarantee a required level of safety when flies over a given geographical zone in a given time window. Moreover, if this analysis should give a negative result, our system must be capable of suggesting one or more countermeasures to be taken in order to increase the mission's safety. This last part is called risk management [6]. The procedure to follow is usually organized in four steps [9]:

Mission Definition and Hazard Identification A description of the mission has to be provided and consequently a definition for the safety bounds is derived. In this particular scenario, since we are working with civil operations in an urban environment, the risk's limit is imposed by the national flight authority and can be considered as constant. Practically, this means that the Flight Agency has to decide a maximum rate of victim per hour, which will be used by the Cloud-Based Traffic Manager has upper-bound for every mission.
Then, all the possible hazards that can happen during the flight have to be identified.

Risk Assessment Evaluate for every hazard the corresponding risk value.

Risk Reduction and Managment The overall risk is compared with the safety bound. If greater, some countermeasure have to be taken to reduce it.

Risk Acceptance Once the risk is below the limit, the mission is approved and can starts.

Please notice that the hazard analysis can be very time consuming, especially for what concerns the drone's internal failure, and there are already many tools to perform it. It's not matter of this work to identify all the possible hazards, but instead aggregate them into a coherent risk value. This concept will be clarified in next chapters, however we can suppose to divide hazards in two categories: due to internal failures and due to external causes.
Finally, once the risk analysis has ended a Risk-Aware Map Manager must create the corresponding risk map. It's important to underline that, though similar, the risk assessment and the risk map generation are two different process: the first, more theoretical, must be followed by latter, more practical. The implementation of the risk map generator will be the core of the last part of this work, while in this chapters the most innovative techniques to perform the risk assessment will be discussed and possibly improved.
In our idea, the risk map have to be function both of space and time: while the first is evident, the latter aims to takes into account the variation in time of the elements that practically affect risk: for example, the number of people present in a zone significantly varies in function of the day's hour.
The creation of such kind of map allows the path planner to find the low risk path for the UAV to accomplishes the mission.
In this context , the need of a Risk-Aware Map Manager (RAMM) arises. As an independent "block" within the Cloud-Based UASs Traffic Management, its role is basically to control and merge all the information available which affect risk for drones, then commute them into a coherent "map of the risk" for the drones' flight. The RAMM has to manage in parallel static and dynamic geographical structures, providing for each Unmanned System (US) its own risk-map (also function of its constructive parameter). Moreover, it has to work in real-time, updating its parameters when something change.
In the rest of this chapter a quantitative analysis of the risk is proposed, then in chapter 3 the structure of the Risk-Aware Map Manager is presented, describing all its basic principles. Finally, in chapter 4, an implementation of the concepts contained in 2 and 3 will prepare the reader to the results obtained through a software simulation in chapter 5.

2.1.2 UAVs' Types of Accidents and Equivalent Level Of Safety

Before starting the quantitative analysis to measure the risk, a brief introduction is needed to present main UAVs' type of accident.
According to [6] during missions the Unmanned Aerial Systems are involved in many types of hazards, that can cause three different kind of accidents:

Drone Involuntary Mobility This category includes all the accidents that happen when the drone is on ground and still have to take off. They are due mainly to human inattention or error and involve the flight's crew.
Since the scenario we want to analyze doesn't care of the drone before the take off, this kind of accident will no longer be treated in this work. Furthermore, the better ways to avoid fatal injuries in this case just consists in applying correctly all the security protocols for the drone managing [7].

Mid-Air Collisions This category includes all the accident due to in flight collisions, ad for example between two UAVs, or against a building (the latter can be very common in urban environment). Actually at the state of the art the study of mid air collision's risk mainly concerns the "flight's victims" directly involved in the crash, as for example people present on a line flight. Since we are just considering Unmanned vehicle, and we can easily suppose to not impact with big airplanes in a urban environment, the analysis of mid air collision will treat only the people on ground that could be killed by the falling debris following a collision. [8]

Early Flight Termination In this case we consider all the hazards that can cause a loss of control in drone's flight and consequently an anticipation in the UAV's landing. This operation can be partially controlled or not and can ends on ground or in water: it's interesting to notice that the cloud could control this process in order to impose a desired landing zone or certain velocity in order to reduce the probability of fatality [8]. However, as explained in previous section the worst case scenario will be supposed during risk analysis, imaging that no emergency control algorithm has been implemented.

This categories represent three different ways in which an unmanned vehicle can cause (directly or indirectly) serious or deadly injuries to the people presents in the accident's area.
It's interesting to notice that when talking about injuries we have to consider both the physical, as for example the pollution following the diffusion in the area of an harmful payload, and the social ones: an high number of accident, though without death, will cause a repulsive feeling in common people respect to UAVs' missions and consequently a limitation on them [6].
Once again then, the need of a framework to quantify and reduce risk arises.
It has been interesting to find out that there is still not a commonly accepted definition of risk, since its mean changes according to the operative scenario and the objectives that have to be achieved. However, in the avionic environment it is often used the following one:

Definition 3. It is called risk f_F of an Unmanned Aerial Mission the frequency of fatalities, in term of victims per hour of flight, that a given drone, in a certain area will produce.

How to evaluate this number is the core of the risk assessment procedure. Key concept that derives from the definition is that the risk have to be a frequency, so it must be linked with time t. As said before, we will consider the risk concerning two kind of accident that both can cause on ground victims: the correct evaluation of the number of fatal injuries per flight hour that a given US can produce is the mission's risk itself [6]. First of all the two accidents' risk evaluation will be studied separately in order to give a proper metric to both of them. At the end, they will be merged and an overall ground impact risk model will be provided.
In order to be considered safe, the overall mission's risk value must be lower that the objective upper-bound imposed by the flight's national authority. Practically, this limit can be used to differentiate safe mission and from unsafe ones, so it's interesting to understand how it can be evaluated.
The fundamental principle used by the flight's agency is the "Equivalent Level of Safety" (ELOS) with the aircraft driven by humans. According to [11] this means that "all the operations or tests

conducted by unmanned aircraft must demonstrate to have a risk for human life not higher then the same operation conducted with on board pilot".
This definition doesn't impose to use the same safety parameters actually used in classical aviation: this one is on one hand more dangerous, since involves bigger systems with hundreds of people on board, while on the other can be considered safer since doesn't relays only on control systems whose failure will produce the aircraft's fall (in this case, the man in the control loop is still considered a plus).
A better way to evaluate the correct maximum risk's value is based on statistical studies [6][12] taking in consideration only on ground victims instead of flight's passengers. Starting from this, it's possible to increase the typical risk lower-bound, which is $10^{-7}h^{-1}$ for classical manned aviation,moving to a maximum fatalities rate for US mission in the range of:

$$f_{F,Max} \in [10^{-5}h^{-1}, 10^{-4}h^{-1}] \tag{2.1}$$

That means one fatal event almost every 10^5 hour of flight. However, and this will emerge along all this thesis works, the risk-aware system we designed is independent from the value imposed as upper-bound for the risk: it will be a task of the National Flights Authorities to calculate it, with its own principles and parameters.
At the actual state of the art, a big range of UAVs is provided by the producers, with different dimensions and characteristics. According to [31], Unmanned Aerial Systems can be categorized in a variety of ways based on vehicle attributes as type of aircraft(fixed wing, multi-rotor,ecc.), functionality, weight, speed,ecc. In our case, since we are going to produce a framework useful for multi purpose UAVs (so we are not considering a specific kind of mission), it can be interesting to have a look to a dimension based classification of the most common drones, in order to better understand the items we are going to work with.
Figure 2.1 shows the NATO UAVs' classification, dividing them in classes and categories and providing some examples: According to what has been already exposed, to have a safe urban flight only Micro and Mini UAVs will be allowed to operate in our framework, since they can guarantee a small weight (under 2 kilos) in parallel with a satisfying mission radius (25 kilometers). From now on, we will suppose to operate only with this kind of objects, assuming that no dangerous payload is carried on.
That's said, the quantitative risk analysis can begin.

2.2 Early Flight Termination Risk Modelling

The main aim of this section is to deduce an analytic expression that, given the flight conditions, correctly evaluates the number of victims per hour due to an uncontrolled landing of the UAV. The unexpected flight end can be caused by different problems, like internal failures or external causes,but without considering mid-air collisions.
Thanks to the works provided in [6] and the fundamental principle exposed in previous sections, the risk model is obtained as:

$$f_{F,EFT} = N_{exp} \times P(fatality|exposure) \times f_{EFT} \tag{2.2}$$

where:
$f_{F,EFT}$ = fatalities frequency due to an early flight termination. Measure Unit: $[\frac{people}{h}]$
N_{exp} = number of people exposed to the accident. Measure Unit: $[people]$
$P(fatality|exposure)$ = probability that a person involved in the UAV's crash will suffer fatal injuries
f_{EFT} = frequency of failures that cause an unexpected end of the flight. Measure Unit: $[\frac{1}{h}]$

NATO Classification

Class & Weight, w (kg)	Category & Weight , w (kg)	Normal Employment	Normal Operating Altitude, h (ft)	Normal Mission Radius (km)	Example Platform
Class I w < 150	Small w > 20 kg	Tactical Unit (employs launch system)	h ≤ 5000 AGL	50 (LOS)	Luna, Hermes 90
	Mini 2 ≤ w ≤ 20 kg	Tactical Unit (manual launch)	h ≤ 3000 AGL	25 (LOS)	ScanEagle, Skylark, Raven, DH3, Aladin, Strix
	Micro w < 2	Tactical Patrol/section, Individual (single operator)	h ≤ 200 AGL	5 (LOS)	Black Widow
Class II 150 ≤w ≤ 600	Tactical	Tactical Formation	h ≤ 10,000 AGL	200 (LOS)	Sperwer, Iview 250, Hermes 450, Aerostar, Ranger
Class III w > 600	Strike/ Combat	Strategic/ National	h ≤ 65,000	Unlimited (BLOS)	
	HALE	Strategic/ National	h ≤ 65,000	Unlimited (BLOS)	Global Hawk
	MALE	Operational/ Theater	h ≤ 45,000 MSL	Unlimited (BLOS)	Predator A, Predator B, Heron, Heron TP, Hermes 900

Figure 2.1: NATO UAVs' classification

This apparently simple formula contains all the interesting parameters that affects an unmanned mission. Furthermore, it combines them in a way that guarantees a risk analysis:

1. Coherent with the statistical data actually available.
2. Not expensive from a computational point of view.
3. Easy to extend in case some new interesting parameters emerges.
4. Independent from the dimension of the considered area. The analysis can be performed both with very small and big resolution.

Finally, it is the best compromise between a too detailed analysis and an inconsistent one. All the elements appearing in 2.2 will be explained in following sections, in order to explore the detail to perform risk evaluation. In any case, it's important to underline some interesting features.
First of all $P(fatality|exposure)$ it's probably the key point of the equation. First introduced by [6] and then expanded by [17][18] it takes into account drone's kinetic energy, human body's vulnerability and geographical sheltering factor to obtain the probability of having fatal injuries after an Unmanned System collision with people.
Then, the number of exposed people N_{exp} can be easily evaluated as $N_{exp} = \rho \times A_{exp}$ where ρ is the population density and A_{exp} is the drone's impact area, whose exact formulation will be

provided in the following section.
For what concerns the f_{EFT} term, it has been a crucial element in the discussions that preceded the beginning of this work. As explained before, this element contains the information about the time frequency of the drone's ground impact: practically, it introduces in the equation the time element.
In the first stages of this thesis, many attempts were made to provide a framework able to correctly evaluate this parameter working on the elements that could affect the crashes frequency. The estimate of f_{EFT} was obtained thanks to a probabilistic study, merging external causes acting on the drone (as for example bad weather conditions) with the probability of an internal failure. This approach however showed two criticism that couldn't be ignored:

1. Evaluate the ground impact frequency starting from the hazards that can causes it strictly implies the complete knowledge of all the possible hazards and the quantitative impact that each of them has on the probability of having a crash. Ignoring even only one hazard or underestimating its impact will falsify all the risk analysis, since the risk will be under or over estimated.
   An example of this criticism turned out when trying to provide the probability of having a crash due to the high wind speed: it was possible to use the maximum wind speed predicted by the weather report for the period in which the mission should be performed, but however also the more detailed weather forecast cannot consider the wind's gusts, which are the real cause of crashes. Finally, this means that the risk analysis performed was quite meaningless.

2. Trying to give an estimate of f_{EFT} means working with the probability that each hazard can happens, and that after it will cause a crash of the UAV. The final result (whose trustworthiness has been discussed in previous point) turn out to be a probability too, without any measure unit. The lack of time element is a critical issue, since as it's been explained in previous section the risk must be measured in fatal injuries per hour (according to the actual standards). The need of time in the equation 2.2 will be explained in next chapters, when a method to evaluate the overall risk of the mission will be provided.

After this considerations, it seems quite clear why we decided to avoid a complete pure probabilistic approach in evaluating f_{EFT}. Instead, according with [6] and [10] a statistical way has been preferred.
Practically, the number of ground impact per flight hour is established looking at the history of the drone: each producer have to makes a quite big number of test in order to produce the more correct evaluation of the f_{EFT} of its drone. If this value is not provided by the UAV's builder, it can be still estimated after a proper number of flight hour, using as initial value a very high failure's frequency. Finally, if a new model of UAV is designed starting from an already tested one, it will be possible to initially assume the f_{EFT} of the new drone equal to the one of the older UAV and use it for the risk analysis. Obviously, also in this case it will be possible to update f_{EFT} after some hour of flight.
Although they seem simple, or not completely clear at this point, the concept described in the previous lines are really crucial to deeply understand this work: we moved from an hazard analysis, very technical and hard to perform, to a risk analysis in which the ground impact frequency is simply assumed to be known. This could initially seems barely a simplification but it's instead the right change of point of view, since the aim of this framework is not to validate the work developed by our aerospace colleagues but instead to provide the instruments to manipulate and manage the information given by them in order to guarantee a safe mission.
According to [17] and [18], a realistic value of f_{EFT} for light and cheap drones is $f_{EFT} = 0.01 \frac{impact}{h}$. For this reasons, that number will be used in following chapters when simulations of the Map Manager work will be provided.

An interesting modification in equation 2.2 can be introduced in case we have very detailed information about the sheltering of an area and the number of people in the zone that can be considered sheltered from a possible fall of the UAV. This study can be performed for example merging statistical/historical data with the mobile phone signals, that can be analyzed by the cloud system in order to estimate the number of people really present and covered.
Once the level of sheltering of each group of people is known, 2.2 can be re-formulated [6]:

$$f_{F,EFT} = \sum N_{i,exp} \times P_i(fatality|exposure) \times f_{EFT} \tag{2.3}$$

Where $N_{i,exp}$ and P_i refers to the i-th group of people. However, in the rest of this work the risk formula used will be 2.2.
In the following sub-section, all the elements that compose risk equation will be discussed deeply, starting from their theoretical definition till the way it's possible to calculate them. The approach used is a top-down one: starting from the higher level expression of the risk (2.2), let's go down in the details of each part of it, showing how to calculate every term.

2.2.1 Evaluation of Exposed Inhabitants: N_{exp}

When evaluating the risk of an unmanned mission, it's crucial to correctly estimate the number of people that an eventual crash would involve. Since this work focuses on avoiding human victims more than UAV crashes, every risk's measure must be linked with the number of inhabitants present in the overflown area. Obviously, minimizing the hazards that cause crashes is fundamental in order to reduce the ground impact risk, and this is why the other part of this research project will focus on cloud based algorithms to manage danger situations, as for example collision avoidance. As presented before, the number of people exposed to an accident, in a given geographical zone is:

$$N_{exp} = \rho \times A_{exp} \tag{2.4}$$

where:
N_{exp} = number of people exposed to the accident. Measure Unit: $[people]$
ρ = population density in given area. Measure Unit: $[\frac{people}{m^2}]$
A_{exp} = exposed/impact area. Measure Unit: $[m^2]$
We suppose an uniform population density(ρ) in the given area. Since the map resolution will be around $25m^2$ (refer to following chapters for detail) this assumption doesn't affect badly the risk assessment. Now, let's show how to estimate the last two parameters.
First of all, it's important to underline that A_{exp} is defined as the lethal area of the drone, and it have to be evaluated taking into account features like UAV's glide angle, dimensions and human mean height and radius [13]. A huge number of works have been published upon this topic, since there are several methods (due also to the specific aircraft or a particular scenario) to calculate the injuries-space of a falling drone.
According to [17], one optimal way to evaluate the impact area is the one exposed in FAA,2000:

$$A_{exp} = 2 \times (r_p + R_{UAV}) \times d + \pi \times (r_p + R_{UAV})^2 \tag{2.5}$$

where:
$r_p = 0.23m$ = mean value of human body radius
R_{UAV} = maximum linear dimension of the US
$d = \frac{Hu}{\tan\gamma}$ = horizontal distance travelled by UAV during the fall. Measure Unit: m
$Hu = 1.75m$ = mean value of human height
γ = glide angle
Please notice that in case the considered UAV is a quadcopter, then it is possible to suppose a

vertical fall that will impose $2 \times (r_p + R_{UAV}) \times d = 0$ since this term of the equation represents the horizontal component of the fall. For what concern the fixed wing drones, instead, a glide angle equal to $\gamma = 45°$ should be used. This first results shows that quadctopter-like UAVs can be consider safer then fixed wing, at least for what concern the impact area. The values chosen both for Hu and r_p are intended to modify the first version of 2.5 proposed in FAA,2000 in order to make it more coherent with real statistical values, as exposed in [18].
Then, a discussion about population density ρ is needed, that since from the begin of this project is been addressed as one of the key point where focuses the power of the cloud-based approach. Once again, the people and not the drones are the center of this work, so a wrong knowledge of this parameter would falsify all the rest of the risk analysis.
In the older works in the safety fields, like [6] and [19], it's suggested to suppose ρ as a constant, that in urban environment can be setted equal to $200\frac{people}{km^2}$. This number came out trying to consider all typical scenario of civil aviation, so its a mediation between moments in which the unmanned system flies over very crowded zones and ones in which instead it is over sparsely populated areas. It's obvious that such approach in our context, where the operative scenario is almost always a city, is too inaccurate, since oscillate from being too restrictive when the drone is passing over parks, rivers, ecc. and instead not conservative enough in all the other situations.
In order to obtain a more realistic ρ, the idea is to consider for every area the corresponding population density, already present on web: the cloud system can connects to internet (thanks to the TIM network) and from here extract updated data about the population density. Practically, with this second step we can move from an a priori estimate ($200\frac{people}{km^2}$) to a measured value of ρ, with a level of detail that can reach the district dimension.
This estimate,however, is still static and in some case not realistic. The main problems that arises are:

1. It takes in account only the people that live in a certain zone (this is how population density is commonly measured), excluding from the risk evaluation all the people that can be in the area for other reasons.

2. The resolution level of population density maximum reaches the district scale. No more detailed information (for example how many people live on a street or in a buildings) are available on internet.

3. Still doesn't take in account the time element. The day's hour in which the mission takes place is a crucial element, that drastically changes the number of people that the drone can involves in a crash. In order to understand this concept, just think at the same street at 12 am and 12 pm. Furthermore, this approach doesn't take in account all the special events, like concerts or football matches, that can't statistically be predicted and create big crowds that an UAV must avoid.

During the discussions inside the TIM's Joint Open Lab emerged the will to create a dynamic framework, capable to collect data and provide a real time risk evaluation. This system must start from a dynamic knowledge of the people that are in danger due to the drone's flight. The final result to which we are moving to is to change the classical expression of density from a function of space to a function both of space and time (the time interval τ in which the mission will take place). In formula:

$$\rho(x, y) \longrightarrow \rho(x, y, \tau) \tag{2.6}$$

Starting from the population density, also the risk will become dependent from the flight's time window.
Thanks to the cloud, it is possible to connect to the internet network in order to acquire data about

the areas the drone is going to fly over: this allows the knowledge of events whose participation can be estimated but also simply the historical information of the number of people usually present in a given zone at a given time.
Furthermore, an higher level of precision can be obtained, thanks to the new technologies available. Through it's network, the TIM group can give an estimate of the number of mobile phone connected to a particular cell. Merging this information with a known mean value for a given time hour, it becomes possible to provide an estimate in real time of the variation of people present in an area. The same approach is used by the most common maps provider in order to give an estimate of the cars' traffic on the streets.
Finally, the cloud framework will be capable of updating itself also during the drone flight, allowing the path planner to find in real time the best path. Moreover, it will share and conserve the new estimated $\rho(x, y, \tau)$, so that next time a mission will be planned in the same time and place its value will be already updated.

2.2.2 Estimate of fatalities probability for people exposed to crash: $P(f|e)$

The term $P(fatality|exposure)$ in the risk equation represent the probability (so it is an adimensional number in range [0,1]) of having very serious injuries after the impact of the falling drone with one or more inhabitants. To describe this probability, many studies tried to provide a model of the human vulnerability taking into account factors as position, age and physical conformation in order to estimate the damage that the body will suffer impacting with an Unmanned System at a given speed. The most interesting works in this fields developed in the last twenty years are [14] [15] [16].
Although this models will not be used, since the information they need to give a good estimate of $P(fatality|exposure)$ are too detailed and never available, their results have been the inspiration of the framework we are going to present.
The core concept is to suppose $P(fatality|exposure)$ as function both of the kinetic energy of the falling drone and of the sheltering factor of the area in which the crash happens, whose metrics will be provided later in this section. Thanks to the ideas exposed in [6] and [9] and modernized by [17] and [18], it is possible to calculate $P(fatality|exposure)$ as:

$$P(fatality|exposure) = \frac{1-k}{1-2k+\sqrt{\frac{\alpha}{\beta}} \times (\frac{\beta}{E_{imp}})^{\frac{3}{P_S}}} \tag{2.7}$$

Where the elements of the equation are:

P_S is the sheltering factor, defined in the interval $[0, +\inf]$, used to take in account the natural and artificial shelters that the area offers to protect the people from the UAV fall. Its mean value is 1 and increasing P_S correspond to decrease the $P(fatality|exposure)$. In the following paragraph a detailed description of how to evaluate this term will be discussed.

$k = min(1, (\frac{\beta}{E_{imp}})^{\frac{3}{P_S}})$ is a term properly studied in [6] in order to provide a better estimate for low level of kinetic energy.

E_{imp} is the kinetic energy of the UAV at the impact.

α is a term defined as the kinetic energy requested to have $P(fatality|exposure) = 0.5$ when $P_S = 6$ [6]. An optimal value to use is $\alpha = 34kJ$

β is the lower level of energy needed to have a fatal event in absence of sheltering. According to the previously presented studies on human body vulnerability, the correct value correspond to $\beta = 34J$

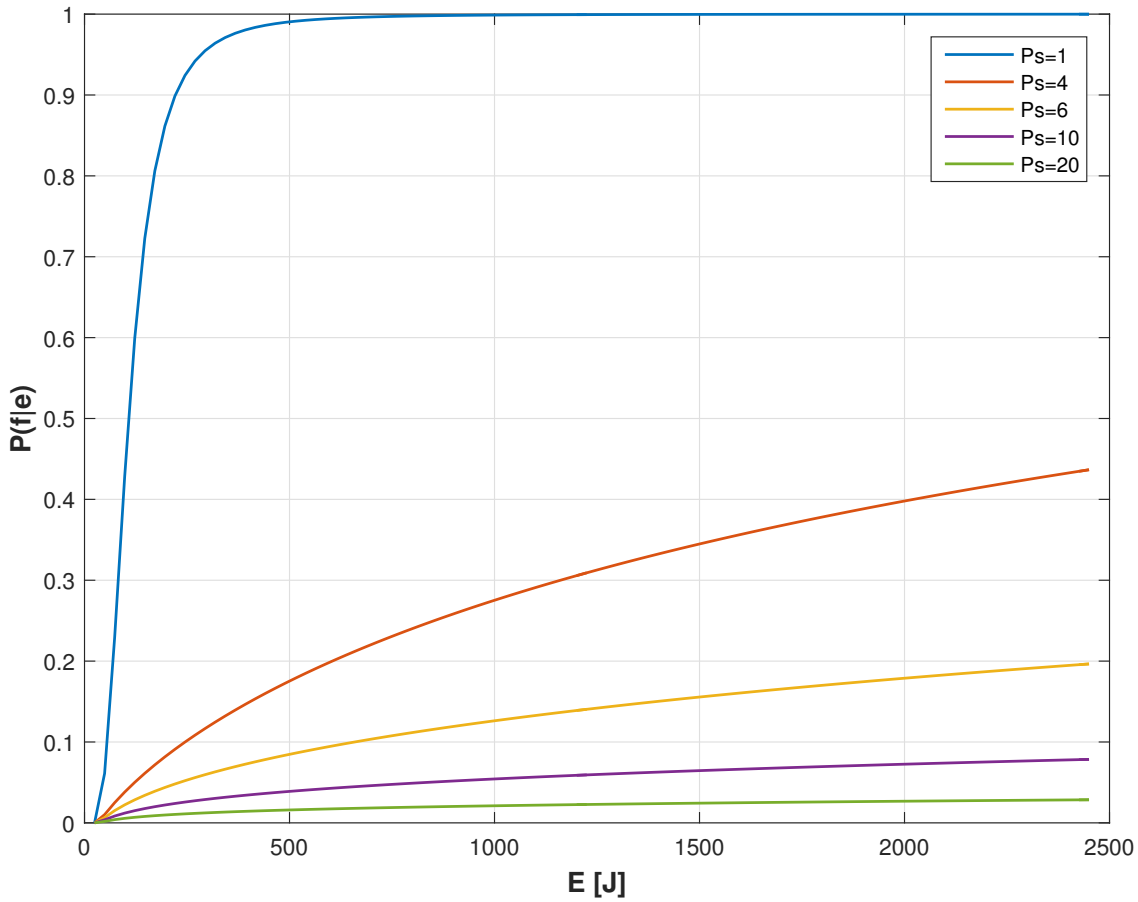


Figure 2.2: $P(fatality|exposure)$ evolution respect to Kinetic Energy and Sheltering

Figure 2.2 shows the behaviour of $P(fatality|exposure)$ varying Ps and E_{imp}: The main problem with this kind of formulation is the lack of a coherent metric in order to describe the sheltering factor. Actually, there is not a tested method to deal with it and this create issues in correctly evaluating $P(fatality|exposure)$. For example: what does practically mean $P_S = 6$?
In [6], that is the state of the art for what concern the sheltering factor computation, it is defined as:

Definition 4. The sheltering factor of a geographical area is its capability of protecting people on ground, through artificial or natural structures, from the fall of an Unmanned Aerial System.

This definition, more qualitative then quantitative, led to a quite confused situation in which P_S belongs to the range $[0, +\text{inf}]$, but this choice creates two problem: first we cannot deal with P_S as a probability (for example for people on ground to be protected from the UAV's fall) since it has to be from 0 to 1, and second it seems impossible to quantify P_S simply looking at the operative scenario, because there aren't bounds for it.
Once again, we must give to P_S a metric, such that it becomes possible to evaluate it according to some interesting features of the field.
In order to do this, let's have a look to the behaviour of $P(fatality|exposure)$ when P_S is varying. First of all, it's interesting to notice that when $P_S = 0$ and people are without protection, the

$P(fatality|exposure)$ behaves like a threshold: when the impact kinetic energy is less then 34J [9] the fatal event has zero probability, while when E_{imp} is over 34J then $P(fatality|exposure) = 1$. An operative scenario with $P_S = 0$ is considered to be very unrealistic, especially in an city, and practically no UAV will guarantee the safety standard in a situation like this.
When it's imposed $P_S = 1$ instead an $E_{imp} = 108J$ is requested to have $P(fatality|exposure) = 0.5$ while 1000J will guarantee the certainty of the fatal event. Obviously, also in this case under the 34J the $P(fatality|exposure)$ turns out to be null. For increasing values of P_S let's have a look at the following :

This computation has a very precise aim: to show that, though we can use infinite value of P_S

P_S	P(fatality | exposure) = 0.5	P(fatality | exposure) = 1
1	108J	1000J
4	4KJ	100KJ
6	34KJ	1MJ
10	3.4MJ	$+\infty$

Table 2.1: $P(fatality|exposure)$ evolution in function of P_S

in $P(fatality|exposure)$ in a real scale $P_S = 10$ is already an upper-bound. As a matter of fact, when the sheltering factor is equal to 10, 3.4MJ are required to have 50% probability of having the fatal event. In order to better understand this number, let's show a simple example. Since we are considering quite light unmanned vehicle, with a weight equal more or less to 1Kg, this value of kinetic energy can be obtained only with a speed in the order of $2600\frac{m}{s}$. Since a situation like this one is absurd, we can suppose $P_S = 10$ as the upper-bound of P_S. Practically, a zone with $P_S = 10$ have to be consider as a complete recovered area, where there is no risk for people inside. Concluding, in the rest of this work we will suppose $P_S \in$ [0,10]

2.2.3 Evaluation of Impact Kinetic Energy: E_{imp}

From equation 2.7 the need of evaluating the kinetic energy in the moment of impact has emerged. Many different solutions about this have been proposed, some more accurate take in account factors as air density or dragging, others simpler instead just use drone velocity and mass, in order to simplify the computation [6]. Looking to the works exposed in [17]

$$E_{imp} = \frac{1}{2} \times MV^2 \tag{2.8}$$

where:
M is the vehicle's mass
$V = \sqrt{2 \times g \times h}$ is the free fall speed from height h

2.2.4 Estimate of Sheltering Factor: P_S

The sheltering factor P_S is the item used to model and evaluate the protection level that the scenario offers, through natural or artificial means, to the population on ground. Some example of sheltering factors are the trees, rooftops,etc.
Although according to the previous paragraph it is not a probability, let's start the discussion on it supposing P_S as the probability that the UAV's fall will be stopped before it will hit someone. The history and the state of the art of this parameter is quite interesting, since the definition of

its metric has been a problem form the beginning of the risk analysis.
In the first approach developed, this factor didn't appear, practically assuming that no protection was available [10]. The reason behind this choice was that one of the first risk models for unmanned flight was taken from the aerospace field, where spatial debris were considered: due to the enormous dimensions of them, it was useless to consider the environment protection. Such a model was however too conservative and didn't allow to anyone the flight, since also very light UAV turned out to be too dangerous.
The introduction of the sheltering factor happened in [6], and as reported before, it was defined in the range $P_S \in [0, +\inf]$. This changed completely the study of the risk: for the first time the protection, and this implies the characteristics of an area, were considered when evaluating the risk, practically reducing it. Since we are dealing with high density zones, it is crucial to find a way to reduce risk, otherwise it will be impossible to satisfy the risk bounds imposed by the national authorities. Especially in the urban scenario, were there are many buildings (which roofs can cover people) and light drones, the sheltering factor seems to be crucial in order to produce a correct risk analysis.
Buildings, trees but also vehicles and other objects can drastically reduce the probability of having a fatality, simply absorbing all or a part of the energy of the drone. For this reason, an analysis of the kinetic energy required to penetrate inside an object is needed, taking into account the materials with which it was built. After that, a way to map the covering skills of an area in the range [0,10] have to be found.
First of all, let's define the sheltering coefficient C_S of a structure as its ability, once hit, of reducing the drone kinetic energy. It's important to underline that we are not considering just the ability of a structure to stop the UAV, but also, in case it is not stopped, the amount of kinetic energy reduced by the structure. Once again, this analysis should be conducted taking into account the materials and the way a structure is built and also the minimum kinetic energy needed to penetrate it. A complete analysis in this sense is contained in [9], while a simpler but however interesting resume is the one produced by [17] and [18]. The results, taking into account the most common features of an urban scenario, are exposed in this table:

C_S	Structure's Tipology
0	Free Area
0.25	Shallow and Slightly Leafy Trees
0.5	Tall and Leafy Trees
0.75	Residential Buildings
1	Reinforced Concrete Buildings

Table 2.2: Sheltering Factor definition

Practically, $C_S = 1$ means a building that completely stops the UAV's fall, while $C_S = 0$ will be used for the completely open areas. Obviously, this is a very simple analysis since the ability of resisting at the impact with a drone depends on the kinetic energy and dimensions of the UAV itself. This means that in this works, as in the ones that preceded it like [6] [17][18], we suppose to consider only drones that cannot penetrate reinforced concrete buildings (and it is quite credible, due to their typical dimensions).
Once C_S is known, the sheltering factor value can be calculated as:

$$P_S = C_S \times \frac{A_C}{A_{tot}} \times 10 \tag{2.9}$$

where:
C_S = Sheltering Coefficient.
A_C = Covered Area of the zone. Measure Unit: $[m^2]$
A_{tot} = Total Area of the Zone. Its dimensions depend on the map resolution. Measure Unit: $[m^2]$
The multiplicative factor 10 is used to scale P_S, as to make it compatible with the 2.7 equation. Moreover, the term $C_S \times \frac{A_C}{A_{tot}}$ evaluates the probability that the drone falls in a covered area times the capacity of the zone itself to reduce its energy.
One way to modify equation 2.9 is to consider, inside the same area A_{tot}, different values for the sheltering coefficient C_S. Obviously this operation can be done only if very detailed information about the sheltering of the zone are available. In this case, since we are dealing with mutual exclusive events (the drone can falls only in one point), the probability of the union is equivalent to the union of the probabilities [20], weighted by the corresponding C_S factor. Then:

$$P_S = (\sum_{i=1}^{n} C_{S,i} \times \frac{A_{C,i}}{A_{tot}}) \times 10 \tag{2.10}$$

Where $C_{S,i}$ is the sheltering coefficient of i-th sub-zone with area $A_{C,i}$, while P_S is the overall sheltering factor. In this way, we can exploit all the power of the cloud, using all the information available.
Once both the impact kinetic energy and the sheltering factor are known, the $P(fatality|exposure)$ term can be evaluated, and consequently it becomes possible for the cloud system to evaluate the risk for people on ground to be deadly injuries due to an unexpected end of the flight.
It seems interesting to plot the evolution of $P(fatality|exposure)$ respect both to the sheltering factor and the kinetic energy, using this time a set of typical operative values. In our idea, the cloud based mission manager should work in a urban environment, where only small and light UAVs are allowed to operate. According to the most common UAVs' datasheets and also to [28], we can assume to move from the littlest UAV ($M = 0.1kg$ and $R = 0.1m$) to the biggest one ($M = 1kg$ and $R = 1m$) Supposing to have a flight's height of $25m$,the results is that we will deal only with kinetic energies from $60J$ to $250J$. This kind of analysis is useful in order to plot $P(fatality|exposure)$ and to understand the order of magnitude we are going to manage (figure 2.3). This plot shows that though the value of $P(fataluty|exposure)$ is defined in the range [0,1], practically it is bounded below 0.4. Moreover, it appears crucial the role of the sheltering factor, which in the extremest cases (with Ps moving from 7.5 to 1) reduces the killing probability of the UAV around the value 0.01. This numbers are perfectly coherent with the risk analysis we expected to perform, since they will guarantee the flight only in very sheltered environment or for very light drones. However, this kind of study will be better discussed in the Simulation chapter.
With the sheltering analysis, ends the study of the risk associated with an early flight termination of the drone. Practically, what we modelled in 2.2 and in the other equation is the number of victim per hour of flight that the UAV can causes, considering only the situation in which the crash on ground is due to an internal failure of the drone or to an external event without considering in-flight collisions.
The results of this analysis, mixed with the ones proposed in the next section, will be discussed in the Simulation chapter.

2.3 Mid-Air Collisions Risk Modelling

In this part of the work we will try to provide a way to model the risk, for the people on ground, to suffer deadly injuries due to the impact with an unmanned system, whose fall is caused by a mid-air collision. Once again, according to the National Transportation Safety Board ([12]), when we are talking about mid-air collisions we refer to:

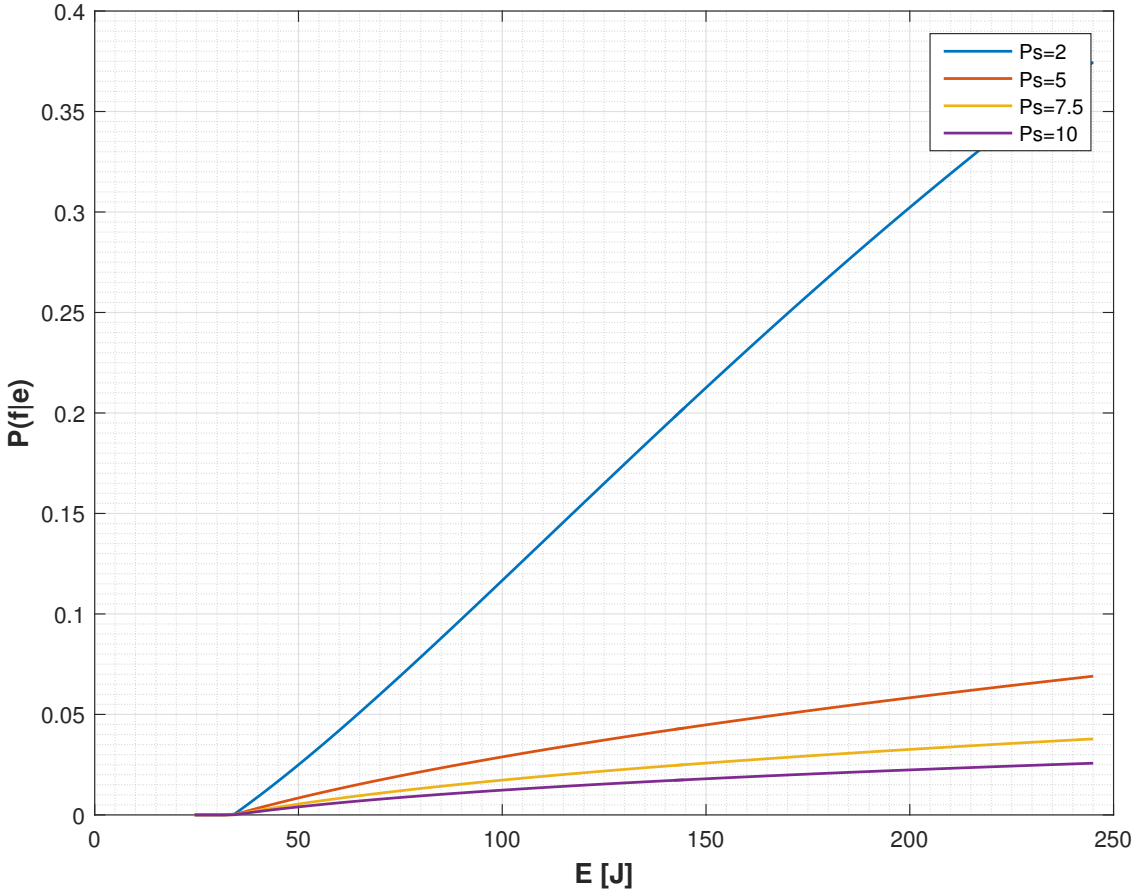


Figure 2.3: $P(fatality|exposure)$ evolution

1. Impact between two flying UAV
2. Impact with a building
3. Impact with other obstacles, like trees, birds and electric lines

The risk modelling of this kind of situations has always been very complicated, especially in a quantitative way since it requires the knowledge of a lot of information very often impossible to know a priori, as for example the exact trajectory that all the other UAVs will perform during their flights [6]. We are not going to consider possible collisions with airplanes, which will cause many more death since they have passengers on board. This allows to make some simplification in modelling the risk.
First of all, the only possible victims are the people on ground involved in the crash, in particular the ones that are hit by one or more of the debris following the mid-air collision [8]. So, as before, in the risk evaluation the number of people present in the zone of the crash have to be considered. Furthermore, we can assume that the debris falling on people has the same probability of cause deadly injuries that the whole UAV: this can be easily proved because we don't know how the drone will break and in how many different parts, so we simply have to consider the worst case scenario, in which there is a main debris with almost all the initial momentum of the drone. Moreover, if we suppose the kinetic energy doesn't vary after the mid-air collision, the result is that also the

$P(fatality|exposure)$ doesn't change.
According to [6], the frequencies of fatalities due to in flight accidents can be modelled as:

$$f_{F,MAC} = N_{exp} \times P(fatality|exposure) \times f_{MAC} \tag{2.11}$$

where:
$f_{F,MAC}$ = frequency of fatalities due to mid-air collisions. Measure Unit: $[\frac{people}{h}]$
N_{exp} = number of people exposed to the accident. Measure Unit: $[people]$
$P(fatality|exposure)$ = probability that a person involved in the UAV crash will suffer fatal injuries
f_{MAC} = frequency of UAV mid-air collisions. Measure Unit: $[\frac{accidents}{h}]$
Since we are still considering the on ground victim, the maximum acceptable value for $f_{F,MAC}$ remains the same of previous section (f_{Max}), and have to be evaluated with the same main criteria.
Looking at equation 2.11, it's evident the similarity with formula 2.2: for this reason, three out of four elements of the previous equation have been already explained, and the next section will focus only on the evaluation of the mid-air collision rate f_{MAC}.

2.3.1 Estimate of the Mid Air Collision's Rate: f_{MAC}

The term f_{MAC} , whose measure unit is the number of accidents per flight's hour, is needed in order to have an estimate of the number of collision that can happen during a mission. In this case, we are just considering in this parameter the collision with other UAVs, buildings,and others unpredictable obstacles. In the estimation of accidents' rate, the most common approach is to use statistical methods based on a huge database of the last years' accidents. The results is a very small number, in the order of $10^{-7}\,\frac{accidents}{h}$ [6][12]. In formula, we obtain:

$$f_{MAC} = f_b + f_{UAV} + f_{TBE} \tag{2.12}$$

where:
f_b is the collisions rate with buildings. In this term we consider all the collisions that could happen between the UAV and not expected buildings. It's important to underline that all the known structures will be handle as no-fly zone by the risk map (this procedure will be explained later on) so f_b has to be used only to parametrize the number of collisions with buildings that are due to a lack of precision (or error) in the geographical map.
f_{UAV} is the rate of collisions with other Unmanned Systems flying over the same area.
f_{TBE} is here to parametrize all the unpredictable collisions, as trees, birds,ecc.
Obviously, the measure unit for all this three elements will be the classical one for the frequency.
Both f_b and f_{TBE} can be only estimated in a statistical way, after many hours of flight on the same area. Since both this parameters model apparently unpredictable events, it could seem obvious to include the first one in the latter. In our opinion, instead, though they are effectively working on casual events, they have however a complete different meaning: while f_{TBE} measures unrepeatable and really casual events, f_b instead describes fatal accidents due to a lack or simply wrong information. For example, a building with a known height of $30m$ that is instead tall $40m$. This will produce a series of repeatable crashes that have to be avoided by the cloud-based UASs traffic manager. According to this, it's important to maintain this two parameters split, since they have a different meaning for the cloud system that have to handle both differently in order to choose the safer path for the UAV.
In next section, the theories behind the evaluation of f_{UAV} will be explained. The other two parameters instead, that can be only estimated through an empirical series of flights, will not be analysed again.

2.3.2 Evaluation of the Collisions Rate Between UAVs: f_{UAV}

Following the discussion, a way to model the frequency of impact between two UAVs in the same area is needed. Please notice that we are still talking about risk prevision, this means that we are not going to provide here algorithms for the collision avoidance: this work has been developed by other members of the team, and consists in a collection of real time procedure implemented in order to allows two drones whose trajectories are going to collide, to avoid each other. In the risk analysis work instead, we want to model the frequency of collision that a drone will have flying on a precise area, knowing a priori some interesting information. Practically, one approach is used a priori to provide a risk value for the path planner. The other instead is used in flight, and provide a series of way point for the drone to avoid others unmanned systems.
For what concerns the estimate of f_{UAV}, many different techniques have been developed in order to obtain a correct measure. The main problem in this case is to known all the trajectories of the drones over a defined area, instant by instant, in order to be able to evaluate the number of collisions.
According to [6] and [22] we can write:

$$f_{UAV} = E(CT) \times P(collision|CT) \tag{2.13}$$

Where:
$E(CT)$ is the expected value of conflicting trajectories in a given zone. Measure Unit: $\frac{trajectories}{h}$
$P(collision|CT)$ is the probability of having a collision, given two conflicting trajectories.
One of the most used techniques for the estimate of $E(CT)$ is the gas model [6] [22], where a volume of airspace V around the drone is considered. In this model, every other aircraft that can impact with the UAV is considered starting from its exposed area: then, the volume that it will occupy during its trajectory is evaluated. In formula:

$$E(CT) = \frac{A_{exp} \times d}{V \times t} \tag{2.14}$$

Where:
A_{exp} is the exposed area of the drone. Measure Unit: m
d is the distance that the aircraft will cover in the volume V. Measure Unit: m
V is volume of airspace around the UAV. Measure Unit: m^3
t is the time that the other aircraft will spend in the volume V. Measure unit: h
A practical way to understand this equation is to visualize it, as in image 2.4. Unfortunately, the use of this kind of model is almost always ineffective [6] since it requires the knowledge of the exact trajectory of all the other UAVs flying in the same area. This turns out to be absurd for two reason:

1. Even if a single Traffic Manager was used to plan all the trajectories (so a completely centralized approach) for the drones flying in our scenario, it still cannot forecast all the changes in the path that can happen in real-time, due to flight control systems or simply for external unexpected events.

2. The Risk-Aware Map Manager here proposed aims to quantify the risk for an area, providing different values of risk according to its resolution, then finding the path that minimizes it. This conflicts with the gas model approach, because in order to evaluate the risk for the drone to fly in an given area it requires to know when it will be there, but this information will become known just after the risk analysis.

That's said,it seems we have to find a new method to evaluate $E(CT)$. An interesting approach, that is both simple and conservative, is a "statistical worst case analysis": knowing how busy

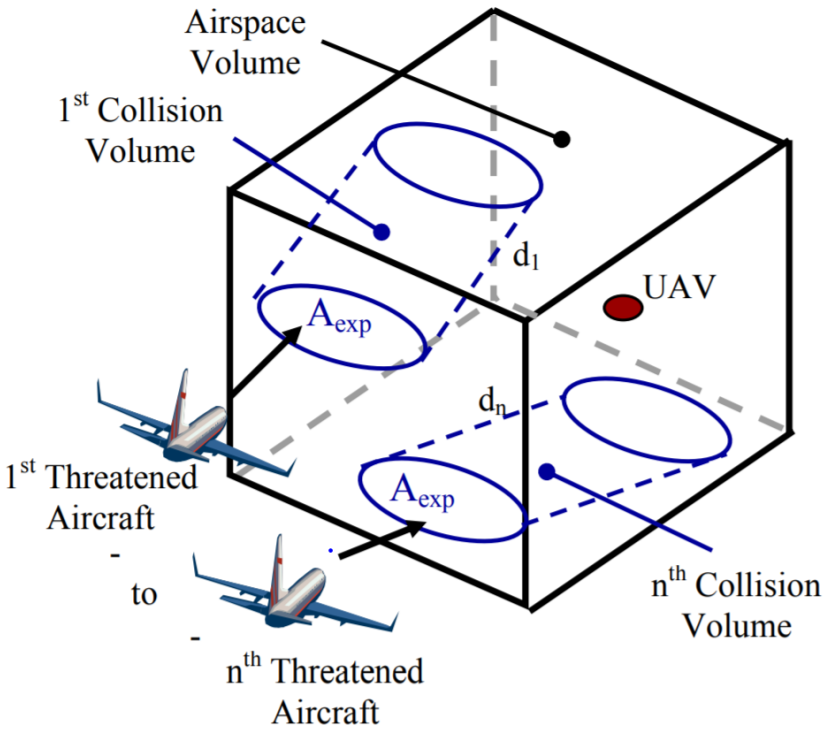


Figure 2.4: Gas model application

is a given area in the time window in which the mission take place, it's possible to provide a coherent value for $E(CT)$. According to simulations([6]), a typical value for high busy areas is $E(CT) = 4 \times 10^{-5} \frac{CT}{h}$. Obviously, this value greatly changes in function of the dimensions both of the considered airspace and of the aircraft. This means, as for other elements already seen of this risk-aware framework, that only a huge number of simulations (maybe a Montecarlo ones) can provide a coherent value for $E(CT)$. Finally, we can conclude the mid-air collision risk analysis with the study of the term $P(collision|CT)$, that was originally introduced to take into account the possibility that one ore both the conflicting UAVs implement on board collision avoidance algorithms [6]. Practically in our scenario we cannot suppose to know if and what algorithms are implemented on other drones, so this analysis should focus only on the UAV which is going to perform the mission whose risk we are evaluating.
In our opinion, two different methods can be implemented to estimate $P(collision|CT)$:

1. Analyze the algorithms implemented in the UAV in order to quantify its real ability in avoiding collision and provide a number.

2. Simply assuming $P(collision|CT) = 0$ when the anti collision systems are implemented and $P(collision|CT) = 1$ otherwise.

Today, the state of the art in collisions avoidance algorithm guarantees optimal performances, also in complex scenarios as the one we are working with. Simulations also proved their capability (chapter 5): thanks to this, and to the cloud-based traffic manager which implements the algorithms

(and the connection with the drones), we can actually easily suppose $P(collision|CT) = 0$ for every flight.

2.4 Ground Impact Risk Modelling: A Complete Framework

According to the first part of this work, to properly evaluate the risk we studied both the possible accidents' situation: lost of control during the flight, due to a failure of the UAV, and mid-air collisions. Each of this hazards can bring to a ground crash, which is the real danger for the people. The results we obtained, already exposed, relays on a simple metric: to every area, depending on its peculiar characteristics and on map resolution, the risk assessment procedure is able to associate two number (one for each type of accident) that coherently represents the risk.
The last step to complete the risk assessment problem consists in unifying the two metrics previously introduced, in order to provide just one value to describe the whole ground impact risk. This process seems to be quite simple, starting from this assumption: in every area, just one of the two kinds of accidents can happen. This practically means that the ground impact can be caused or by an in-flight lost of control or by a flight's collision, not from both of them. Since we are building a conservative risk analysis, between the two risk values the framework must choice the bigger one. In formula:

$$f_F = \max(f_{F,EFT}, f_{F,MAC}) \tag{2.15}$$

That can be developed as:

$$f_F = \max(N_{exp} \times P(fatality|exposure) \times F_{EFT}, N_{exp} \times P(fatality|exposure) \times f_{MAC}) \tag{2.16}$$

And finally becomes:

$$f_F = N_{exp} \times P(fatality|exposure) \times \max(f_{EFT}, f_{MAC}) \tag{2.17}$$

Because the first two elements are the same for both the accident's situations.
Equation 2.17 can be addressed as the final ground impact risk formula, since it contains all the parameters of interest, in both the crashes' scenarios. For what concerns our operative environment however, some considerations must be done, that strictly follow all the risk's discussion.
Beside N_{exp} and $P(fatality|exposure)$, we have also to deal with f_{EFT} and f_{MAC}, which are the only two parameters that differentiate (and this makes sense) one hazard from the other. According to what has been said, let's resume them in few words:

f_{EFT} it's the frequency of ground impact due to an internal failure of the aircraft, or in general to everything that can causes an unexpected end of the flight in a zone doesn't equipped for landing. Unfortunately (for the traffic manager, that have to manage risk), for small and cheap drones as the ones that fly over cities this kind of event is "quite" common. In particular, according to [17] the order of magnitude is around $f_{EFT} = \frac{1}{100}\frac{1}{h}$.

f_{MAC} it's the frequency of ground impact due to an in-flight collision between two drones, or more in general between the aircraft we are considering and any kind of static or dynamic object. As discussed in section 2.3, main components of this term are unexpected obstacles on the trajectory (static elements) or other aircrafts(dynamic). For what concerns the first, we can assume (and will be explained in chapter 3) that the Risk-Aware Map Manager will take into account almost all the possible criticisms, and only a few of them, due to misleading information, could produce real risky situation. Moreover, thanks to the collective learning of cloud systems, the RAMM will improve over time, finally reducing to very low values both f_b and f_{TBE}.
On the other side, we have seen that collisions between drones, which are theoretically the

most affecting component of f_{MAC}, can be assumed to be null (or tending to) if we implement (as we did) on the Traffic Manager proper collisions avoidance algorithms. That's said, and according with section 2.3, the order of magnitude of f_{MAC} in a generic instant is around $10^{-7}\frac{1}{h}$.

For this reason, since is always $f_{EFT} \gg f_{MAC}$, equation 2.17 can be simplified omitting f_{MAC}:

$$f_F = N_{exp} \times P(fatality|exposure) \times f_{EFT} \tag{2.18}$$

Starting from the latter equation, we are now able to provide an operative and quantitative definition of risk, as victims per hour of flight, that respects all the common standards of the most important national flight's agency. It's interesting notice that, since the risk is defined as a frequency it varies in the range $(0, +\inf)$.
With this theoretical basis, the Cloud-Based UASs Traffic Manager is capable of developing a a complete risk assessment procedure. However, we can easily imagine that each mission will be performed through areas with different risk, and the path planner has to guarantee the safety for the whole mission. In detail, as exposed in the first section, its possible to consider safe all that mission that provides a number of victims per hour less then $f_{F,Max} = 10^{-5}\frac{1}{h}$.
How to move from the risk definition based on areas to the risk for the complete flight's of the drone will be exposed in the next chapter, after the formal definition of risk map and of the Cloud-Based UASs Traffic Manager architecture.

Chapter 3

Cloud Based UASs Traffic Management

3.1 Unmanned Traffic Manager: State of the Art

An Unmanned Traffic Manager (UTM) is a "traffic management" ecosystem for uncontrolled operations that have to be complementary to the classical aviation ATM. Main objective of such system is to identify services, roles/responsibilities, information architecture, data exchange protocols, software functions, infrastructure, and performance requirements for enabling the management of low-altitude UAS operations [54].
Actually, different federal agencies (in USA) or international research teams (in Europe), both with the support of the sector's industry, are working to define standards, metrics and functions of a global UTM (for civil operation but not only). Thanks to the Unmanned Traffic Manager, a co-operative interaction between drones' operator and the National Flight Authority can be established, in order to have a real-time (or near-real time) organization of the airspace and finally increase the safety for people on ground. Furthermore, the development of new cloud technologies will allow to remove the man (in this case the UAV's pilot) from the control loop: the primary means of communication and coordination between the UTM and the aircraft will be through a distributed network of highly automated systems via application programming interfaces (API), and not between pilots and air traffic controllers via voice [54].
According to the International Civil Aviation Organization (ICAO), an UTM framework will include many components, three of which are crucial:

1. Registration System to allow remote identification and tracking of each UAS

2. Communications Systems

3. Geofencing-like system

In the rest of this work, how we addressed this features will be treated in detail, focusing, for what concern this thesis, on risk assessment and mitigation.
In order to build a completely new traffic manager (that in our case can be called Cloud-Based UTM), the study of the most important already existing ones has been fundamental. In this section, a brief review of two different UTM will be performed.
NASA's UTM project started in 2015, thanks to a partnership between the National Aeronautics and Space Administration and the Federal Aviation Administration (FAA). The development of

the work is coordinated by a Research Transition Team (RTT), which maintain in contact the two agencies and their industrial supporters [54]. Quoting [55]:

> The UTM system would enable safe and efficient low-altitude airspace operations by providing services such as airspace design, corridors, dynamic geofencing, severe weather and wind avoidance, congestion management, terrain avoidance, route planning and re-routing, separation management, sequencing and spacing, and contingency management.
>
> One of the attributes of the UTM system is that it would not require human operators to monitor every vehicle continuously. The system could provide to human managers the data to make strategic decisions related to initiation, continuation, and termination of airspace operations. This approach would ensure that only authenticated UAS could operate in the airspace. In its most mature form, the UTM system could be developed using autonomicity characteristics that include self-configuration, self-optimization and self-protection. The self-configuration aspect could determine whether the operations should continue given the current and/or predicted wind/weather conditions.
>
> NASA envisions concepts for two types of possible UTM systems. The first type would be a Portable UTM system, which would move from between geographical areas and support operations such as precision agriculture and disaster relief. The second type of system would be a Persistent UTM system, which would support low-altitude operations and provide continuous coverage for a geographical area. Either system would require persistent communication, navigation, and surveillance (CNS) coverage to track, ensure, and monitor conformance.

Crucial points that emerge from this description are the autonomy of the system from humans, its capability of determining if a mission has the safety condition to continue and its real time awareness of the geographical, atmospheric and congestion condition of the airspace.
NASA tests its progress through a series of experiments (of increasing difficulty) called "Technology Capability Levels (TCL)". The next one, UTM TCL3 is scheduled for Spring 2018 and will be the second-last before the ending of the project.
The second and last UTM to present is the so called U-Space. It is born inside SESAR (Single European Sky ATM Research), a plane wanted by the European Commission and leaded by a public-private partnership (SESAR Joint Undertaking) whose aim is to overhaul the european airspace and its traffic management. Differently from its US analogue, in this case the project is more focused on the problem of ensuring safety in a urban context, also without having a negative impact on the natural environment [56]. Quoting [57]:

> U-space is a set of new services relying on a high level of digitalisation and automation of functions and specific procedures designed to support safe, efficient and secure access to airspace for large numbers of drones. As such, U-space is an enabling framework designed to facilitate any kind of routine mission, in all classes of airspace and all types of environment - even the most congested - while addressing an appropriate interface with manned aviation and air traffic control. The U-space blueprint proposes the implementation of 4 sets of services to support the EU aviation strategy and regulatory framework on drones:
>
> U1: U-space foundation services covering e-registration, e-identification and geofencing.
> U2: U-space initial services for drone operations management, including flight planning, flight approval, tracking, and interfacing with conventional air traffic control.
> U3: U-space advanced services supporting more complex operations in dense areas such as assistance for conflict detection and automated detect and avoid functionalities.

U4: U-space full services, offering very high levels of automation, connectivity and digitalisation for both the drone and the U-space system.

Although quite similar to NASA UTM, in U-space are very interesting its focus on dense populated areas and the division of the work in four services. Such kind of structure has also inspired a part of our innovative framework, and will be exposed in next section.
Moreover, the European Aviation Safety Agency is working to propose a common EU-wide set of safety rule that have to be proportionate to the risk of the operation: this introduces the need of a risk assessment procedure (and also a metric!) as the one we described in chapter 2.
In general, from both the cases of study, some features emerged as crucial for an efficient Unmanned Traffic Manager:

- Registration and Identification
- Model of the Environment
- Risk Assessment
- Path Planning
- Collision Avoidance
- Real-Time Re-Planning
- High Level of Automation and Connectivity

All of them have been addressed by our research team through the new possibilities offered by cloud robotics, creating a framework that we called Cloud-Based Traffic Manager (CBUTM).

3.2 Cloud Robotics

The term cloud robotics was first used in 2010 by James Kuffner, an American professor and researcher who has a great impact in the robotic field. According to him, the cloud robotics is a new paradigm in which robots and automatic systems exchanges information and execute computations using a common on line network. The main benefits of such a system are ([24] [4]):

Big Data Access to an almost infinite database. Practically, the cloud offers a huge library of images, maps and data that every robot can access.

Cloud Computing A big computational power, that allows the engineers to design and use also very demanding (and complex) algorithms in their robots. This massively parallel computation is used for example in on demand path planning, motion planning and multi-robot collaboration.

Collective Learning The robots connected to the cloud can exchange information, trajectories, ecc. The knowledge can be instantly moved from one element of the network to all the other systems connected to it (or just to the interested ones). Furthermore, humans can share open source code and data, augmenting human-robot interaction.

Nowadays, cloud robotics is being deeply used in many applications, as for example autonomous mobile robots, medical and domestic robots and industrial systems. It' s importance is growing up day by day since it perfectly merges with two other advanced technologies, machine learning and internet of things. After this presentation, it becomes evident the reason of the choice of a cloud-based framework to realize the controller described in the previous section. On one hand, there is

the need for everyone who wants to quantify the risk in real time to exchange data, information and geographical maps. In our idea, each UAV connected can transmit\receive to\from the cloud, updating the common shared information if its sensor perceive something different from what the cloud expected.
This will provide an always updated and common database for the risk analysis.
Finally, the study of many path planning and collision avoidance algorithms has shown the need of a very high computational power, that only a cloud framework can provides. For this reasons, the ideas and results that will be presented in the rest of this work have been developed supposing to use a cloud network. Thanks to the Joint Open Lab's resources, we were able to prove and verify the results on a real cloud simulator (see chapter 5).
Please notice that for the first development of this framework, the cloud is supposed to be ideal: practically, this means no latency or package loss during the drones' flight. This assumption is very strong, since the wireless connectivity to the network is one of the most important issues of the cloud systems. According to [5] Cloud-based applications can get slow or unavailable due to high-latency responses or network hitch. If a robot relies too much on the cloud, a fault in the network could leave it "brainless". Since we are working in a very risky environment, a fault in the cloud can cause people's death, that is obviously unacceptable.
Future works are going to integrate in the framework new algorithms for the managing of this kind of situations on the on-board controller presents on each UAV.

3.3 Cloud-Based UASs Traffic Management: General Architecture and Assumptions

Cloud Based Traffic Management (CUBTM) is an air traffic management solution based on the Cloud Robotics paradigm and designed for small commercials UAS systems. Its final aim is the creation of a structured low-altitude airspace which allows drones operations also in urban environments while preserving a certain level of safety for humans.
Starting from the actual state of the art in the field of ATM, CBUTM expanded this concepts in a completely new way, finally removing the men from the flight's loop and ensuring safety along the whole mission. The logical structure we developed is the described in figure 3.1, where each block has its own purpose, crucial for the optimal work of the others. The flow chart here depicted shows both the user's and the authority's point of view.
Basing on what seen in the previous chapter, in this one we show the general architecture of CBUTM and the strategies we adopted in order to accomplish our objectives.

The idea is to provide a central gateway to the unmanned missions, both for customers and authorities. One key characteristic of CBUTM is that it work on a standardized risk assessment, and so is able to operate assuring the same safety level for each possible mission in the city. The authorities have at the same time a unified access to monitor and to intervene on all those parameters that condition the mission requirements and so the characteristics these missions will have.

In our opinion this feature would be of interest particularly in the short term, accelerating the ongoing tests and so the adoption on large scale as well as it accelerates the velocity of the feedback loop for control agencies.

But also changing parameters, the metric could be maintained. This in turn would allow different instances of CBUTM, each one under the control of the respective authorities, and at the same time a common language (the metric itself) among them. In this way, for example, different cities could use different requirements basing on the characteristics of the local environment, while continuing be inside the parameters' range imposed by the national authorities. The same mechanism can be applied even on a larger scale, like among different states of the European Union.

On the other side, a unified portal to access to unmanned missions and to the relative services is offered to the citizenry as well as to private company that would receive huge benefits operating in this field.

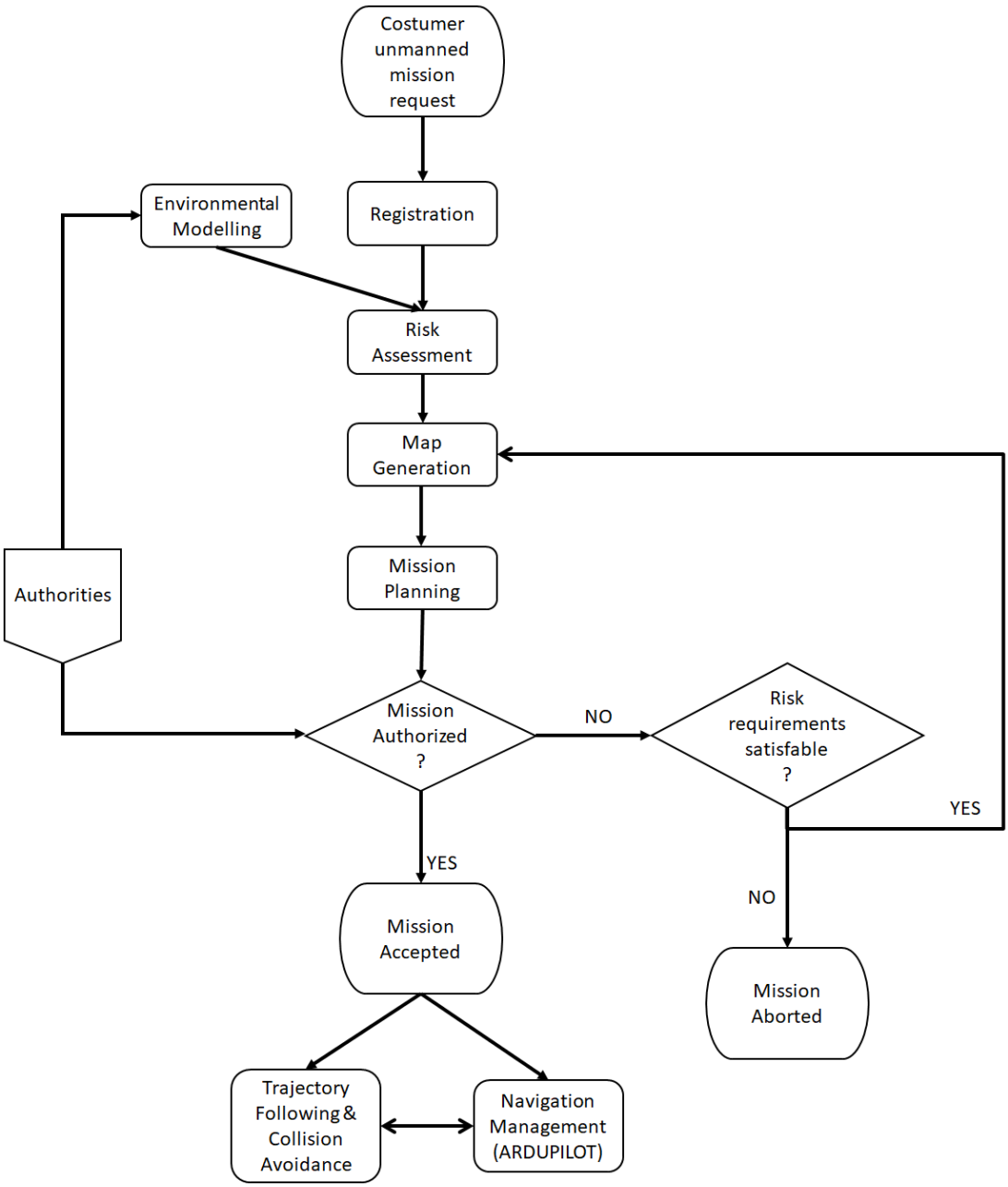


Figure 3.1: Cloud Based UASs Unmanned Traffic Manager: Architecture

In order to understand the overall way of function of CBUTM, it can be useful to have a better idea of what happen when a mission's request is submitted to the system. For this purpose, figure 3.1 is the optimal reference to consider. Please notice that what we are going to do here it's just an

introductory overview of the working principles of our Traffic Manager: the detailed explanations of each blocks will be treated in a specific section.
As said before, if a drone wants to fly in the city's airspace it must interact before with the Traffic Manager, in order to join correctly the CBUTM. The first step it has to accomplish is the "Registration". In this phase, the UAV communicates all its structural parameters, goals and requirements it must fulfil to successfully complete its mission. The cloud system, from its side, assign to the drone a coherent level of priority, according to its need, and a "virtual identity" that will be the drone image in the CBUTM. Then, it's possible to start the procedure to identify if the requirements of the Unmanned mission are compliant with the safety standard of the Authority that manages the traffic manager. From here till the moment in which it will receive the allowance to fly, the drone has only to wait, since all the procedure will be performed transparently by the cloud itself.
Beside the "Registration" process, there is the Environmental Modelling. This phase is not triggered by the mission's request but instead happens at a constant rate till the CBUTM stays online: it must have, at any time, an update and coherent geographical map of the urban environment where the flights are performed. Although not available for now, it is reasonable to suppose that in future municipal offices will share their data on the city structure. Furthermore, this information are going to be merged by the "Environmental Modelling" block with the ones received in real time by the flying drones: finally, the result is the environmental map of the city we were looking for. Differently from the rest of the flow chart, this part of the system is independent from the drone that has advanced the request, and remains the same for ever aircraft joining CBUTM.
Once the environmental map is available, and the registration process has began, it is mandatory to start the "Risk Assessment" for the specific UAV. Practically, this means to apply the Risk Modelling techniques (already seen in previous chapter) to build a point by point map of the risk that associates at each area of the geographical map its corresponding risk value. How to do this in detail will be described later on in this chapter.
The risk map, function both of the geography and the UAS's constructive parameters, is going to be managed by the "Map Generation" block, which consider it just as a layer of a more complex cost map, built weighting all the other requirements of the unmanned mission. As output, it provides a cost map, this time function of geography, drone's parameter and mission standard objectives. "Environmental Modelling", "Risk Assessment" and "Map Generation" together form the Map Manager structure.
The map coming from "Map Generation" becomes the input of the "Mission Planner", whose final aim it's dual: on one side, it must find the best (lowest cost) path for the drone to follow, on the other it must evaluate if this trajectory is compliant with the standard (for example of safety) imposed by the National Authority. This two different souls, called "Path Planner" and "Path Validator" co-operate in this block, since the output of the first is the input of the latter.
At this point, three different situation are possible. In the first, the mission's requirements are impossible to be satisfied: the only solution in this case is to abort the mission, that's unfeasible. The second possible way instead happens when the choosen trajectory doesn't satisfy the requirements, but it's still possible to change the weights of the cost map and re-calculate another path. Finally, the last scenario is when the trajectory accomplish the mission's request and the safety bound, so that the drone can be authorized to fly.
In this case, a controller of the in-flight operations is needed: inside CBUTM, the two blocks "Trajectory Following & Collision Avoidance" (TFCA) and "Navigation Management" are implemented for this specific purpose.
The first, as the name says, is responsible for the high-level navigation control. In brief it is a system executed alongside every mission and it's responsible to continuously translate a path (the output of the mission planner) into a proper trajectory (a path plus a time law). The generated

trajectory has to be equivalent to the received path as long as this doesn't imply that the distance from other vehicles, that clearly depend on time, goes below a safety threshold. In that case, TFCA has to produce a trajectory different as much as necessary to avoid this circumstance. TFCA works in parallel with the Navigation Management, to which it sends instruction and from which it receives a feedback on the actual state.
Navigation Management system, which is the only one to be on-board, translates high-level instructions into low-level ones, interacting directly with electromechanical actuators and with other devices onboard, like sensors.
That's said, we finally conclude the introduction to the Cloud-Based UASs Traffic Manager. In the rest of this chapter, some of the concepts and blocks just presented will be extended, according to the logical purpose of this thesis. In particular, we will focus on:

- The construction of a Cloud-Based Risk-Aware Map Manager, which will include Environmental Modelling, Risk Assessment and Map Generation blocks.
- Path Planning Algorithms (Mission Planning).
- Validation of the UAV's trajectory, defining a Path Validator system (Mission Planning).

In this way, this thesis contribution aims to cover all the features of an UTM, from the moment in which the registration process ends till the beginning of the mission.
Finally, one structural comment: though the Map Managing and the Mission Planning are logically two different parts of the traffic manager, they couldn't be treated separately, since they both share the "risk-aware" characteristics and, furthermore, the latter works upon the output of the first. For this reasons, they will be both discussed and constructed in this work.
An first explanatory flow-chart of the overall process is depicted in figure 3.2.

3.4 Map Manager

3.4.1 Introduction and Risk Map

All the theoretical background exposed in the previous chapter aims to provide the instruments to quantify the risk, measured as the number of victims per flight hour, of an unmanned mission. As already discussed, the risk is both function of space, since it changes according to the area we are considering, and of time, because we supposed to be able to handle useful information for the mission (as population density for example) in different ways according to the time in which the flight is performed. Finally, the risk also depends on the building parameters of the UAV itself, as mass, maximum speed and collision avoidance algorithm implemented on it.
Starting from this concepts, the first step the Traffic Manager has perform to ensure safety is the creation of a risk map for the drone's flight: this map has to be built on the real geographical map where the mission takes place, but it has to contain in every point the corresponding risk value for people on ground when an UAV overflights that area. The resulting map will be analyzed by the path planner in order to find the best trajectory for the drone, among al the possible path, between the starting point A and the goal B.
The core of this chapter will be to explain how and why we decided to build this map. Moreover, we will pass from the definition of risk of an area to the overall mission's risk, providing all the means to pass from the theoretical part (chapter 2) to the implementation part (chapter 4). Finally, also the Path Planner and the Path Validator will be treated.

Definition 5. The mission's risk map for an Unmanned Aerial Vehicle(UAV) is the map that for every point of the geographical area were the mission takes place associates the corresponding value of f_F, function of time, space and building parameters of the aircraft.

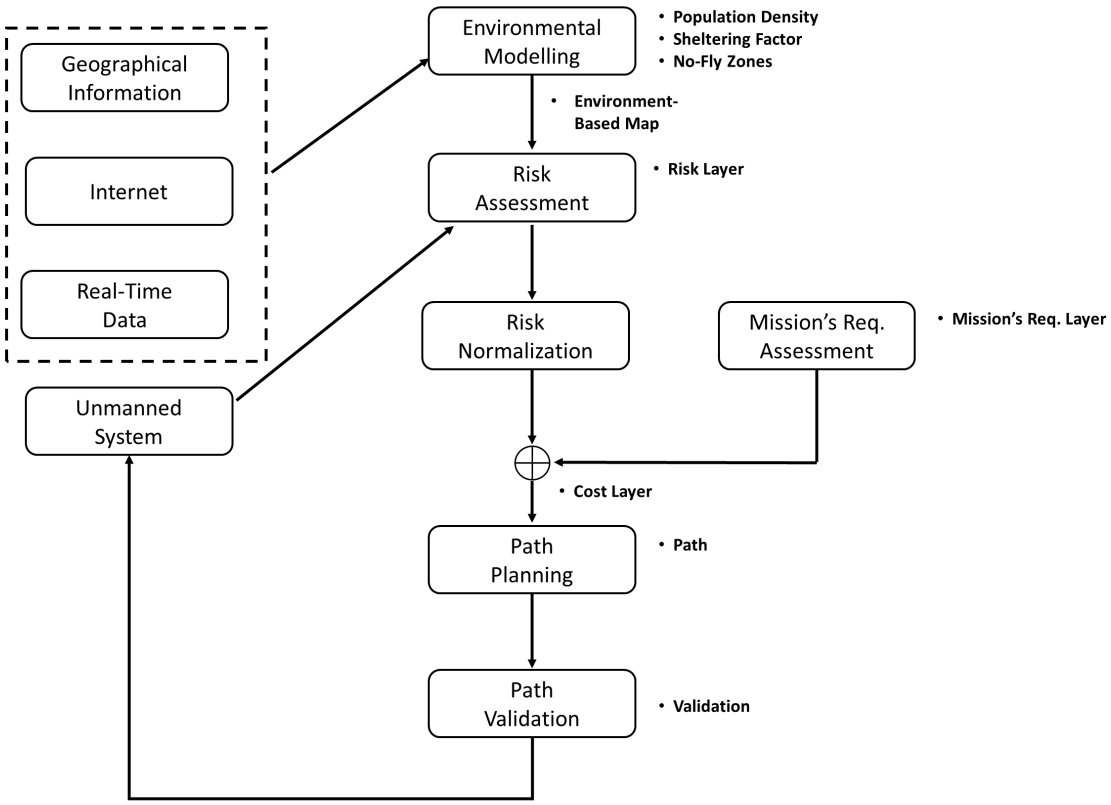


Figure 3.2: Risk-map construction flow-chart

f_F will be evaluated with the risk assessment techniques exposed in chapter 2.
For a practical reason, though we are talking of punctual values, the risk map will not be really punctual, but instead divided in sub-areas that will be considered uniform from a risk analysis' point of view. This choice has two reasons:

1. Building a "point resolution" risk map requires a really high computational capacity. Although we could handle this effort thanks to the cloud system, it seems however useless: no one of the information required to evaluate the risk are defined in a point, but instead have to be considered in areas.

2. Till now we have not consider the UAV's dynamic. However, since the risk map will be used by the path planner to provide a trajectory for the drone, it must be feasible for the UAV: a punctual path planner will be useless because it will not guarantee the feasibility of the flight.

Basing on this idea, we decided to build a risk map with a constant resolution of $25m^2$: the result is a square grid were each sub area has its proper risk value. The path planner will provide as coordinates for the UAV the coordinates of the center of the square. Please notice that the dimensioning of the areas has not been causal, but it has been decided according to the works and studies of prof. Guglieri on the typical flight characteristic of an urban drone. Practically, the $25m^2$ squares guarantee the manoeuvrability of the UAV. However the map resolution can

be easily modified during the implementation part, to better fit the standard requested by the mission's manger.
The idea of constructing a map of the risk was made possible (and more effective) by a particular feature of "grid map", a C++ library which we decided to use in the implementation phase: the layered structure. Without entering in detail that will be treated in next chapters, let's show the potentiality of this kind of environment.
According to definition 5, the risk map is a 2D plane, divided in cell, where each of them contains the risk value $f_{F,i}$ of the corresponding geographical area. So, for each zone the cloud framework will calculate the corresponding risk evaluated at the flight's height h with equation 2.18, so it has to know and handle:

1. Population density, varying in time
2. Mass of the drone
3. UAV's flight height
4. Sheltering Factor of the area
5. Frequency of ground impact due to an early flight termination

Furthermore, the cloud have to know the number of no-fly zone on the map and where they are located.
Each of the above information can be seen as a map itself, since as the risk it is associated to each and all the cells of the map. The results is a layered structure, where each layer of the map is a map itself, containing the geographical distribution of one of the parameter of interest for risk assessment: the risk map is just the last and most important of the layer, obtained working on data contained in the layer below.
Knowing all this information, we can collect them using formula 2.18 and find out the risk f_F for each sub-area of the map. The result of this operation, graphically evident in figure 3.3, is the risk map for a particular drone, at a given time and flight's height. At this point, one important assumption has been done by our work group. Since we worked on this project from scratch, in order to simplify the analysis and to reduce the computational power required, we supposed to have a planar flight: the Unmanned Vehicle has its height of flight on the z-axis, but it cannot modifies it. Practically, this means that it is possible to consider it as a planar robot moving on the xy plane. Please notice that for different heights the risk manager will provide different risk value: for example, a drone flying $10m$ from the ground will encounter many more buildings than one flying at $50m$, but will have a lower impact kinetic energy. From an implementation point of view (detail in following chapters) this means to cut the geographical map at the drone's height and build the risk upon it.
The importance of this results must be stressed, since the first step to have a safe (and acceptable from a legislative point of view) fly is to have a detailed and coherent risk map. Working on it, the path planner will produce the safest path among all the possible ones, and with a particular metric (that will be introduced later on) it will be possible for the cloud traffic manager to compare it with the safety standard required by the national authority.

3.4.2 Environmental Modelling and No-Fly Zones

The first step to produce a coherent map of the risk of a given scenario, is to model the environment as accurately as possible, giving a particular attention to the buildings. As introduced when dealing of mid-air collisions frequency, in a urban context a possible cause of crash is obviously the

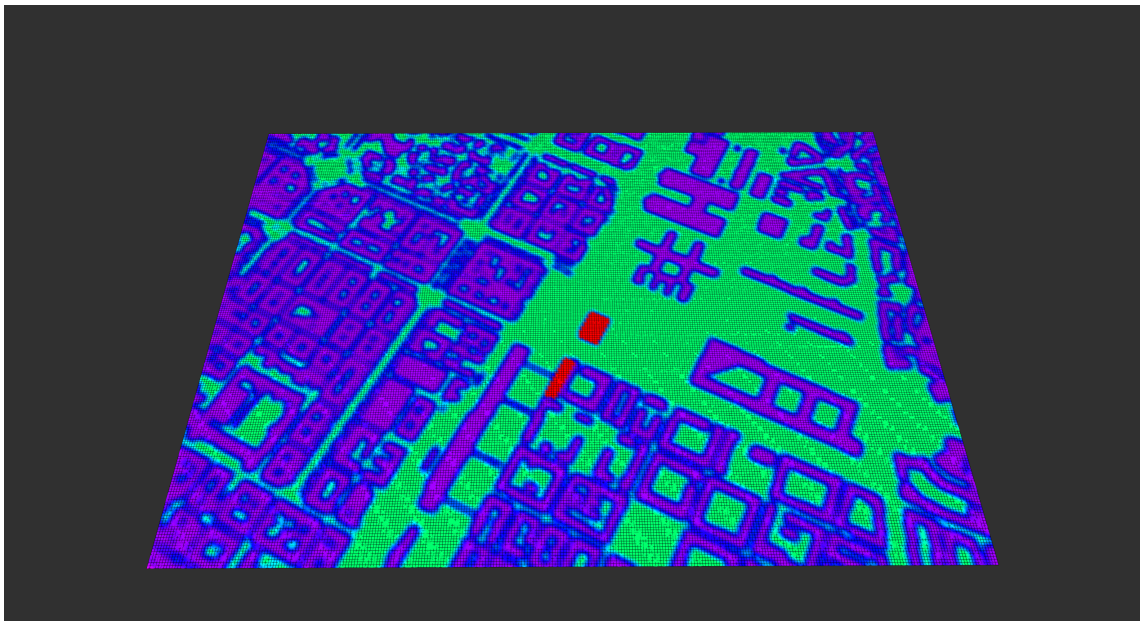


Figure 3.3: Example of Risk Map. Every area of the map has its own colour and height coherent with the risk value calculated. Red squares are no-fly zone

impact with palaces, that have to be managed by the Risk-Aware Map Manager.
3D models of the world are today available on web, but the most famous one is property of a single company, which doesn't allow the sharing of its data. For this reason, a set of data less accurate but open source has been used in CBUTM: Open Street Map (OSM). According to their own definition:" OpenStreetMap is a map freely modifiable of the whole world. It allows anyone to visualize, change and use geographical data with a collaborative approach". This means that volunteer updates the map, that are provided as .osm files. It's interesting to notice that thanks to them, it's possible to create a really effective Cloud-Based Traffic Manager that, once it will be on-line, will be capable of updating with its drone the maps received. This is a really collaborative approach.
However, OSM only provides 2D maps, useful for cars' path planning but less for drones, which are interested in buildings' height. To solve this problem, another open source software have to be used: OSM2World, which is "a converter that creates three-dimensional models of the world from OpenStreetMap data. It can be used as a stand-alone tool, on a server or as a library in Java programs."
Finally, 3D model of the city is obtained, which can be freely modified and used by the cloud. At this point, a cut on the map have to be done, to find if ad where there are buildings taller than the UAV's flight altitude: in this case, they will be treated as no-fly zone, which have a special metric exposed in the rest of this section.
Beside the pure structural information, also the distribution of population (density) and the sheltering factors information have to be collected as layer of the map, in order to be capable of evaluating risk with formula 2.18. How to derive sheltering capability from 3D models (or images) will be treated through neural network in another thesis works. In here, let's suppose to have already this information. Finally the risk map (or risk layer) will be built applying to every cell:

$$f_{F,i} = N_{i,exp} \times P(fatality|exposure)_i \times f_{EFT,i} \tag{3.1}$$

Where each element is about the i-th cell. What emerges during the building of the risk map, an this will be evident when looking at the simulation, are two different "risk situations" that the cloud system have to handle when evaluating the risk of a particular area. On one side there are all the map's zones whose risk can (and have to) be calculated with the classical formula now exposed. On the other side instead, there is a set of special areas that, for different reasons, must be managed by the cloud as no-fly zone with $f_F = \inf$. This implies that for such areas the standard equation must not be applied, while the risk manager have to recognize them and assign the proper value. This issue, that can seems simple, turned out to be quite tricky since it introduces exceptions in the regular risk assignment procedure
So, we must introduce a metric to handle and include them in the risk map. First of all, let's discuss what are the no-fly zones, and why we must manage them.

Definition 6. A no-fly zone or no-flight zone (NFZ), or air exclusion zone, is a territory or an area over which aircraft are not permitted to fly.

Historically, the no-fly zone were introduced in military context, acting like a demilitarized zone in the sky, and usually prohibit military aircraft from operating in the region. In the last years, however, with the growth of the civilian use of Unmanned Systems, this terms started to be used also in urban flight's environment. In particular, it is possible to distinguish two kind of no-fly zone:

Law-Imposed No-Fly Zone are all the areas where the drone cannot fly over due to some legislative restriction. For example, important sportive events (like 2012 Olympic Games in London) are very often considered as no-fly zone. This kind of no-fly zone must be handle by the cloud framework merging information available on internet and data manually added from an operator. Notice that this kind of NFZ still remain valid also varying the altitude of fly.

Structural No-Fly Zone in this category it is possible to include all the no-fly zone that have to be consider as that because there is some structural item (natural or artificial) that impede the flight. The typical example of such element is a building taller then the height of the UAV flight: it seems obvious that, if the drone will try to fly over the area occupied by the building, it will fall down. Notice that for this kind of NFZ, varying the altitude can varies the number of no-fly zone.

Practically, from a risk map point of view, this means an area with estimated fatality rate $f_F = \inf$. How to manage a no-fly zone has created many discussion inside the team that developed this project, because every different way has its pros and cons, but in the end we decided to follow the most conservative idea. In the following part, some of the methods we studied are exposed. In order to better understand them, let's consider just the no-fly zones generated by obstacles that can get in the way of the drone: practically we are working with an area which inside has one or more (it depends on its dimensions!) buildings taller then UAV's elevation.
The first approach we tried to implement was a probabilistic one that, starting from the covered area taller then the flight altitude, provided the number of accidents per hour that happened flying over that zone. In order to really evaluate this parameter, a Monte Carlo simulation has been implemented. Let's take into account that flying in a zone that contains one or more obstacles doesn't implies certainly a crash, since this event depends on the dimension of the buildings both respect to the area considered and the UAV length.
The main advantage of this approach is that it doesn't need to handle no-fly zone: to each area that has a tall building a frequency of collision can be evaluated, then it can be added to the formulation of f_{MAC} previously exposed. The result is that such area has a bigger risk value then

the other ones, and will naturally be avoided by the path planner. In this case, no special way to manage the no-fly zone has to be introduced.
However, according to the meeting at TIM's JOL Crab, we were convinced by our supervisors to avoid this statistical methods. There a few but very compelling reasons to do this. First of all, it soon emerged the uselessness of treating "probabilistically" known obstacles: trying to estimate the number of collisions that an UAV will have flying over a zone where we known there is a building taller then the drone's height seems useless. Simply, we must avoid that UAV will fly over it, and to do that we must introduce in the map, and consequently in the path planner, the concept of no-fly zone. Furthermore, implementing the Monte Carlo simulation we noticed that it is very hard to obtain coherent results, because they are strictly depending on the UAV's trajectory (for example a rectangular building is more likely to be hit by the UAV if it comes from the direction perpendicular to its long side) and other external parameters, as the wind, not known.
The most conservative approach proposed, that is also the one we decided to use, is the following: each sub-area, which we suppose is a $25m^2$ square, has its own risk evaluated with equation 2.18. However, if inside that area there is even only a little part of a building that is taller then the UAV's flight height, then all the area is considered as no-fly zone. This means that its risk value changes from it actual one and becomes inf. Practically, this means that all the area with buildings will be surely avoided by the path planner. The ony way in which there may be a collision is if the information on the buildings height and coordinates are wrong: this case however, is already taken into consideration by the f_b term in equation 2.11.
This approach has two interesting advantages:

1. We are still performing a conservative risk analysis. In the worst scenario, we marks as no-fly zone an area that contains just a small part of a structure. This is negative on one side, because it can brings to an over conservative approach, but on the other it seems good since it creates "naturally" a buffer safety area around the buildings.

2. It automatically prevents the path planner to chose trajectory that may involves collisions with buildings. Thinks at the following situation: two ways are available to go from point A to point B, the first uninhabited but with very high buildings while the second is normally populated with the typical $30m$ buildings. Since in the first case we have $N_{exp} = 0$, the risk will be null, so the path planner will choose the less risky trajectory, but it will cause almost surely a collision and a crash for the UAV. With the introduction of the no-fly zone instead, we prevent the path planner to consider trajectories that are practically unfeasible for the drone.

3. It allows us to handle also the "Law-Imposed" no-fly zone in the same way. For all the areas that are not geographical obstacles emerging from the map,indeed, a special way to mark them as no-fly zone was required, since no Monte-Carlo simulation could be performed (in this case it's not a matter of collision to avoid, but simply fly over prohibition).

In any case, the problem of creating and handling the no-fly zone has not a great impact on the real mission's scenario, since just a really small part of all the buildings in a city is taller then the flight's height, that usually is not less then $25m$. However, during the implementation part some issues occurred.
Practically, while implementing the software we simply cannot impose $f_F = \text{inf}$. At the end of this section, a way to provide an equivalent value for the no-fly zone will be presented.
The theory just exposed to evaluate the risk of a trajectory provides also the instruments to handle the no-fly zones. As said before, a no-fly zone has to be considered as an area with risk equal to infinite, but this is impossible to impose. Instead, we must find an equivalent finite number to define this "special" areas. The idea is the following: every path that contains a no-fly zone has to be considered unfeasible, and rejected by the mission acceptance routine.

This must be true despite all the other risk value of the path. Practically this means that though the UAV's path is zero risky except for the no-fly zone (this is a limit case) it has to be refused by the cloud framework. In figure 3.2 that extrem case is shown, where the grey areas have $f_F = 0$, the black area are the no-fly zones and the red square are the starting point and the goal. As

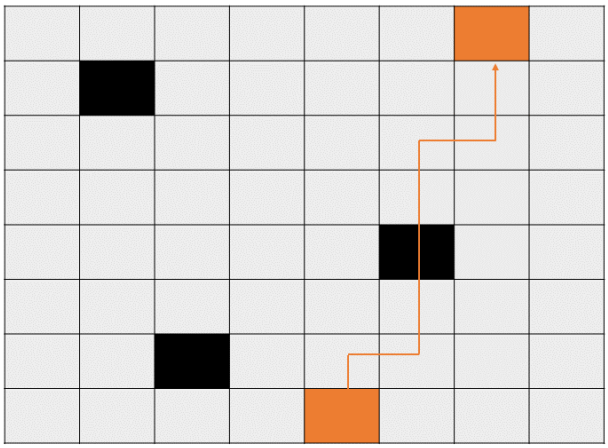


Figure 3.4: No flight zones' risk definition

we see, we are in a situation in which the path includes an area where the drone cannot fly. If it was possible to set $f_F = \text{inf}$, then we would not have problems since the planner would have automatically avoided this area itself. Instead, at a practical point of view, this is not possible. So, what is the value of f_F that insures this path will be avoided?
The first approach we developed was to impose:

$$N_{Traj} \geq N_{Max} \tag{3.2}$$

Where N_{Traj} is the number of estimated victims on a given trajectory, obtained multiplying each cell's risk frequency ($f_{F,i}$) by the corresponding time (τ_i)spent over it:

$$N_{Traj} = \sum_{i=1}^{n} f_{F,i} \times \tau_i = f_{F,NFZ} \times \tau_{NFZ} \tag{3.3}$$

Where:
$f_{F,NFZ}$ is the risk of the no-fly zone, to be defined.
τ_{NFZ} is the time spent by the drone over this area, already defined and constant.
So, we finally obtain:

$$N_{Traj} = f_{F,NFZ} \times \tau_{NFZ} = N_{Max} \rightarrow f_{F,NFZ} = \frac{N_{Max}}{\tau_{NFZ}} = \frac{f_{F,Max} \times \tau_{Miss}}{\tau_{NFZ}} \tag{3.4}$$

Where τ_{Miss} is the total time length of the mission.
In this case, however, some problems emerge, that seriously affects the coherency of this approach.

Let's enumerate them:

1. We don't know τ_{Miss} a priori, because it depends on the path decided by the path planner. Practically, different paths cause different τ_{Miss}, but the no fly zone has to be imposed in the map generation phase, before that the trajectory has been chosen. An idea to overcome this issue is to impose $\tau_{Miss} = \tau_{Aut}$, where τ_{Aut} is defined as the maximum flight autonomy of the Unmanned Aerial Vehicle, but however this doesn't really convince at all.

2. This approach allows the drone to fly over areas with local risk $f_{F,i} \geq f_{Max}$, since it's enough that the overall trajectory goes also in very low risky zone to have still $N_{Traj} = \sum_{i=1}^{n} N_i \geq N_{Max}$. This also implies that longer trajectories will be preferred in the validation phase.

For all this reasons, that method was discarded.
The idea we decided to implement is the simplest, but also the more effective. Practically, what we need to define a no-fly zone is an upper-bound for the risk: the NFZ has to be the more risky area, and both the path planner and the validation blocks has to recognize it as different from the other areas.
Supposing for example $f_{NFZ} = 10^{-5} \frac{people}{h}$, this value will be the greater for all the risk map, and has to be treated differently: in a very practical way, every path chosen that contains one ore more of this areas will be automatically discarded (see validation section for details). It's interesting to notice that having an upper-bound for the risk is also useful because it allows us to normalize the risk map, which values are in the range $[10^{-7}, 10^{-5}]$. Also this part will exploited later on. Following this idea, it seems obvious that the risk value to assign to a no-fly zone will be:

$$f_{F,Nfz} = f_{max} \tag{3.5}$$

So the upper-bound for the risk, that is equal to $f_{F,NFZ}$, is the one imposed by the legislation to consider a flight safe.
Finally, every area that, for any reasons, as to be consider a no-fly zone, will be marked with a risk $f_{F,i}$ equal to the maximum acceptable risk imposed by the National Flight Authority. The Cloud-Based UASs Traffic Manager, which knowns this value, can easily perform this operation: automatically for what concerns structural NFZ, or through an external input in case of law-imposed ones. Later in this chapter we will see how this "marker" is used by the Path Validator to find out not valid trajectories. Practically, with formula 3.5 we found a finite upper-bound for the risk.

3.4.3 Risk Map Normalization

Once the risk map is completed, it can be imagined as a huge matrix, where each cell contains a risk value in the range of $[10^{-7}, 10^{-5}]$ more or less.
The idea of normalizing the values of the risk map came out for two reason. The first is a matter of comfort: working with very low value can be stressing, especially when this little variation has to be recognized on the map through changes in colour or height. The second reason instead is more practical, and arises when we tried to unify the risk with other cost variables in a general cost function (see section 3.4.7): for this operation, a known range for the risk is need, and has to be equal to the one of the other elements.
According to the classical probabilistic definition, the normalization process is:

Definition 7. The normalization of ratings means adjusting values measured on different scales to a notionally common scale, often in the range [0,1].

According to this and to what exposed in previous sub-section, before developing the theory behind the no-fly zone the normalization process wasn't possible, since no upper-bound for the risk existed. Calling $x = \frac{f_i}{f_{Max}}$ the ratio between a generic risk value and it's upper-bound, this value is the simplest way to linearly normalize the risk between [0,1]. Furthermore, since we commonly decide to use variables for the cost function in range [0, 100], the first normalization technique used was:

$$R_i = \frac{f_i}{f_{Max}} \times 100 \tag{3.6}$$

Where R_i is the risk value of each cell. Please notice that after applying this operation, the risk becomes a pure number, without measure unit. This has no impact on the path planner, that can use the risk as a weight from a cell to another. When validating the trajectory,instead, the time spent on each area has an important role (as explained before): It will be easy at that point to convert back from R_i to f_i.
In figure 3.5 the results of this operation is shown. As expected, the result is a line from 0 to 100,

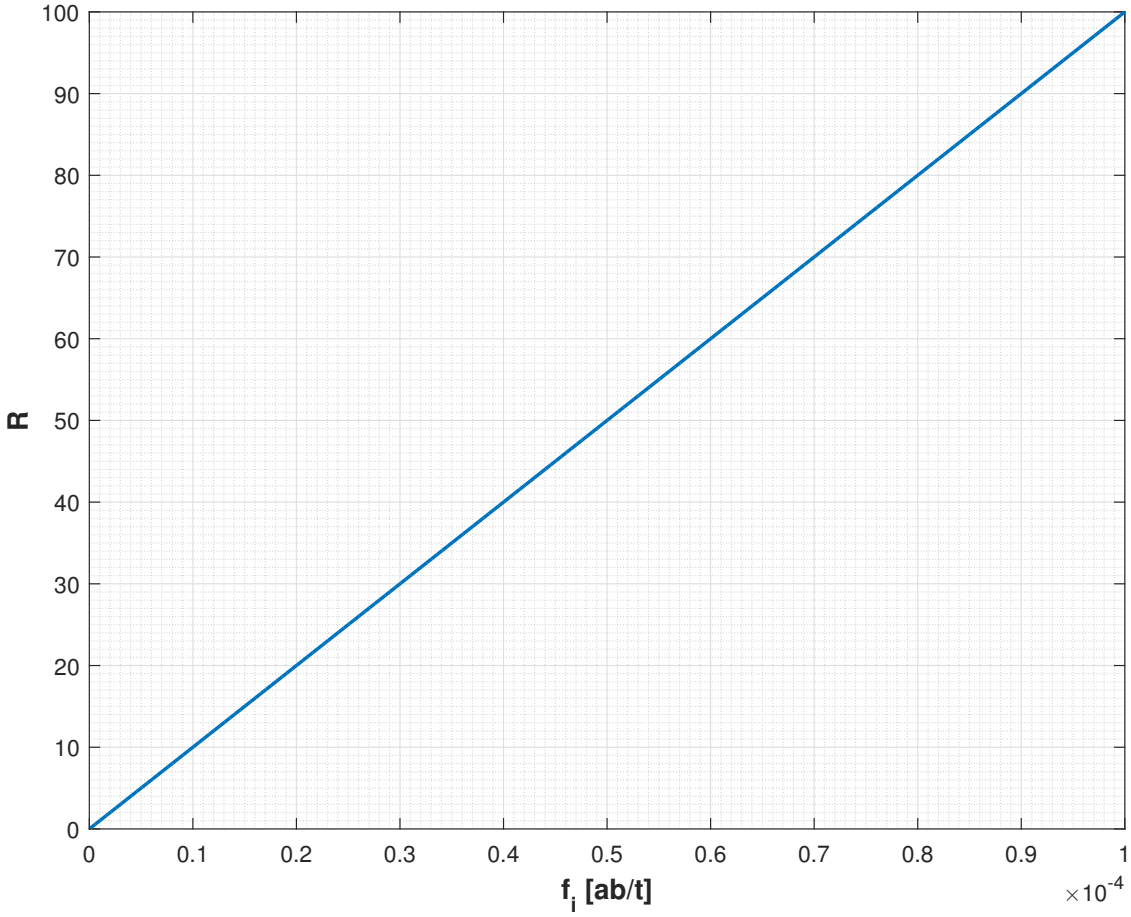


Figure 3.5: Linear Normalization of risk R respect to original risk value

while the x-axis move from a generic lower-bound for the risk (in this case 10^{-7}) to the upper-bound, that in this case its supposed to be 10^{-4}. Although the results seems good, when practically working with this value an issue occurs: real risk values, moving in the range [10^{-7} , 10^{-5}], are

still too small with this normalization. This is obviously due to the fact that the upper-bound is too bigger than them. In the next table some numerical result. From this examples, it emerges

f_i	R
10^{-7}	0.1
10^{-6}	1
10^{-5}	10
10^{-4}	100

Table 3.1: Risk normalizzation examples

that a great part of the normalized risk are mapped in the range[0.1,1], that it's quite useless: a different kind of normalization has to be found.
An interesting idea came out from the image processing techniques, and consists in using the gamma correction technique as a normalization method.
According to [32]:

Definition 8. Gamma correction, or often simply gamma, is the name of a non-linear operation used to encode and decode luminance or tristimulus values in video or still image systems. In the simplest case, the gamma correction operation is defined as: $V_{out} = A \times V_{in}^{\gamma}$ where V_{in} is the non-negative input and V_{out} the output, typically in the range [0,1].

In the most common case, it is imposed $A = 1$. For what concern the gamma parameter, it has to be chosen according to the need of each particular case: changing its value changes the shape and the curvature of the gamma function. When $\gamma < 1$ the non-linear operation is called gamma compression, otherwise gamma expansion.
In our case, we found the optimal formulation:

$$R_i = (\frac{f_i}{f_{Max}})^{\frac{1}{2}} \times 100 \tag{3.7}$$

So we are in the case of a gamma compression with gamma factor equal to 0.5.
The function plotted in figure 3.6 has a very huge shape for small number, while it becomes linear when approach the upper-bound. In this way, small number are no more mapped in range [0.1,1]. Let's have a look to some examples, in table 3.2. This normalizing approach seemed optimal to

f_i	R
10^{-7}	3.1
10^{-6}	10.3
10^{-5}	31
10^{-4}	100

Table 3.2: Risk Gamma correction

us, and it's the one we adopted in CBUTM. However, as everything in this framework, it can be changed according to the particular preferences of each user.

3.4.4 Risk Assessment: Analysis on UAVs Building Parameter

Now that we provided all the instrument to perform a correct risk assessment and create the corresponding risk map, it is time to perform some calculation to show how the risk varies in

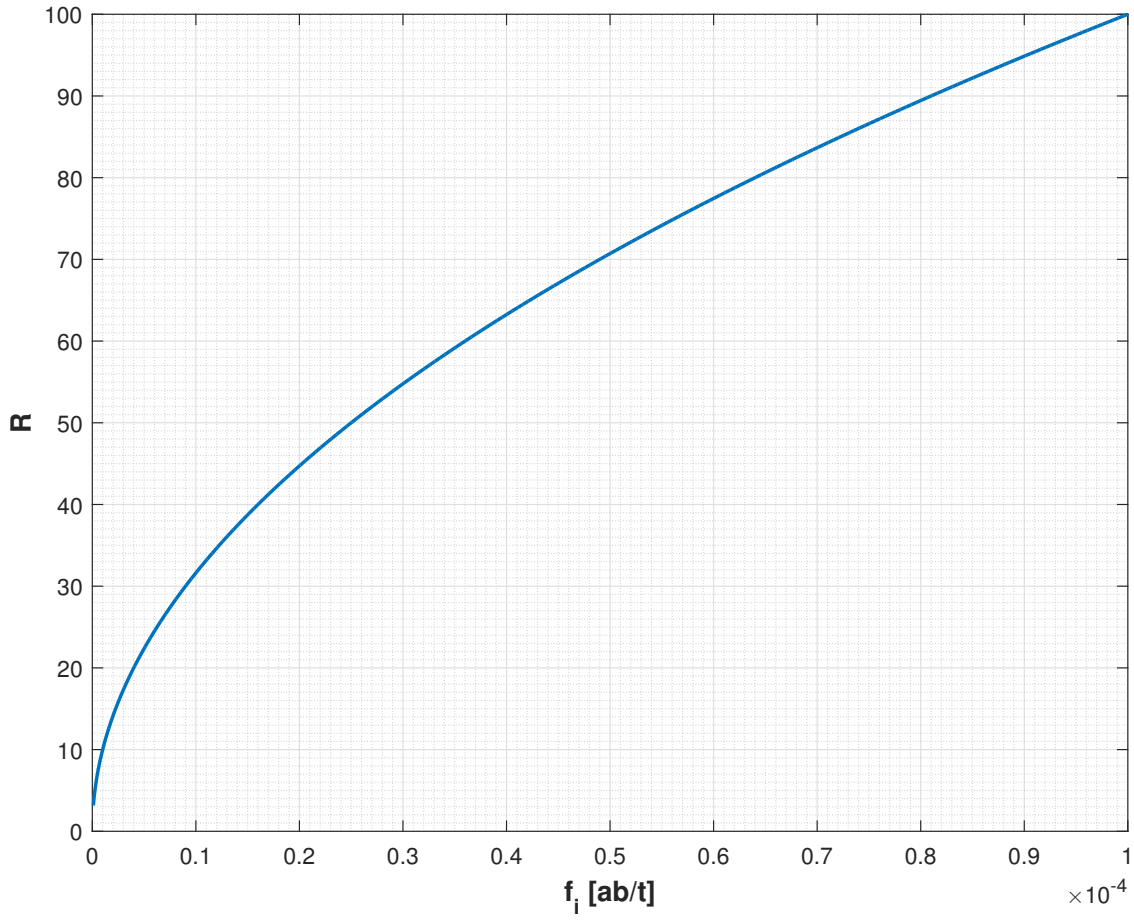


Figure 3.6: Gamma Correction of risk R respect to original risk value

function of the most important building parameter of an aircraft. In this sense, we will show the differences between quadcopter and fixed wing, increasing their length and masses. The final results will be a quantitative constructive bound, that industries should take into account during the design process of their drones.
Obviously, this results have not to be intended as a general rule, since their are just an example of risk assessment, based on assumptions (for example on the density of the area) that are not always true. What we are going to present has a dual effect, a practical one as example of risk evaluation, and a theoretical one as case of study for a generic urban scenario.
In order to better understand the rest of this section, it is really important to have a complete learning of the concepts and equations exposed in chapter 2, because for brevity reason it is impossible to explain again all the terms of each formula.
First, let's have a look to the main characteristics of the urban environment (in figure 3.7 a picture of a typical flight area). One of the main aspects that differentiate the cities from all the other flight situations is the massive presence of people, that for our framework means risk. In a city like Turin, an average value for the population density is $\rho = 6000 \frac{ab}{km^2}$ that must be converted in the proper measure unit: $\rho = 0.006 \frac{ab}{m^2}$. Since the example we are developing is static, we can assume ρ constant in time and equal almost in all the city: we are trying to provide bounds in nominal

Figure 3.7: Vision of a typical urban scenario

conditions, that in our case means always using the most conservative values.
In order to calculate the risk, its mandatory to build a grid, in which each cell will have his own risk value. As we said, a good dimension can be $A_{TOT} = 25m^2$, that is a quite big area for a small drone. Starting from this, it is possible to evaluate the second crucial geographical parameter: the sheltering factor. A previously discussed, to evaluate PS we have to know the shelter coefficient of each element in the map, and the area it covers. A rough but useful division is the following:
Half of the Area is covered by Concrete Buildings: $Cs = 1$.
A Quarter of the Area is covered by tall trees: $Cs = 0.5$.
The last quarter is empty, for example streets. $Cs = 0$.
Now we can apply the formula:

$$P_S = (\sum_{i=1}^{n} C_{S,i} \times \frac{A_{C,i}}{A_{tot}}) \times 10 = [(1 \times 0.5) + (0.5 \times 0.25)] \times 10 = 6.25 \tag{3.8}$$

The result is a typical averaged value of sheltering in a urban environment, considering the most common element that appears in the scenario and the mean area they occupy.
We end the geographical analysis underling that we are supposing to not fly over a no-fly zone: this is obvious, because such a zone will categorically deny any kind of risk assessment.
At this point, we can move to the Unmanned System analysis: since what we aim to achieve is the risk evolution in function of mass and length, this two parameters will remains variables and not specified yet. However, a couple of consideration on the drone have to be done. The UAV is supposed to fly at an height of 25m, considering as 0m the level of the street. This height guarantees both to be over a great part of the buildings, and not having a too big acceleration in case of fall (the higher is the drone, the harder is the crash). Moreover, we can assume that in case of failure, the drone will be in a free fall condition (no initial velocity along y-axe): practically, we are considering drone capable of recognizing internal failure and, if the situation appears critical,

able to stop their engine. In this case, the kinetic energy at the impact can be evaluated as:

$$E_{imp} = \frac{1}{2} \times MV^2 = M \times g \times h \tag{3.9}$$

Where M is the drone's mass, variable, and g the gravity's acceleration.
Since we calculated both the kinetic energy and the sheltering factor, it is now possible to evaluate the probability of having a fatality given a crash, in function of the mass of the drone. This interesting result is shown both in formula 3.10 and in figure 3.8.

$$P(fatality|exposure) = \frac{1-k}{1-2k+\sqrt{\frac{\alpha}{\beta}} \times (\frac{\beta}{E_{imp}})^{\frac{3}{P_S}}} = \frac{1-k(m)}{1-2k(m)+\sqrt{\frac{34000}{34}} \times (\frac{34}{6.25})^{\frac{3}{6.25}}} \tag{3.10}$$

Where both k and E_{imp} are function of the mass.
Once the drone falls, the area of exposure have to be calculated with the formula:

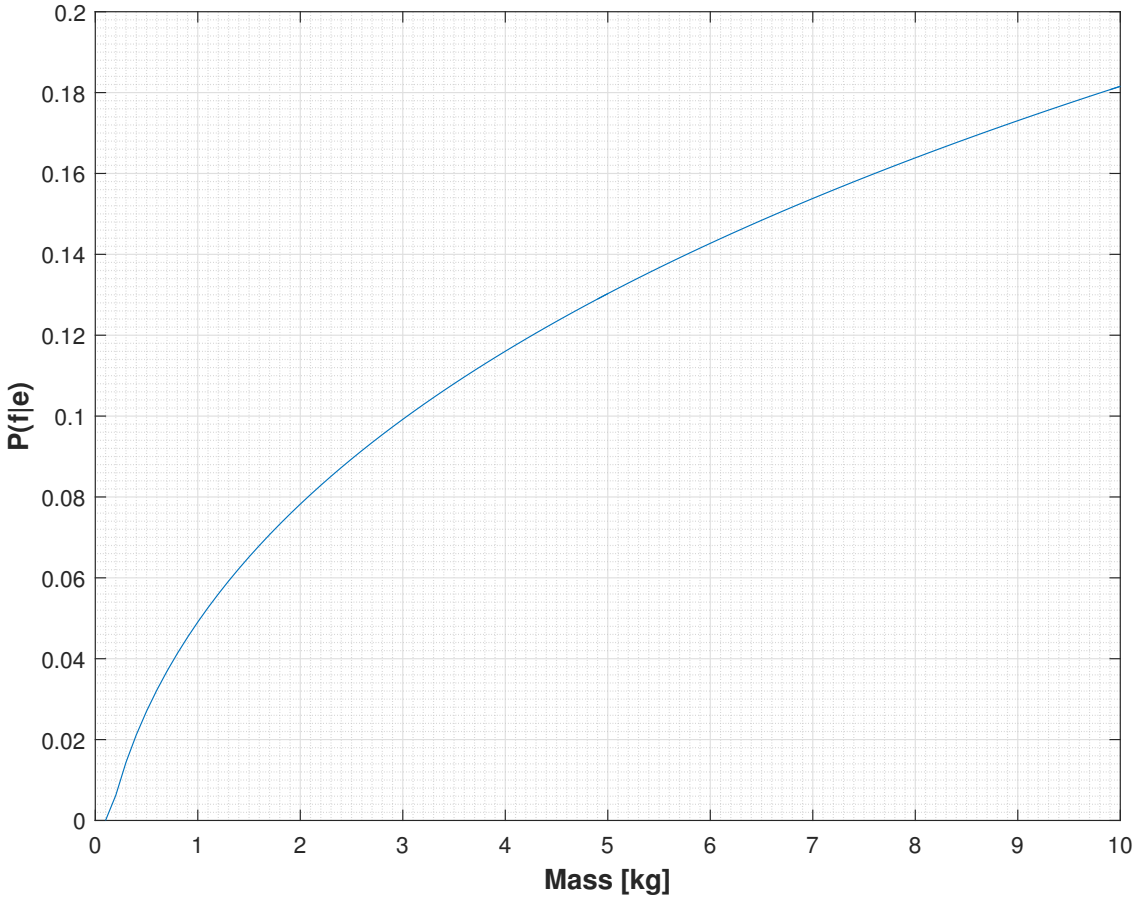


Figure 3.8: Evolution of P(fatality | exposure) in function of Mass

$$A_{exp} = 2 \times (r_p + R_{UAV}) \times d + \pi \times (r_p + R_{UAV})^2 \tag{3.11}$$

where:
$r_p = 0.23m$ = mean vaulue of human body radius

R_{UAV} = max linear dimension of UAV, variable
$d = \frac{Hu}{\tan\gamma}$ = orizzontal distance travelled by UAV during the fall. Measure Unit: m
$Hu = 1.75m$ = mean value of human height
γ = glide angle = 45°
The glide angle only exists for fixed-wing drones, since quadcopters are supposed to fall on their vertical line ($d = 0$).
In this case, the variable parameter is the length of the UAV R_{UAV}. Since we are working with civil drone, it is possible to suppose a variation from 0.1m to 10m (that is a quite huge US) and plot it to see the evolution of impact area in function of drone's linear dimension. What emerges from

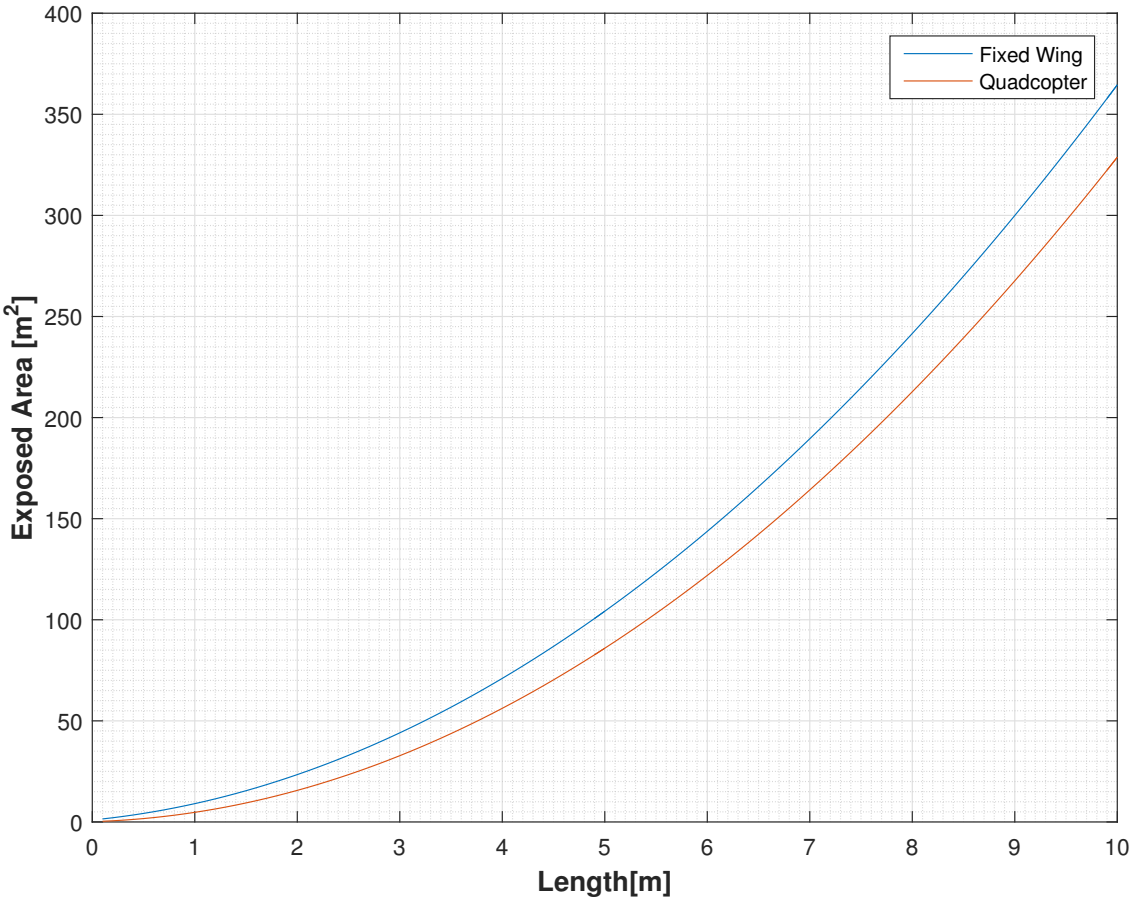


Figure 3.9: Evolution of impact area A_{exp} in function of drone's length

figure 3.9 perfectly matches with the theoretical considerations we did. First, it is interesting to notice that, as for P(fatality | exposure), the evolution of A_{exp} respect to the Length is not linear. Then, the superiority of quadcopter (in term of safety) that we supposed from the equations also becomes evident, since for every value of Length the exposed area is extremely lower.
Finally, we impose UAV's mean time between failure equal to $100h$: consequently, the rate of crashes will be $\lambda = 0.01$. At this point, we are able to compute the risk of the cell, applying the famous risk equation:

$$f_{F,i} = N_{i,exp} \times P(fatality|exposure)_i \times \lambda = \rho \times A_{exp} \times P(fatality|exposure)_i \times 0.01 \quad (3.12)$$

That finally becomes:

$$f_{F,i}(m,l) = A_{exp}(l) \times P(fatality|exposure)_i(m) \times 6 * 10^{-5} \tag{3.13}$$

Where $A_{exp}(l)$ and $P(fatality|exposure)$ are variable and in function respectively of length and mass.
Before evaluating the risk with equation 3.13, we have to impose an upper bound for the risk, the so called f_{Max} parameter. It's choice is demanded to the flight authority, which knows the level of safety for unmanned mission they want to impose. The actual state of the art for manned aviation imposes a value of $f_{Max} = 10^{-7}$, that however is too conservative for drones. Following the previous section, for this analysis has been chosen a less stringent value of $f_{Max} = 0.0001$. Obviously, independently from f_{Max} , the risk analysis works well since it has been developed in a parametric way. The only difference is that with a more conservative risk's upper bound, to many more drone will be denied the authorization to fly.
In any case, we proved that risk is both function of R_{UAV} and Mass. This makes us capable to plot in a 3D space the evolution of risk respect this two parameters. The results is shown in figure 3.10. The red plane cuts the curve at an height of $f_{F,i} = f_{Max} = 0.0001$: this means that all the

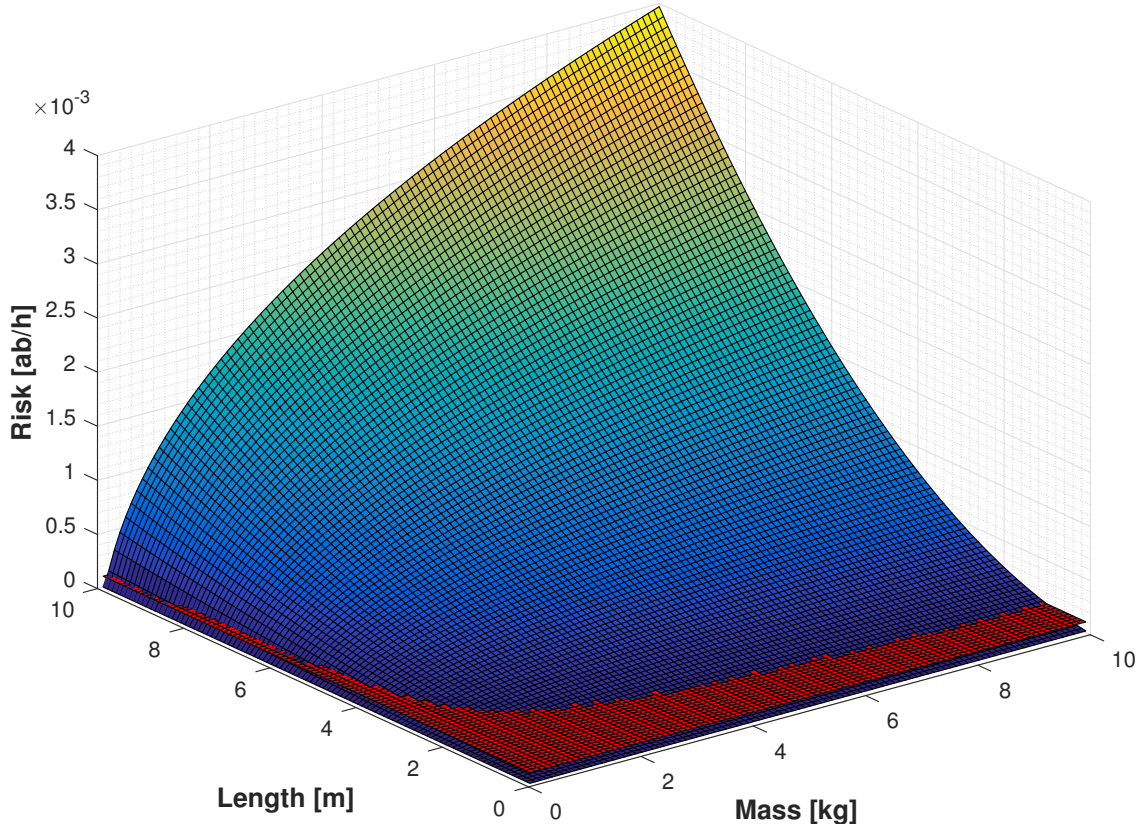


Figure 3.10: Evolution of Risk $f_{F,i}$ in function of quacopter drone's length and mass

drones whose building parameters' combination brings to the upper part of the curve cannot be authorized to fly. Although the upper bound is not too much restrictive, however just a few kind of UAV can fly on urban scenario, in detail only the ones that are more or less around the length of 1m and the weight of 1 kg. The curve related to the fixed wing has not been reported, since it

has the same shape but is obviously still more restrictive.
To complete the analysis of the risk in function of the UAV's building parameters, we can apply our risk assessment's technique to a set of common civil drone. Remembering all the conditions (urban environment, ecc.) previously exposed, the result we obtained, including the normalized risk R_i, are presented in table 3.3. Once again, the fixed wing drones arises as less safe than the

Name	Type	Mass[Kg]	Length[m]	$f_{F,i}$	R_i
Mavtech AGRI 1900	Fixed Wing	2.5	1.9	1.16×10^{-4}	100
MavTech Q4E	Quadrotor	2.6	0.6	1.1874×10^{-5}	34.45
DJI Phantom 4 PRO	Quadrotor	1.388	0.35	3.9×10^{-6}	19.77
Parrot Bebop 2	Quadrotor	0.5	0.38	1.8945×10^{-6}	13.76
Husban FPV Fly Hawk	Fixed Wing	0.18	1.2	2.96×10^{-6}	17.21
Drone ST 4	Quadrotor	0.035	0.165	0	0

Table 3.3: Common drones risk assessment

others. Moreover, it is interesting to notice that the Drone ST 4 is too small to have dangerous effects on people (Kinetic Energy less than 34J)and its risk is equal to 0.

3.4.5 From Static to Dynamic Risk Map

At this point, we provided all the instruments and techniques to correctly generate a risk map for an Unmanned Aerial Vehicle. Starting from the risk assessment concepts, we create a series of methods to encapsulate all the information on a certain area and gives back a number to represent the risk of flying there.
What we did is not so obvious since for the first time a cloud system, already programmed and simulated, is capable of collecting all the data and analyze them in a coherent way. This is the first step to a common standard to use in the drone's flight.
However, what we realized it's a kind of open chain system: all the input are provided to the "Map Generator" block, that gives as output the proper risk map. In figure 3.11 the actual architecture. This framework has many limits. The first is the impossibility for the drone to provide data back to the cloud: it just receives the map, and cannot co-operate with the cloud, updating it with the information provided by its on-board sensor (see also figure **??**). Furthermore, according to figure 3.11, the input are received by the cloud just one time, before the mission starts. Once it provides the risk map to the UAV, its work ends. This means for example that real-time updating on the input are not handled by it, so are not communicated to the drone.
Many times during the previous sections we explained that many of the variables taken into account should be in function of time. Once again, this part is crucial to have a real operative safety framework. Our final aim is to have a dynamic risk map, that changes in times according to the variation of its input. In figure 3.12 this architecture its represented. It's important to notice that on one side there is the "input" block, that in this case instead of being static, can dynamically update the risk map according to new information. For example news about a particular event, or IoT based sensors distributed in the cities. On the other side there is the drone itself, that receives the path to follow and while flying can detect with its sensors on board if there is something different from what expected by the map. An example can be a building, not present in the risk map (maybe for a lack of information), but detected in real time by a laser scan on the UAV. What we propose here is that the aircraft not only avoids the obstacle (with the algorithms developed by my colleagues) but also update the cloud information, in order to have an every time more coherent and precise risk map. The system we developed already is capable to handle time varying

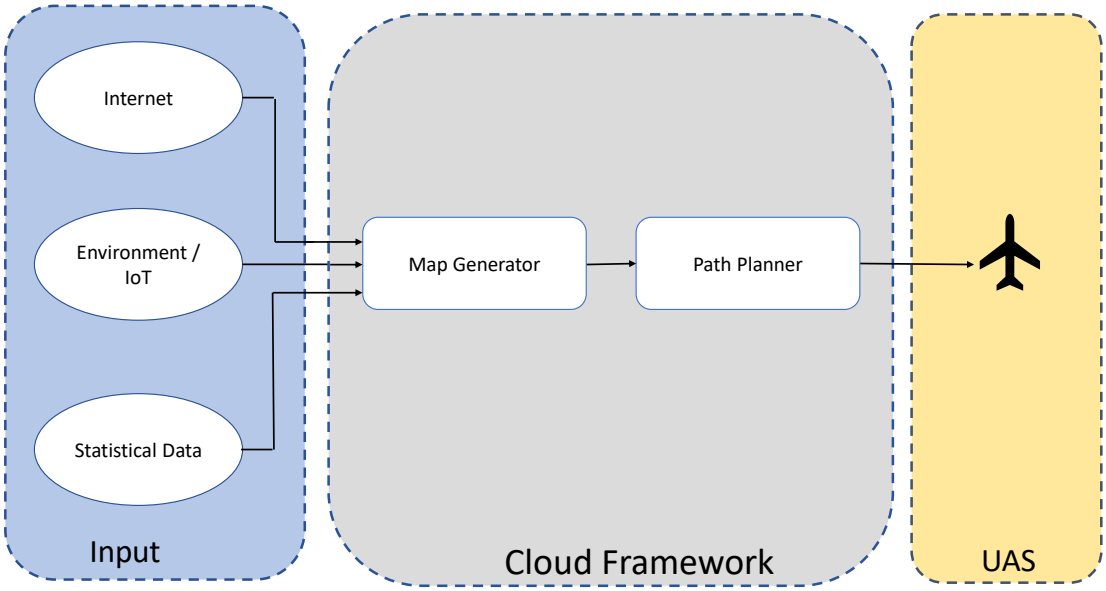


Figure 3.11: Open chain cloud framework

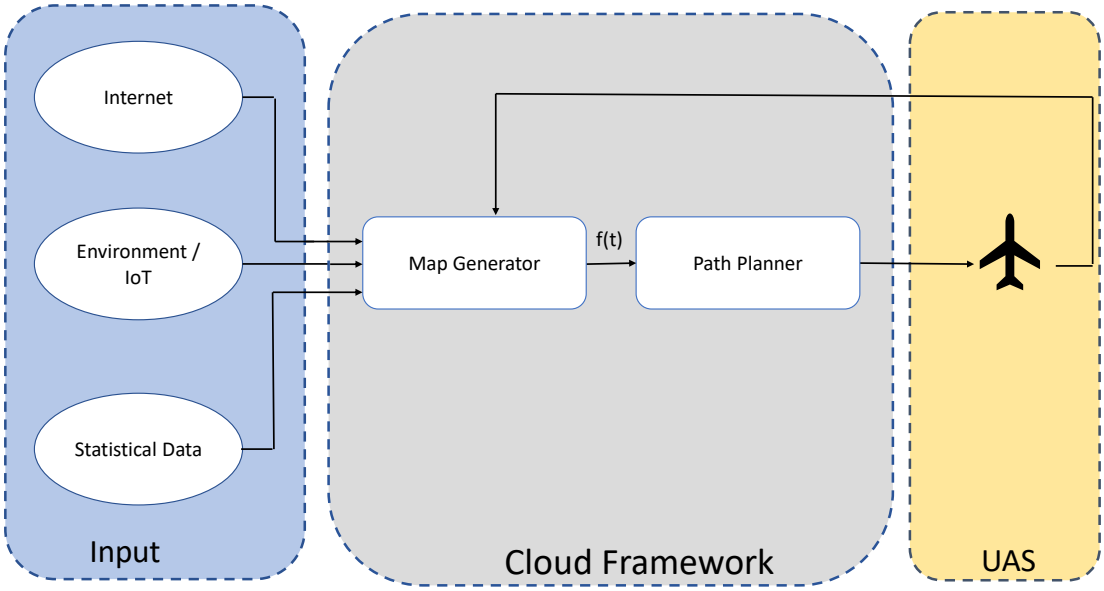


Figure 3.12: Closed chain cloud framework

input, and consequently updates the map. For what concern the UAV's sensor, instead, a more complex discussion must be done. Today, in many application in which UAS are used there is no need of a risk map, and the fly is almost completely sensor based: when an obstacle is detected,

the proper avoidance algorithm is used. According to this point of view, one may argue that all the other stuff presented is useless, or at least redundant. Some answers to explain why we didn't consider only on board sensors are needed.
First of all, what we developed is a framework independent from a specific kind of drone. We aimed to have a very generic and multi-purpose system, able to work with almost every UAV on the market. According to this, the assumption of considering each aircraft equipped with a sensor is too restrictive. We don't know if and which sensor is on board, so we must consider the worst case scenario, where the UAV is blind. It's important to notice that not-equipped drones are not so uncommon as it seems: we are working in a urban environment with low price items. The presence of measure instruments has three effects: it increases the weight and consequently the probability of causing fatal injuries if it falls, reduces the flight's autonomy and it also increases the cost for building the aircraft.
The optimal scenario for us its a very light drone, also provided with sensor on board, but it's very hard with cheap objects. What we developed is a system capable of handling every aircraft that wants to join the operative scenario.
Finally, one last thing is crucial to better understand the potentiality of this system. As we said, very cheap UAS are very often blind, since there is no money and space to introduce sensors on board. However, they will move in the same scenario of bigger drones, that thanks to their sensors will update the map. Practically, less equipped aircraft will benefit from the presence of the other ones, though they doesn't give any input to the "Map Generator" block.
This makes our framework dynamic, conservative and cooperative.

3.4.6 Mission's Risk

Once the path planner algorithm completes its job, the result is a series of geographical coordinates (way points) that the unmanned vehicle has to follow to complete its mission minimizing the risk. Practically, this means that the Traffic Manager knowns each cell that the drone will pass over, and thanks to the risk map also the $f_{F,i}$ value associate to each of them.
However, we still miss the last step: on one side we have the trajectory, created to provide the less risky path, on the other we have the maximum acceptable risk for the overall mission $f_{F,Max}$. Finally we must find a way to calculate, starting from its trajectory, the risk f_F associated to the mission in order to compare it with the maximum safety value imposed by the local national flight authority. Then, two situations have to be faced:

$f_F \leq f_{F,Max}$ The path chosen by the path planner guarantees a safety level that satisfy the imposed standard. The mission is approved.

$f_F > f_{F,Max}$ The path chosen by the path planner doesn't guarantee a safety level that satisfy the imposed standard. The mission is not approved. It's possible to change some parameters in the risk map and trying again with the path planner.

Please notice that the mission's risk analysis have to be done a posteriori, when the path has been chosen. In figure 3.13 a schematic representation of all this framework is provided, while later in this section the method to evaluate the f_F will be explained.
Once the trajectory to follow is known, the cloud framework has to estimate both f_F and $f_{F,Max}$. The first step in order to accomplish this is to calculate the overall mission's time lenght τ_{Miss} and the time spent in each sub area of the map τ_i. This procedure can be easily performed looking at the data sheet of the unmanned vehicle: in it, it's possible to find out the cruise speed v of the drone, that is enough for this calculation. Infact, known the maximum linear length of the area, we can apply the formula 3.14 to evaluate the time τ_i spent in each subzone.

$$\tau_i = \frac{l_i}{v} \tag{3.14}$$

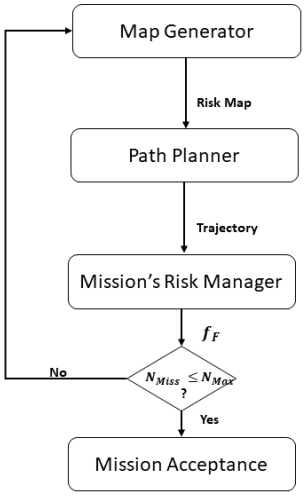


Figure 3.13: Mission's risk managment and acceptance

Where:
l_i is the maximum linear length of the i-th area. Measure unit m.
The importance of knowing the time window the drone is supposed to spend in each area is evident, since the risk is expressed in victims per hour of flight. What we do in equation 3.14 is just a rough estimate of it, since it considers the drone flying always at its medium speed and straight along the side of the squared area. Although this level of precision can be enough, a better estimate can be otained thanks to the path planner.
As said before, at this point of the analysis we already know the trajectory that will be performed by the aircraft (for detail, see the proper section). This means knowing the real line of flight (not necessarily the square side) and the speed. Using this information in equation 3.14 will provide the real time spent by the drone in each cell.
Finally, if we define n as the total number of cells overflown by the UAV and $f_{F,i}$ the risk associated to each of them, the overall estimated number of victims N_{Miss} for the mission will be:

$$N_{Miss} = \sum_{i=1}^{n} f_{F,i} \times \tau_i \tag{3.15}$$

The operation just performed appears simple: on one hand we know which cells of the map will be overflown by the UAV, and the time it will spend on each of them, on the other the victims per hour of flight of that cells. The product of this two elements provides the number of victim per cell that the aircraft will produce. Finally the sum of all the N_i of the trajectory gives back the total number of victim of it.
I think it is important to stress one key concept that emerges at this point, that was very hard to face at the beginning of this project. When we started, two different ways to evaluate the risk were possible: the first one is the one already presented here, where the risk is in victims per hour of flight. The second instead aimed to provide an estimate of the victims the drone could do overflying a given area. Practically this second approach didn't consider the time spent on each

cell, providing just a probability of having a victim if the drone fly over that area. This approach was interesting because worked with probability, but had a great problem due to the lack of time in his expression. This can be explained with an example: in the first approach, the more an UAV stays on a map's cell the more the risk increase. This is correct. In the latter instead, the risk didn't change in time, so a drone that simply flies over the cell and a drone that instead loiters on it had the same risk. This is wrong. Furthermore with the risk analysis here presented it is possible to change the sub-area dimensions and this will consequently change the risk, since the drone will spend more or less time on it. This feature also is key for our cloud framework.
Anyway, the idea of merging this two ways to approach the risk fascinated our work group, and finally found its expression in the mixed approach we are going to present. Let's have a look to the flowchart of figure 3.13. What comes out from the "Map Generator" block is the risk map of a big area (for example a city) for a given drone, at a given time and altitude (It's important to repeat this concepts to understand the overall approach we are proposing). In this map, each sub-area, whose dimension is due to the risk map's resolution imposed by the cloud framework, has its own risk, which is in $\frac{ab}{t}$ and is evaluated without considering the UAV's trajectory (that is obviously still unknown). At this time, the risk of each cell is independent from the other cell and its position respect to them. The formula used is the one exposed in previous chapter:

$$f_{F,i} = N_{i,exp} \times P(fatality|exposure)_i \times f_{EFT,i} \tag{3.16}$$

Where each term is the one related to the i-th cell.
It's important to notice that at this point the "time element" is introduced by the term $f_{EFT,i}$, that, as said in chapter 2, can be considered as the frequency of failure of the UAV, or as:

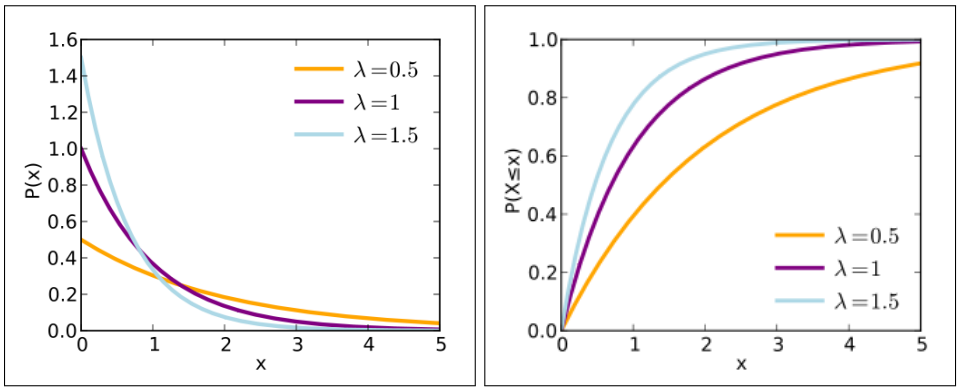
$$f_{EFT,i} = \lambda = \frac{1}{MTBF} \tag{3.17}$$

Where λ is the inverse of the Mean Time Between Failure (MTBF) of the aircraft.
Since λ is function of a constructive parameter of the drone, it will be the same for all the cells of the risk map. Once again, when building the risk map each cell is treated without considering the past path of the drone, but as it was just started.
Upon this map, the "Path Planner" block works to find the better trajectory, and then provides its waypoint.
Finally, according to figure 3.13 is the turn of the "Mission's Risk Manager" block, which has to evaluate the overall mission's risk, given the trajectory. This part of the work can be done as explained before, or introducing a new probabilistic term, to take into account the probability of having a crash. Practically, what we are going to do during the overall risk calculation is not to consider each part of the trajectory as independent from the others, but instead taking into account when it is going to be overflown by the UAV. This probability term will be both function of λ and of the time interval spent of that area. In formula we have:

$$N_i = N_{i,exp} \times P(fatality|exposure)_i \times P(C|NC)_{i,i+1} \tag{3.18}$$

Where $P(C|NC)i, i+1$ is the probability of having a crash in the time interval [i,i+1] given that the aircraft didn't crash before time i. The result is the estimated number of victim for that area, knowing how many time the drone will spend on it and when it is going to fly over it. This point is crucial.
How to model $P(C|NC)i, i+1$ will be now exposed.
In probability theory and statistics, the exponential distribution (also known as negative exponential distribution) is the continuous probability that describes the life length of a process that doesn't get old. In this kind of process, or Poisson point process, the events occur independently, continuously and with a constant rate λ. This distribution is used in many different fields, in

particular physics (to model the decay of radioactive particles), queuing theory and in all the applications of reliability theory. In figure 3.14 are presented its probability density function(pdf) and the cumulative distribution function(cdf). In formula, the pdf and cdf are is:



(a) (b)

Figure 3.14: Exponential distribution: pdf and cdf

$$f(x,\lambda) = \begin{cases} \lambda \exp^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x \leq 0. \end{cases} \tag{3.19}$$

$$F(x,\lambda) = \begin{cases} 1 - \exp^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x \leq 0. \end{cases} \tag{3.20}$$

While the expected value $E[X]$ is the Mean Time Between Failure:

$$E[x] = \frac{1}{\lambda} = MTBF \tag{3.21}$$

The most interesting property of the exponential distribution is the memorylessness, crucial in the case of conditioned probability (as our situation):

$$Pr(T > s + t | T > s) = Pr(T > t) \tag{3.22}$$

The probability of an exponential random variable exceeding the value s + t given t has the same probability distribution as the variable originally exceeding that values, regardless of t. Practically this means that past events has no effect on the future.
The choice of the exponential distribution has been made according to many studies about drone's reliability, also developed inside the aerospace department of Politecnico di Torino. However, the modularity of this approach allows us to eventually change the model of $P(C|NC)i, i+1$ if something new is found.
Finally, with this new metric is possible to provide the total number of victims of the chosen trajectory:

$$N_{Traj} = \sum_{i=1}^{n} A_{exp,i} \rho_i P(fatality|exposure) P(C|NC)i, i+1 \tag{3.23}$$

In this section we provided all the instruments to properly evaluate a trajectory in function of the risk. We moved from the definition of a risk map till the definition of risk for a path, whose

measure unit is in victims. We are now able in determining between different paths which is the best, or eventually what changes should be done in order to have a safest trajectory.
As said before, we have now a safety metric.
One way to use it is to verify if the safety standard are satisfied: calling $\tau_{Miss} = \sum \tau_i$ the estimate of the time length of the mission, the maximum acceptable number of people victim is:

$$N_{Max} = f_{F,Max} \times \tau_{Miss} \tag{3.24}$$

That said, the criteria to evaluate if the mission satisfy the standard is:

$$N_{Traj} \leq N_{Max} \tag{3.25}$$

In the "Path Validation" section the procedure of acceptance of a path will be explored in depth.

3.4.7 Objective Cost Function: From Risk Map to Cost Map

Till now, we explored the main assumption and techniques that have been used in the Cloud-Based UASs Traffic Manager to create a risk map for the drone's flight. We focused, also in the previous chapter, only on the risk topic, since this is the first step to perform in order to get the allowance for the Unmanned systems to fly over cities.
However, is quite obvious that beside the risk there is also the mission itself, which has is goals and features. When planning the path for the drone to follow, it is important to find also a way that satisfies all the specifics imposed for that mission.
Practically, what we want is to to have a mission planner that tries to provide the best trajectory for the UAV in order to satisfy all is goal: then it has to verify if the risk bounds are satisfied and, if not, re-calculate another path. The risk map in this scenario becomes only a layer (the most important one) of a more complex cost map that will encapsulate all the information useful to accomplish the mission.
In order to better understand this concept, let's imagine to have a map (fictitious, just as example) that, cell by cell, describes the TIM's signal quality of service (which will be better explained later on) (see figure 3.15). The connectivity is a crucial point of this cloud based framework, since without a link with the cloud the UAV is blind. Moreover, we are working on a civil scenario, so let's suppose to have a data streaming from the aircraft to the cloud (maybe a video recording of the flying drone) that in order to be maintained has to provide a signal connection power greater than a lower-bound S_{lb}. In order to accomplish the mission, the path planner have not only to find the less risky path, but also guarantee that this path will have always $S_i \geq S_{lb}$, for every i belonging to the trajectory, where S_i is the signal power of each cell of the map.
As for the risk, let's imagine to have a normalized signal map in the range [0, 100], where 0 means no signal coverage and 100 the full power (blue and red areas of the figure). The cost map we are going to generate has to merge this information with the risk, taking into account that the path planner will minimize the cost of the path. How to define this cost is up to us.
The cost of every cell follows the formula:

$$C_i = \alpha \times R_i + \beta \times (100 - S_i) \tag{3.26}$$

Where:
C_i is the overall cost of the i-th cell.
R_i is the normalized risk of the i-th cell.
α and β are two weighting variables in the range [0,1]. $\beta = 1 - \alpha$
All this variables are without measure unit, since we are working with normalized values.
This way to parametrize the cost is the classical linear one. Every variable that affects it has its own weight, α and β in this case, that can be used by the cloud traffic manager to change the

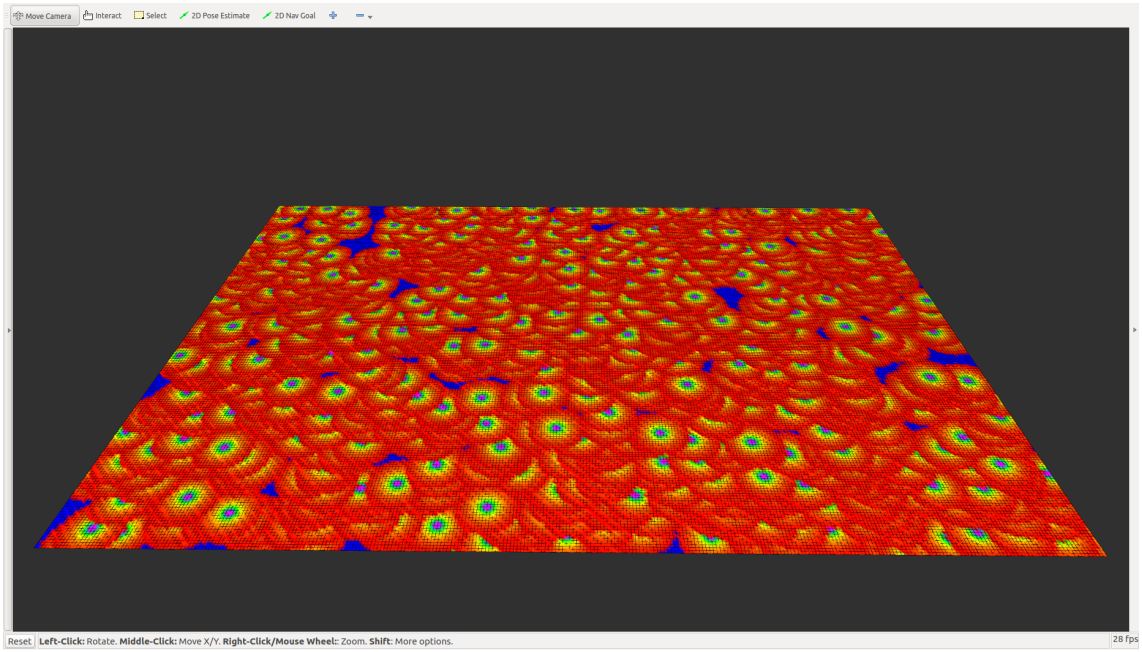


Figure 3.15: Signal power cost map

impact that each parameter has on the mission. Once defined this two values, the variables are simply added to evaluate the overall cost of the cell.
It's important to notice that since we have a minimizing path planner, the signal value must be subtracted to 100 in order to have the real cost: a cell with perfect signal (100 on its layer), should be always chosen by the path planner, so it must be equal to cost (0), and vice versa.
This approach can be generalized for a multi variable mission, so extended to the case in which the drone has different features to monitor in order to correctly accomplish the mission.
In formula:

$$C_i = \alpha \times R_i + \beta \times (100 - S_i) + \gamma \times I_i ... \tag{3.27}$$

Practically we can simply add each term, just taking into account that the final sum of the weighting parameters must be:

$$\alpha + \beta + \gamma + ... = 1 \tag{3.28}$$

In any case, from now on just the case with risk and signal power will be treated, in order to provide a case of study that is however simply expandable to more complex mission.
What emerges from this section is that the most important action the Cloud Based Traffic Manager has to perform is correctly choosing the weights: a too big α makes the risk too impactful on the cost, and the signal standard required will be probably missed. On the other hand, choosing an high value of β can brings to an unsafe situation, that will force the validation block to reject the proposed path. The procedure of choosing the weights, that now will be proposed, has been developed in parallel with the Path Validation, and can exists thanks to this: It's important to remember our framework is capable to recognize if a trajectory satisfy the requests (thanks to the metrics proposed before) and refuse bad path. This part will be exploited later on, but to understand the next part is important to know that the Validation block analyzes the path and refuses it if doesn't satisfy all the requests (in particular the risk bound): this practically means also than from the aggregated cost of the cost map, it can come back to the corresponding de-normalized risk value and signal power value.

The procedure we adopted, whose block diagram is reported in figure 3.16, is here presented in the proper order.

1. Set $\alpha = 0$ and $\beta = 1 - \alpha = 1$. In this way the risk doesn't impact on the cost map and consequently on the path decision.

2. Apply the Path Planner (P.P.) to the cost map. The trajectory found is the less costly one, and in this case means the top signal quality's trajectory.

3. Apply the Validation Block. Two situations can happen. One, the trajectory still doesn't satisfy the quality standard required for the mission. Since this is the best trajectory for the signal power, the mission has to be considered unfeasible and refused. Two, the trajectory satisfy the quality standard but not the risk one. In this case, another cost map has to be produced, this time starting taking into account also the risk analysis: $\alpha = \alpha + 0.1$ and $\beta = \beta - 0.1$. This new map is sent to the P.P, point 2.

4. The trajectory found by the P.P. satisfy both the requirements. This means that the mission is accepted and can starts.

The idea we followed is simple: let's start with the cost map only of the signal quality, without considering the risk. The path found will be the best quality one, because the path planner wasn't affected by the risk while evaluating it. If this satisfy all the requirements, then the mission is accepted. Otherwise, a change in the weights must be done: obviously, since no more importance can be given to the signal quality, if it is not satisfied then the mission is unfeasible.
As it's easy to imagine, the first set of weight ($\alpha = 0$ and $\beta = 1$) improbably will provide an enough safe path. Since the safety bound are very conservative, many iterations are required in order to find a good path. Furthermore, the step of 0.1 to add (or subtract) to the weights has been chosen arbitrary (it seems little enough but can be decreased) and can be changed. In any case, at the end of this procedure, the best acceptable compromise between TIM's signal quality and risk is found.
Finally, a resume of this section. At the begin we provided the definition of risk map, th reasons why we need it and how it can be built. Then we worked on risk, normalizing it and providing numerical studies on drone's safety. Then, a way to pass from the risk associated to a cell to the overall risk of a given trajectory has been explained, providing a completely new metric.
Finally, we used the power of the "layered structure" to move from the risk map to a cost map, extending the mapping procedure to all the other variables that may be interesting for an Unmanned mission. Then a formal way to build it has been shown, in the equations 3.29 and 3.27. In the end we proposed a way to choose the weights, showing how the cost map have to change in order to provide to the path planner an optimum compromise where is possible to find the best trajectory.

3.5 Path Planner

The path planning problem is a really well known task in literature, and many different approaches can be used to solve it, according to the different needs of the robots that has to perform the trajectory. According to a quite common definition [34]:

Definition 9. The motion planning problem is a term used in robotics to address the process of breaking down a desired movement task into discrete motions that satisfies constraints and possibly optimize some aspect of the movement itself.

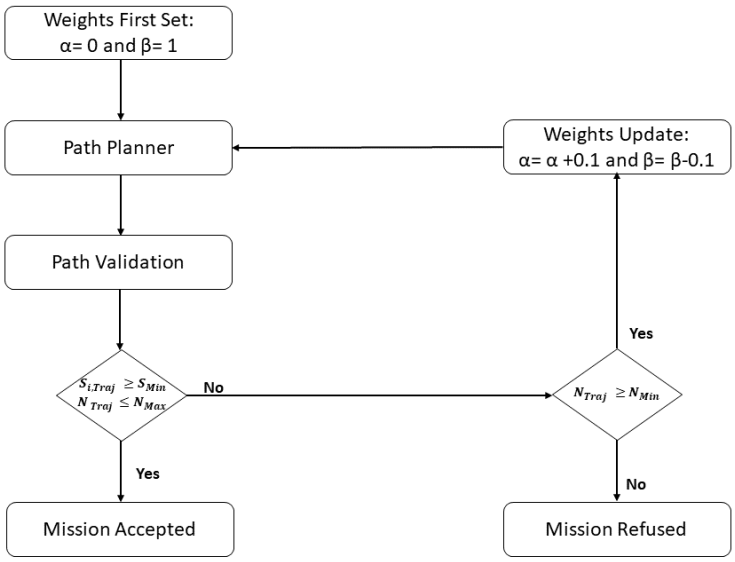


Figure 3.16: Weights assignment procedure

In other words, given the dynamic of a robot, the environment description and an ordered set of goal states (one or more, doesn't change), the path planner block of the Cloud Based Traffic Manager has to find a sequence of waypoints on the map that will bring the aircraft from one state to the following, till the last one is reached.
Please notice that, though we are working with Unmanned Aerial Vehicle, thanks to the assumption of not varying the altitude of flight we can treat the path generation task as if it was for a planar robot. Not considering the z-axe obviously have a great simplification impact.
During years, different algorithms were developed to accomplish this tasks. A quite rough but useful classification is:

Deterministic Algorithms Simple to implement, not hard from a computational point of view.

Probabilistic Algorithms Involves high dimensional state space,ans its execution time tends to infinite. Practically, if no solution exists it will run forever. [35]

At this point, an high level descriptive introduction to both this classes can be interesting and helpful to understand the choices made by our Path Planning group. Since it's not the main purpose of this thesis to develop new motion planning algorithms, we will not go too much in deep in this section, that will instead better explained by the other members of our research group. Just a brief resume of the most important algorithms and the actual state of the art of our Cloud Based Path Planner will be provided in here, in order to make the reader capable to understand the following sections and especially the Simulations.
In order to properly conclude this introduction to the path planners' world, It's important to underline the main features that we are looking to consider one algorithm better then the others. As said before, every approach can be considered optimal, according to the context in which it will is used.
Having a look to our environment, its crucial characteristics are:

1. Big Map. We are going to plan motion over cities (km^2) with a very high resolution. The resulting cost map will be huge and detailed. This implies the need of a light algorithm or the usage of a really great computational power, in order to have a solution in finite time.

2. Changing environment. The obstacles and many of the other risk parameter changes in time, often during the flight. The path planner have to be dynamical, that practically means able to change path in real time.

3. Critical Situation. While running during the flight, the path planner response have to be really fast, in order to prevent collisions. Since we are working with aircraft, a latency in the order of seconds can has catastrophic effects.

4. Drone's Structure. The Path Planner has to consider that the UAV is not a point mass, but instead has its own dynamic and structure. The trajectories have not only to be optimal in theory, but also feasible for the Unmanned System.

Now, let's have a look to the state of the art.

3.5.1 Deterministic Algorithms

The history of this branch of Path Planner starts with E. Dijkstra in 1959, who developed his famous algorithm to find the shortest path between two nodes in a non-negative weighted graph. Since the path planner iteratively follows the edges of the graph till it will find the goal, the Dijkstra one can be classified as a graph search algorithm.
Staring from this, that can be reasonably considered the father of all the motion planning algorithms, many others were developed in the following years. In 1968, Hart, Nilsson and Raphael, developed the A* algorithm, which is still a graph search algorithm but with an heuristic estimate, which classifies each node estimating the best way that passes through it. The result of this process is the shortest path (less costly one) between start and goal [36]. A typical drawback of A* is its computational requirements: on large map (as the one used for drones flight) many states has to be recorded, so a huge memory is needed. During the rest of the century, an impressive amount of deterministic algorithms has been deployed, from scratch or modifying the previous to have a more satisfactory solution. The most important are here resumed:

Dynamic A* or D*, was proposed by Stentz in 1994 [37]. It works exactly has A*, with the main difference that weights between two nodes can changes at run time. Thanks to this feature, it can be used to make real time re-planning of the path, if the robot is supposed to move in a changing environment. The change of cost it's usually detected by sensor mounted on the Unmanned System, and for this reason it is considered a sensor based algorithm.
Stentz itself tried to improve its work, and developed first the Focussed D* and then the D* lite algorithms [38][39]. Both of them are similar to D*, but save computational resources and have beter perfomances.

Theta* developed by Nash et al. [40] is still an extension of A*. The main difference with is predecessor is that for each vertex expansion there must be a line of sight between parent node and its successor. A very similar path planner is phi*. Both of them allows the robot to moves also along diagonal lines: this features, that also provides to them the name of Any Angle Algorithms, really increase the power and the efficiency of the planner.

RA* developed by Guglieri [41], is very most recent one. This algorithm was developed strictly for the risk analysis procedure, so it takes into account parameters as population density.

Many other interesting works in this field has been developed (as Floyd-Warshall, which is a mile stone, or Artificial Potential Field), but each of them has one or more characteristic that doesn't fit with the requirements of our particular scenario.

3.5.2 Probabilistic Algorithms

Probabilistic algorithms are the ones that have a degree of randomness as part of their logic. Usually, the probabilistic (or randomized) algorithms have an auxiliary random input that guide their behaviour in order to have good performance in the average case. The performances oh this kind of systems is a random variable too, as also the running time and the output [42].
Typically, this algorithms works on nonconvex, high dimensional spaces upon which a random space tree is built: in its construction is contained the randomness of the method, that usually takes casual samples from the search space. More the samples, higher the probability of having a correct solution. Furthermore, at every step (when a new sample enter in the tree) an obstacle free trajectory is built, alays checking its feasibility.
Due to their intrinsic difficulty, this algorithms were developed later then the deterministic ones, when computer had already a sufficient computational power to support them. The first path planning randomized algorithm was the probabilistic road map (PRM), developed in 1996 [43]. As Dijkstra was crucial for deterministic algorithms, PRM can be considered the founder of the probabilistic set. It basic idea is the following:

Pre-processing Phase Starting from n random samples, a set (roadmap) of collision free paths is built.

Query Phase The shortest path between start and goal points is found in the roadmap.

The PRM works good in high dimensional space, but has a limit in the construction of the roadmap, that can be challenging and some times infeasible.
In order to provide better performances, a set of new algorithms was developed starting from this. The most important are listed here:

PRM* is a variant of PRM, in which the radius is scaled according to the number of samples. The result is an asymptotic optimality and computational efficiency.

Rapidly Exploring Random Tree or RRT, was developed by LaValle and Kuffner in 1998 [44]. It is probably the most powerful and interesting algorithm presented till now, and one of its variant has been implemented in our Cloud Based Framework. For this reasons, it will be better exposed in a proper section.

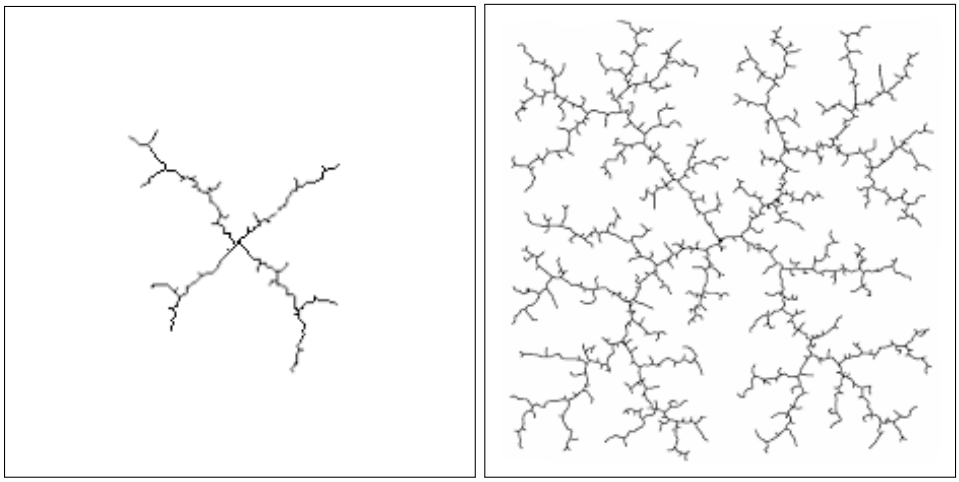
Rapidly Exploring Random Graph also called RRG, is an evolution of RRT. Practically, it is capable to provide quickly a first solution to the planing problem, and then monotonically improving it if more time is available.

Ant Colony Optimization developed by Maniezzo in 1992 [45]. It is a classical evolutionary algorithm, since its behaviour was inspired by real biological phenomena: in this case, ants looking for a path between their colony and food. Although it seems incredible, ants always find the shortest path, thanks to a random iterative process that leads to a common knowledge sharing between all the components of the colony.
ACO algorithm is very powerful and can rapidly provide a good solution. However, its converge time to the optimum is uncertain, and this can be a problem in applications like ours.

Now that we had an overview of the most common path planner, and the principles upon their are built, it's time to describe the one we choose for the CBUTM: RRT*.

3.5.3 RRT*

The algorithm we decided to implement to plan the route of our Unmanned Systems is the RRT*, probabilistic and derived from the classical Rapidly Exploring Random Tree (RRT).
The history of this method is quite recent, since it was firstly developed in 1989, by LaValle and Kuffner Jr, two American computer scientist and professors. It had also many variants and evolutions, among which the most important are RRT* and RRG.
The procedure executed by this algorithm aims to build a tree of open loop trajectories for non linear systems with state constraints: this capability of constructing feasible (for the UAV) paths is one of the key features of RRT. The tree is built extracting random samples from the state space, introducing also a bias to explore in the direction of unsearched areas. Every time a sample is drawn, a connection between it and the nearest state of the tree is attempted: if this link satisfies the constraints (practically, of it is feasible), the sample becomes part of the tree. Some kind of limitations can be introduced for the tree, for example in the length of the connection between the tree and the new state. A classical approach in case where the random sample is too far (i.e, it is farther than the maximum allowed distance) is to substitute it with a new state, at the maximum distance along the line that connects the sample to the tree.
Once this procedure is clear, it's easy to understand the way the algorithm works. The tree borns only with the initial state, than it starts to add random sample to it. This iterative procedure ends as soon as the tree contains a node in the goal state region. In figure 3.17 a typical evolution of RRT's tree is depicted.



(a) (b)

Figure 3.17: RRT's tree evolution in time

The advantages of such approach are many. First of all, it is simple and easy to implement. Moreover, the tree always remains connected and guarantees a feasible path at every step. Finally, it is proved to work better and faster then many other deterministic algorithms. The main drawback instead is due to the need of saving at each step the overall tree, that implies an increase of the computational time while the algorithm is running.
Starting from this, Karaman et al. developed in 2011 the so called RRT* algorithm, which really increased the optimality of the solution obtained with RRT [46]. The mechanism the rules RRT* are obviously almost the same of RRT, but it introduces two new interesting features: near neighbour search and rewriting tree operation. The first finds the best parent node for the new sample that

aims to be inserted in the tree, while the second rebuilds the tree within an area of given radius, in order to maintain always a minimal cost between tree's connections. Thanks to this, RRT* improves asymptotically the quality of its path as the number of samples increases, differently from RRT. In figure 3.18 the differences between the two approaches are shown: it seems evident that the tree built with RRT* is more ordered than the first one, thanks to the operation described before. Obviously, this features have also a computational trade-off, that can however be overcome

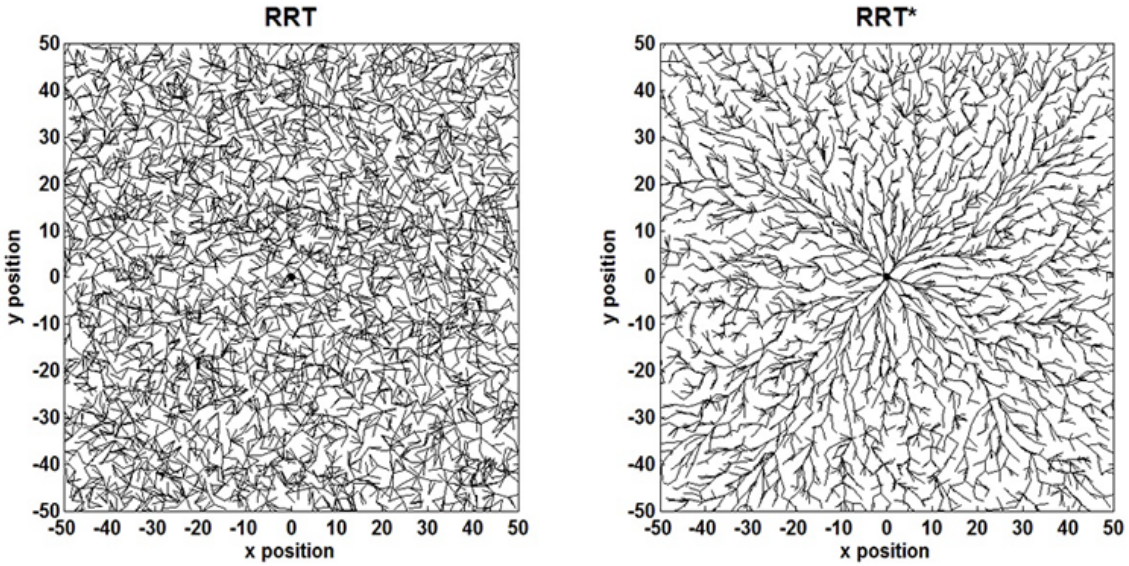


Figure 3.18: Differences between RRT's tree and RRT*'s one

with an high computational power cloud framework, as the one we have. Finally, after the studies of our path planning team, it seems that RRT* is the best compromise between efficiency and quality of the result. For this reason, it has been implemented in our framework as path planning algorithm for Unmanned Aerial Vehicle.

3.6 Path Validation and Risk Acceptance

The "path validation" procedure is a crucial point for the whole cloud based traffic manager. All the analysis and ideas exposed in the previous sections, from the risk assessment to the path planning, brings the system to this point. The aim of the Path Validator is simple to understand from an high level point of view:

Definition 10. The Path Validator block of the Cloud Based Traffic Manager takes as input the trajectory chosen by the Path Planner block, and provides as output the authorization (or prohibition) to fly for the drone.

Starting from the definition, is easy to see the importance of this block, which has to decide whether or not an UAV is able to fly guaranteeing the safety standard and all the other mission features. From a practical point of view this means implementing all the state of the art exposed in previous sections and chapter in order to properly analyze an Unmanned Mission. Furthermore, when a mission is marked as not compliant with the mission's requirements it has to propose different solutions (when possible) according to the causes that made the chosen trajectory not

feasible.

This last part is probably the hardest to implement: the Path Validator has to distinguish which requirement has not been fulfilled, and how the cost function can be changed in order to have a better trajectory (from this parameter point of view) coming out the path planner. This means knowing the different metrics of each parameter, and implement a coherent routine. Moreover, it has to assign the correct priority to each requirement, in order to avoid prevarications from one to another. At this point, it's important to remember that the primary feature for a mission is the safety: the aim of all this work is to guarantee a certain standard in term of victim per hour, and this bound cannot be crossed in any case, categorically. In this sense, a priority assignment to the mission requirements can be done only starting from this assumption.
Let's start going in deep with the path validation, starting from a quick overview of some of the possible different characteristics that can be of interest for an unmanned mission: this features are the ones (not necessarily all together) that the path planner will have to minimize (or maximize) in order to provide the best trajectory for the Unmanned Aerial Vehicle:

Risk is the most important feature for an unmanned mission. It has to be always compliant with the standard imposed by the national flight authority: practically, this means that the risk bounds are not decided by the UAV's owner, but are imposed to him. As already discussed, it can be measured in victims per hour of flight or, once the overall time length of the mission is known, in number of predicted victims.

Wireless Signal Latency is the time interval between the moment in which a packed information is send by the Cloud Based Traffic Manager and the instant in which it is received by the drone. The latency is usually measured in second (or milliseconds) using a ping procedure and limits the rate of data that can be transmitted.
Furthermore, it is possible to define a latency upper bound after which the communication cannot happens. It is evident that for a cloud controlled Unmanned System the latency can have catastrophic effects.

Wireless Signal Coverage represents the capability of the connectivity provider to connect the drone to the cloud. It is usually presented as a map, in which the areas where there is no coverage are blackened. Usually, a cloud based unmanned mission cannot categorically include this kind of area, since the drone will be blind for a quite long time.

Fuel Consumption is often crucial for the success of a mission. It's important to remember that we are considering civil and very often cheap drones, whose autonomy can be very low (in term of flight hour). This impose a new bound to the path planner, since it cannot chose a trajectory longer than the maximum autonomy of the unmanned system. In parallel to this, fuel consumption also implies a cost for the owner, that usually aims to minimize it.

Flight Time it the overall time length of the mission. According to the need of the drone's owner and to the aim of the flight, some missions have to be completed within a give time windows: exceeding the maximum time legnth (that means choosing a too long trajecotry) will affect the success of the mission.

Given Areas' Overflight is one of the most common ad useful features that can be imposed to a mission. Practically, it means to decide when and where the drone has to be: let's think to a surveillance mission, it seems obvious that the mission manager will impose some way points that the UAV have to pass.

Since we are not presenting a new wireless infrastructure, and we are not interested in technical details, in the rest of the work the two parameters "Wireless Signal Latency" and "Wireless Signal Coverage" will be incorporated in a single fictitious one called "Wireless Quality of Service"

(WQoS) that it's assumed to consider all the aspects of the connectivity between UAVs and the Cloud Based Traffic Manager. Obviously, as said before, if one new requirement should become of interest for the mission success, the cloud based validation block is, as all the one already presented, completely modular: once the quantitative description is provided, zone by zone, it can be easily added as a layer of the map and then inserted (with a proper weight) in the cost function: the final cost map, where the path planner works, will automatically incorporate this new feature. The modularity of the presented framework is one of is key point, and it is crucial to underline it when it's possible.
In figure 3.19 the flow chart of the Path Validator block is presented.
Once again it emerges from the image the centrality of the risk, which has the last word upon

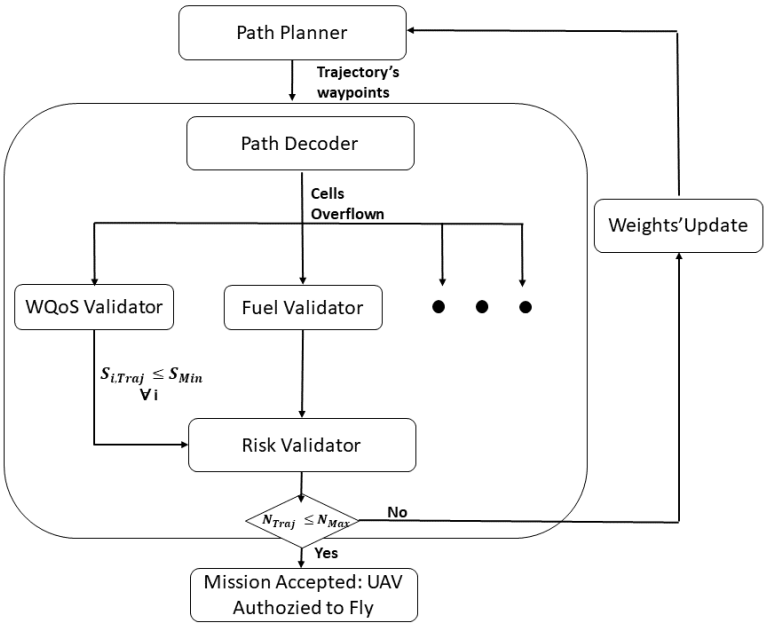


Figure 3.19: Path Validation Flow Chart

the authorization or not to the fly. For brevity reasons, only risk and wireless quality of service will be discussed, since their relationships can be consider as a model for all the other possible requirements. In the following, the blocks "Path Decoder","WQoS Validator" and "Risk Validator" will be exploded.

3.6.1 Path Decoder

According to figure 3.19, the first block encountered by the trajectory's waypoints(wp) coming out from the path planner is the "Path Decoder". Although it can be considered a more implementative object, it is interesting to explain now iy's usefulness, in order to provide a complete overview of what will be later on exposed in the future sections.
Starting from what we already discuss, we know that the path planner produces as output a series of waypoints that the UAV has to flyover in order to accomplish the proposed path. The waypoints are obviously not casual, but chosen in order to minimize a known cost function. A quite common defnition of waypoint is, according to [33] the following one:

Definition 11. A waypoint is an intermediate point or place on a route or line of travel, a stopping point or point at which course is changed. In modern times, it usually refers to a set of coordinates that specifies one position at the end of each leg (stage) of an air flight.

Starting from this, two points are crucial in order to better understand the rest of this section. First of all, we are dealing with coordinates, in a local or global reference frame, which identifies a point in the physical space, thanks for example to GPS system. Furthermore, the waypoints in the path provided by the path planner are not the all the cells of the map where the drone is supposed to fly, but just some "special" points at the end of each flight stage. Crucial aspect of the Path Decoder is to retrieve the real path the drone will perform from a waypoint to the following.
Let's have a look to the operation performed in this block, in a coherent temporal order:

1. Coordinate Transformation. The waypoints provided by the Path Planner are in a global reference frame, so we are dealing with longituine and latitude. The first operation to accomplish is the conversion from this, to the local reference frame of the map. A typical example is to convert each geo-referenced point in the corresponding "cell Index", obtained supposing to center the map's RF at the bottom-left side of the map. The "Index" obtained in this way is a couple of number, the first expressing the cell position on the x-axe and the second on the y one.

2. Path Decoding. Once each waypoints has its proper Index, it possible to retrieve the trajectory of the Unmanned System. In particular, we know that waypoints are located in places where the drone is supposed to turn (left or right, doesn't matter). This means that from a wp to the following, a straight line will be performed. Once the line is found(interpolating with a first order polynomial or with a proper line iterator, see Implementation chapter), also the cells crossed by the UAV are known.

At the end of the path decoding section, we are able to produce as output the Indices of all and only the cells overflown by the aircraft. In the next section, how to use this information to validate the path will be exposed.

3.6.2 Wireless Quality of Service Validator

As said before, the WQoS will be used in this part as example of validator, that can be extended and re-used for many other mission's parameter. The reasons why we decided to perform its analysis before all the other possible characteristics are two: first, the way it can be analyze is really archetypical, in the sense that it seems to be the more general validator case (not considering the risk, that instead as its own approach). Moreover, the context we are considering: completely autonomous cloud based aircraft. It's simple to understand that without wireless connection there is no possibility of having a safe flight. Once again, the WQoS is not a real parameter, but just a fictitious merge of all the interesting characteristics of a cloud network. It's usage in here is purely demonstrative and didactic.
In order to better going in deep with this and the next section, it's important to takes back the concept exposed about the objective cost function and the flow chart in 3.16.
The validation procedure starts looking at the input of this block: a series of the indices of all the cells crossed by the UAV. As we know, every cell has three information associated to it: Risk, WQoS and Cost. The last one is the aggregation of the first two, according to formula:

$$C_i = \alpha \times R_i + \beta \times (100 - S_i) \tag{3.29}$$

Where all the parameters(C_i, R_i and S_i) are normalized between 0 and 100 while $\alpha = 1 - \beta$ are both in the range [0,1].

What is interesting for us, in this section, is just S_i and how to vary α in order to have a valid path.
First of all, let's look at the metric of the WQoS parameter. Our idea was to consider it bounded in a [0,100], where 100 is a full power signal and 0 means a complete lack. Since we are using a path planner that minimize the total cost, in the cost map the S parameter is inverted, with the $(100 - S_i)$ operation: in this way, a cell with low quality signal will be more "expensive".
Then, it's important to define which bound can be applied upon this parameter. A typical scenario is the following: since the unmanned mission has to provide a continuous streaming of data (for example transmitting a video from an on board camera), the WQoS cannot be less than the lower-bound S_{min}. This bound is chosen or by the user, while defining the specs of the mission, or automatically by the cloud traffic manager, in order to prevent the UAV from going in areas where it cannot be governed.
S_{min} is still in range [0,100], with the same metric of S_i.
The validation procedure appears now very simple: once the cells and the bound are known, the cloud based traffic manager has to operate the following verification.

$$S_i \geq S_{min} \tag{3.30}$$

This must be done for every cell in the path. Then, one of this two scenario are possible:

1. The dis-equation is verified for all the cells belonging to the path. This means that the trajectory can be considered valid and consistent from a WQoS point of view. It can be passed to the risk validator to complete the analysis.

2. The dis-equation is not verified for at least one cell. The trajectory is not validated. As seen before, the cost function upon which the path planner works is built iteratively, changing α parameter. Initially, it starts from 0, and then is increased if a not-valid path is found. However, starting with $\alpha = 0$ means building the trajectory taking into account only the WQoS parameter, while the risk is not considered. Practically, this means that the α value we are using is the best one, and no change can be done if the path is considered to be not valid. The mission has to be marked as not feasible or S_{min} has to be decreased.

An important characteristic of the Wireless Quality of Signal emerges from this analysis, and can be extended to all the other requirements of the same type: we are imposing a cell by cell's bound, that must be verified for each area of the map. This is quite different from the usual "path parameter" as the risk, that instead must be verified upon the overall trajectory and not on the single cell. Every requirement that imposes a lower bound like this, can follow the proposed validation approach.

3.6.3 Risk Validator

Finally, the last part of the validation block can be approached. The importance of this element has no need to be underlined, since it has to approve or not the trajectory coming from the Path Planner, and consequently starts the mission. This crucial function emerges also from its position in the chain of the stateflow: at the end of the chain, it has the power to invalidate all the analysis performed in the blocks upstream. Once again, it's important to underline this concept: the main aim of our framework is to guarantee safety for people on ground, when an unmanned mission is going on.
The concepts here exposed are the natural continuation of the mission risk evaluation explained in section 3.4.6. In this part, however, we will enter more in detail, approaching the problem from

a path validation point of view.

According to the National Authorities standards, the Risk validator has to prove effectively that the risk for human beings is always below that upper bound. No compromise can be done in this sense, a path that exceed the maximum risk has to be marked as not valid, always.
Let's have a brief resume of the main concepts exposed before, in order to better understand the concepts we are going to face.
Once again, we have a series of cells crossed by the drone, which form the path of the drone evaluated in order to minimize a proper cost function, used also to build the corresponding cost map. In parallel with the cost, in range [0,100], each cell contains also the risk associated to it, in range $[0, f_{f,Max}]$, where $f_{f,Max}$ is the upper bound for the risk and is measured in $\frac{ab}{h}$.
As consequence of what discussed previously, two different objectives have to be achieved by the Risk Validator:

1. To check if every cell exceed the maximum risk value $f_{F,Max}$, i.e. is the path planner proposed one or more no fly zone.
2. To check if the overall risk of the path exceed the upper bound. Practically this means to verify $N_{Traj} \leq N_{Max}$

For what concern the first point, it can be easily reached acceding to the risk value f_i of each cell, and verifying the dis-equation:

$$f_i \leq f_{F,Max} \tag{3.31}$$

As for the WQoS (same kind of bound) if even just a single part of the path doesn't respect this condition, the path has to be categorically refused. According to what we said before, no fly zone cannot be accepted (and this is obvious even from their name!) in any case, since this will bring certainly to a crash or to a law violation.
The order in which the validation operation are performed is still not casual: if this first condition is violated, the second part of the risk validation procedure have to be avoided, since the path will be invalid in any case.
In any case, once the first step is over, it is possible to operate the trajectory risk analysis, in part already exposed previously: in detail, we saw from an high level point of view, hot to calculate the number of predicted victims of a given path, and this provided to us a metric to be able to differentiate a more risky path to a less one. This is not obvious.
Now, let's go in deep.
What we have at this point is a dis-equation, that has to be valid not cell by cell but on the overall path:

$$N_{Traj} \leq N_{Max} \tag{3.32}$$

Where the measure unit is in victim, not $\frac{ab}{h}$. How to evaluate this two element is the core of the validation process.
In order ro perform this analysis, let's suppose to know (and it reasonable) the cruise speed of the drone v, constant and commonly available on the data sheet. Then, since the waypoints to cross are known, it is possible to draw a stright line from one to the following (this is the trajectory the drone will perform) and simply measure the distance in meter between them. Staring from this, it is possible to evaluate:

$$\tau_{i,i+1} = \frac{d_{i,i+1}}{v} \tag{3.33}$$

Where:
$\tau_{i,i+1}$ is the time to fly from waypoint i to the following. Measure unit: seconds.

$d_{i,i+1}$ is the distance from waypoint i to the following. Mesure unit: meter.
v is the UAV's cruise speed. Measure unit: $\frac{m}{s}$
Applying this approach iteratively to all the n waypoints we obtain:

$$\tau_{Miss} = \sum_{i=1}^{n} \tau_{i,i+1} \tag{3.34}$$

Where τ_{Miss} is the overall time length of the mission, in seconds.
Although quite rough (we didn't consider the change of speed in turns), this approach provides a quite good estimate of the time required to complete the path. In any case, it can be adapted more precisely to every drone, once its characteristics are known. Since we are assuming to work with generic unmanned systems, it is impossible to go more in deep with this.
However, at this point one of the two part of the main dis-equation can be evaluated:

$$N_{Max} = f_{F,Max} \times \tau_{Miss} \tag{3.35}$$

For what concern the first term, we must take back the knowledge developed about the trajectory's risk evaluation. First, let's remember how we calculated the risk associated to the i-th cell, in term of $\frac{ab}{h}$:

$$f_{F,i} = N_i exp \times P(fatality|exposure) \times \lambda \tag{3.36}$$

Where $\lambda = \frac{1}{MTBF}$ is the failure rate of the aircraft, supposed constant and generally equal to 0.01. The idea, already exposed, is to modify the risk calculated taking into account also the time instant in which the drone approach the cell, in order to be able to evaluate the probability of having a crash in that cell. This operation can be performed because we already know the path, and consequently the time window associated to each cell. In formula, once we have the risk we must remove the generic term λ and encapsulate it into the more complex expression of the crash's probability $P(C|NC)_{i,i+1}$:

$$N_i = N_i exp \times P(fatality|exposure) \times \lambda \times MTBF \times P(C|NC)_{i,i+1} \tag{3.37}$$

Now, we are no more in the frequency of fatalities domain since we removed the time dependency. The term $P(C|NC)_{i,i+1}$ represents the probability of having a crash in the time window [i,i+1] knowing that the aircraft didn't crash before time i. The model we used to described it is an exponential distribution with parameter λ, which is the same of the drone.
The evolution in time cane be seen in figure 3.14.
To evaluate the crash conditioned probability, we have to take back its cumulative density function:

$$F(x, \lambda) = \begin{cases} 1 - \exp^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x \leq 0. \end{cases} \tag{3.38}$$

To arrive to the final formula for $P(C|NC)_{i,i+1}$, let's define:
i is the time instant in which the US enters the cell.
$i + \Delta t$ is the time instant in which the US exits the cell.
C is the event of a crash.
That said, what we want to calculate is the probability that a crash happens, in the time interval $[i,\ i + \Delta t]$, knowing that nothing happened till i:

$$P(i \leq C \leq i + \Delta t | C > i) = \frac{P((i \leq C \leq i + \Delta t) \cap (C > i))}{P(C > i)} \tag{3.39}$$

That can be further developed, thanks to probability theory:

$$\frac{P((i \leq C \leq i + \Delta t) \cap (C > i))}{P(C > i)} = \frac{P(i \leq C \leq i + \Delta t)}{1 - F(i, \lambda)} = \frac{F(i + \Delta t) - F(i)}{1 - F(i, \lambda)} \tag{3.40}$$

Now that we have the expression in term of cdf, just some algebra is needed:

$$\frac{1 - e^{-\lambda(i+\Delta t)} - (1 - e^{-\lambda i})}{e^{-\lambda s}} = 1 - e^{-\lambda \Delta t} = P(C < \Delta t) = F(\Delta t, \lambda) \tag{3.41}$$

This results is perfectly in accord (practically, we demonstrate it) with the memorylessness of the exponential distribution: doesn't metter what happens before the time instant i, since the probability is the same as if we considered $i = 0$.
Finally, we are able to evaluate the total number of victim of the path, simply adding the ones of the single cells:

$$N_{Traj} = \sum_{i=0}^{n} N_i \tag{3.42}$$

Now we can apply dis-equation 3.32 to validate the path.Two scenarios can happen:

1. $N_{Traj} \leq N_{Max}$. The path is approved and the Unmanned System is authorized to fly.
2. $N_{Traj} > N_{Max}$. The path is too risky to be approved. It is possible to find another path, recalculating the cost map imposing $\alpha = \alpha + 0.1$. Obviously, if $\alpha = 1$ the mission has to be marked has not feasible.

At this end of this section, the flight can starts. As resume of this chapter, we provided all the instruments to build the "Map Manager" part of a Cloud Based Traffic Manager. Starting from the risk assessment seen in chapter 2, we developed a series of techniques to pass from a generic definition of risk to a real risk map, with structured areas and well defined function. Furthermore, we expanded the concept of risk map to the cost map, including also other interesting aspect of the Unmanned Mission. Finally, we saw how to find an optimum path and the skills necessary to validate it.
This structure, merging theoretical and practical aspects, is the solid base upon which a coherent traffic manager can be deployed.

Chapter 4

Implementation

4.1 Introduction

The implementation of the cloud based unmanned traffic manager has been a crucial aspect of our work. Although in this thesis we divided in theoretical (the first part) and practical (this one) to perform a better exposition of our framework, actually the real work has been developed in parallel: it has no sense to build a perfect conceptual background, that has no link with the real environment in which its ideas has to be implemented. At the same time, it seems however useless starting programming without a main line to follow.
The process used to create the Cloud Based Unmanned Traffic Management has been cooperative, de-centralized and it has gradually brought us to define the theoretical standards while facing real implementation issues. In this way, the results we present are really operative and at the same time can be considered a step forward in the definition of a new state of the art for Unmanned Traffic Managers.
In this section, the CBUTM's flow diagram that we already saw in Chapter 3 will be presented in a different way, showing not only the logical links that connect one block to the others, but also the practical flow of data and information needed to properly works. Firstly, we will present the environment in which we developed our framework, which is the quite famous Robotic Operative System (ROS). Every results we presented, and each piece of code we wrote, has been obtained using only open source materials and software, starting from ROS itself. This caused many difficulties and slowdowns, but it is the only way to move in the actual scientific research environment. In any case, facing more issues makes, in my opinion, our framework more robust and expandable.
What we built, and we are now going to presents, is not the result of the work of a single man, but it's the achievement of our group entirely. Just for a matter of brevity and coherence, in this thesis only the map managing part will be explored in deep, while an introduction overview will present that framework in its totality.

4.2 Robotic Operative System

Robotic Operative System (usually known as ROS) is an open source, meta-operating system for robotic applications. More in detail, it is not a real operative system, but instead it better fits with the definition of middleware, so a collection of software frameworks, tools and libraries for robot's sofftware development [49] [50].
According to the description provided by the ROS community, it provides services designed for heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation

of commonly used functionality, message-passing between processes, and package management.
ROS processes can be described by a graph network, where each node is a process that exchanges data with the others through the edges of the graph, called Topics. In this way, each application (node) can transmits and receives simultaneously to and from the network a complete set of messages, which can be the standard ones or custom defined by the users, as for example sensor information, or the robot position in a map.
Nodes can be written using a huge set of libraries already existing in parallel with one or both the two programming languages C++ and Phython (for them, the ROS client library are called roscpp and rospy). The core library for the mapping is called Grid_Map, and will be presented more accurately further on. More in detail, a node is a process that performs computation. Nodes are combined together into a graph and communicate with one another using streaming topics, RPC services, and the Parameter Server. The advantages of using such nodal architecture are many, in particular an additional fault tolerance, since error are isolated inside the single node crash, and a decreased code complexity respect to monolithic systems [51]. Each node has a name and a type, used to identify it on the filesystem.
Two ways of exchanging data between nodes will be used in CBUTM:

Topic Topics are unidirectional streams of communication and can be represented as buses over which nodes exchange messages. Each topic has a publisher and one or more subscribers: nodes that are interested in data subscribe to the relevant topic, while nodes that generate data publish to the relevant topic. This means that the process of creating information is decoupled from the one of using it. Furthermore, usually nodes are not aware of who is using the data they produce. Topics main feature is the type of ROS message they transmit: once defined, only that type can be published upon it, and subscriber nodes can only receive this. At the state of the art, ROS communication protocols are TCP/IP-based and UDP-based.[52]

Service are used to answer a very common need of distributed systems, which is the request / reply interaction. Differently from the publish / subscribe model, this one solves the situations in which one node needs a function provided by another one. Practically, this is done via a Service, which is defined by a pair of messages: one for the request and one for the reply. A ROS node offers a service under a string name, and a client calls the service by sending the request message and awaiting the reply.
Like topics, services have an associated service type that is the package resource name of the .srv file. [53]

In image 4.1, a representation of this concepts it's provided. Finally,ROS is inherently open-source:

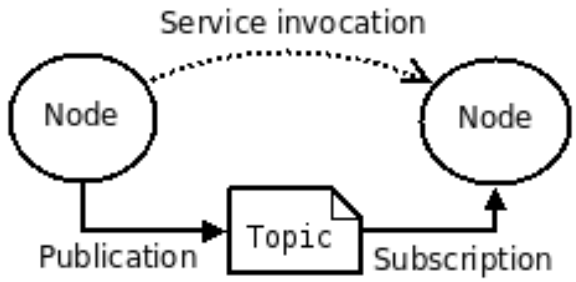


Figure 4.1: Node and Topics Architecture

this encourages collaborative work and software development, not only inside the same laboratory but with all the researchers who join it. In this way, a very hard problem as the robotics software

development can be addressed not only by expert computer scientist but also by a large part of robotic researchers.

4.2.1 Grid Map

Grid_map is a C++ library with ROS interface used to manage two dimensional grid maps with multiple data layers [47]. Its usage is very common in the mobile robotics application and in general in all the scenarios that requires both a real time mapping and an optimized computation effort. The data storage is based on the linear algebra library Eigen, which is very popular and provides versatile and efficient tools for data manipulation [48].
As said before, the key features of this packages are the layers and the way they can be used. Starting for example from a geographical grid map, it happens very often the need to add additional information to one or more of this cells. In this case, the grid map library uses the layers to store different type of data about the same point. A graphical representation of the multi-layered grid map concept is the one in figure 4.2. The Grid map library offers many functions in order to

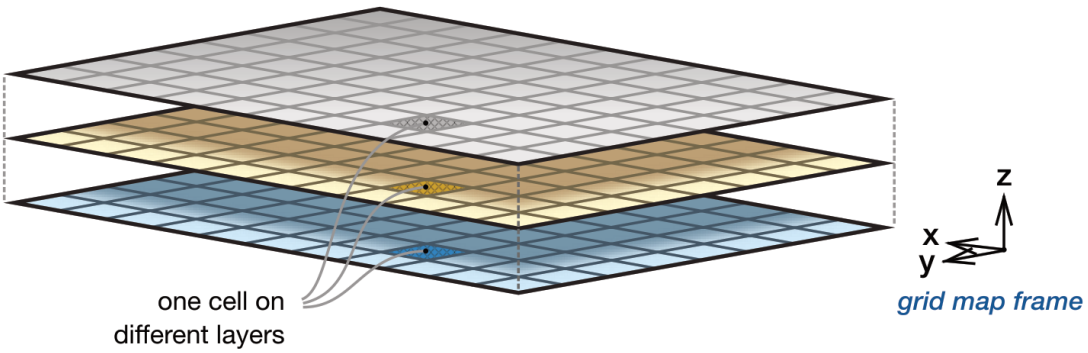


Figure 4.2: Example of Grid Map

initialize, manage and work with the layers and the most useful will be descripted in the following sections. In order to be consistent in the data's representation, it is important to define some geometrical parameters of the map [48]:

- Map's Reference Frame. The map is aligned respect to a specific grid map frame that can be set by the users.
- Map's position, which is the center of the map in the previously defined reference frame.
- Map's geometry, such as lengths and resolutions.

Once the map's geometrical properties has been set, it's possible to start working with the data contained in the layers. In particular, we can access each cells of the layers in order to manipulate the value stored in it.
For what concerns this work, we will apply the UAV's risk analysis previously exposed using the data contained in the cells as input, while the output will be the the corresponding cost value. The layer structures allows us to assign one layer to each of the parameters that characterize the risk assessment's procedure. Moreover, it will be possible, if needed, to modify the value stored in one or more layers at run time, while the drone is flying, thanks to the simple access tools provided by this package. This will allow the dynamism of the cost map.

Finally, this will bring to the construction of a last layer containing in every cell the cost for the Unmanned Systems that wants to fly over it, and it will be possible for the path planner to apply the techniques discussed before to find an optimal trajectory.
It's important to underline that, though each layer contains a different kind of information, the map is built over a real geographical map, so the path evaluated by the path planner over the risk layer is the real geographical path the drone should follow.
In order to visualize the grid map, the grid_map_visualization package provides a simple tool to convert ROS grid map message in RViz. All the following images are obtained thanks to specific RViz plugins.
RViz (or ROS visualization) is a 3D visualizer for displaying sensor data and state information from ROS. In our case, it has been used to print the cost map on screen, and to visualize the movement of drones upon it.

4.3 Cloud Based UASs Traffic Management: ROS Architecture

In this section, the overall structure of the CBUTM architecture will be explored. Actually, each one of the blocks that constitutes our traffic manager are completed and operative, and this will be shown in the Simulation chapter. It's important to underline that the implementation part has been central in our work, and it is probably an important part of our contribution to the research: an open source available Unmanned Traffic Manager, really implemented with the most advanced risk based theories.
The working principle of the Robotic Operating System itself, with different instances, namely nodes, running independently and communicating by means of topics and services, seems to be perfect for describing how the CBUTM system works. It can be seen, in fact, as the cooperation of 4 main nodes, each one main subject of the work of some colleague within the Joint Open Lab, but in the whole result of the cooperation of each.
In any case, though the work was developed by the whole team, this thesis wants to focuses only on the risk map managing so in this section just an overview of the CBUTM way to operate will be provided, while in the next part of the chapter we will go in deep for what concerns the map generation approach. The blocks that are not explained in detail here, will be obviously covered by the works of my colleagues.
As always, the best way to represent the logical approach of a complex system as the one we are describing, is using a flow chart of the nodes that constitute it. In particular, figure 4.3 has been obtained through the ROS topic RQT Graph: "rqt_graph" creates a dynamic graph of what's going on in the system. It is part of the rqt package and shows nodes and topics currently running and the relationships between them. For this reason, beside the nodes (and their name that we choose) there are also the topics in which they can exchange information.
The situation reported in figure is characterized by the presence of three UAVs: Iris_1, Iris_2 and Iris_3. Since each of them has its own topics and data, the resulting diagram is quite complex, and not easy to understand. The choice of having three aircrafts instead of one is due to the need of showing the capabilities of our collision avoidance system also in a complex scenario. However, to provide a better knowledge of how the CBUTM works, a logical flow chart has been developed in figure 4.4. In this image, a middle way between the pure logical(3.1) and the pure low level(4.3) is depicted. It is very useful to look at the ROS architecture in this simplified way, because it shows the relationships between nodes without being over detailed.
The Cloud Control Station (CCS) node, our version of the classic Ground Control Station, is meant to be the central entity of the system, the one in charge of holding relevant information's of the network and linking the other elements of CBUTM, managing the communication streams

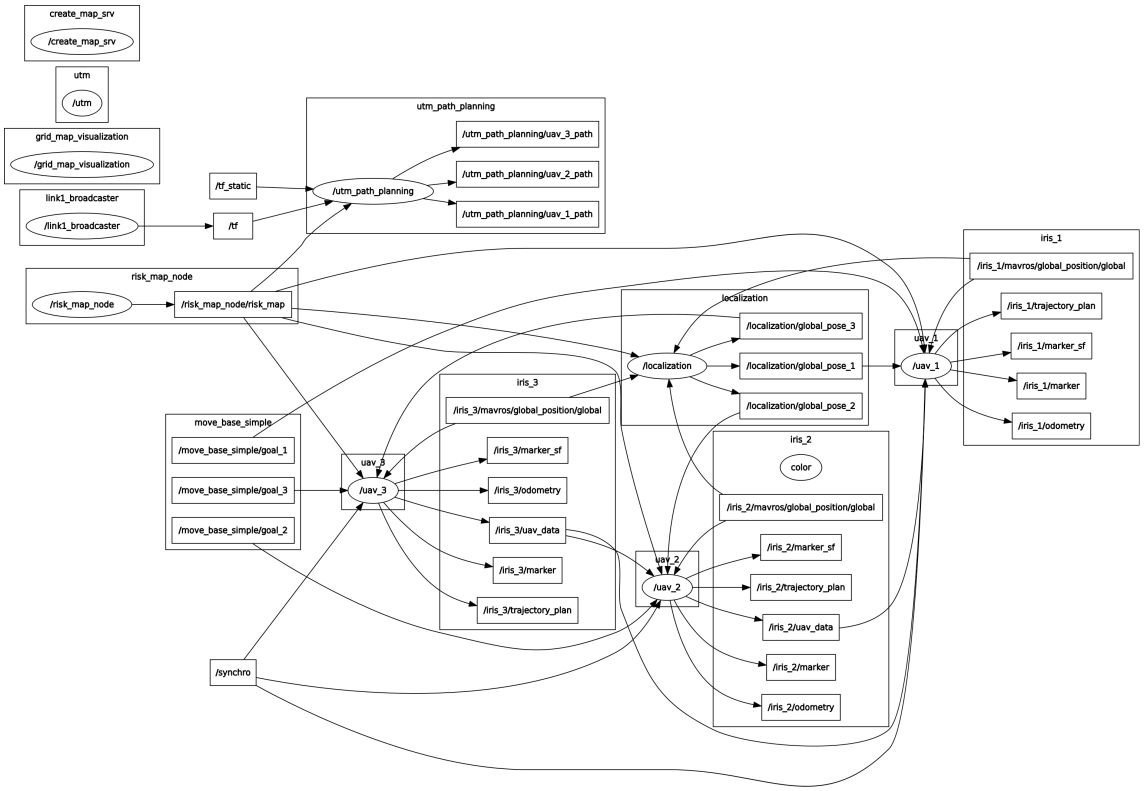


Figure 4.3: CBUTM ROS Architecture

within the system. It's job starts as soon as a user wants to join the network, since it provides the registration service, first step of the whole process. The user must call the aforementioned service using a proper client, using the registration.srv message shown in 4.1, through which it transmits its identity to the system (contained in the custom defined message shown in 4.2), and receives its computed priority level and the authorization to fly.

Listing 4.1: registration.srv message type definition definition

```
id_message uav_id
---
bool registration_status
float priority_level
bool response
```

The definition and then the transmission of uav's id happens through UAV node, which is intended to be the drone's virtual image on the cloud: in our idea, each aircraft that wants to join the urban airspace has to create its digital double, which will be used to communicate with the cloud traffic manager, to check its status regularly, and finally to guide the US during the flight.

Listing 4.2: id message type definition

```
string uav_name
string owner
```

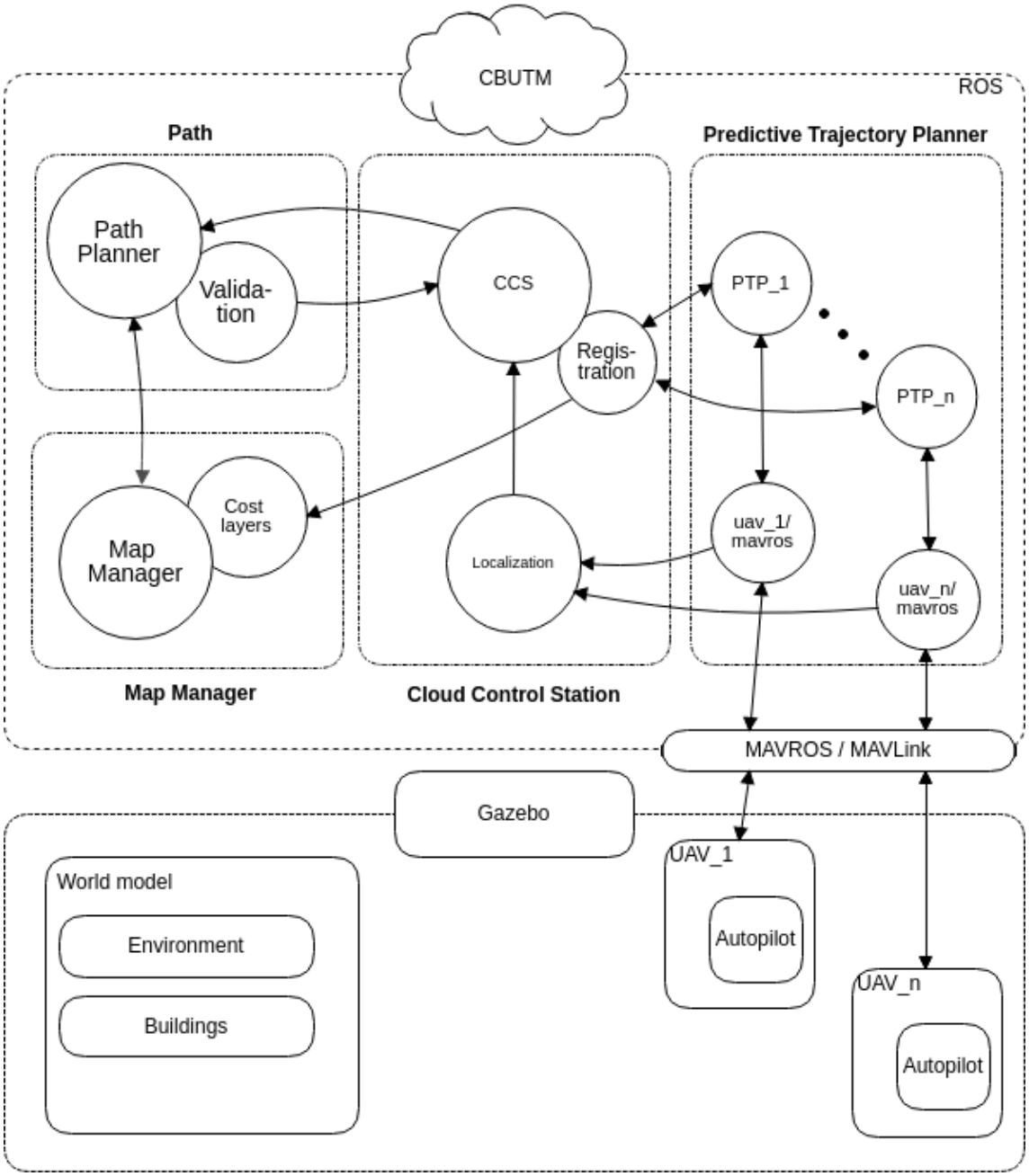


Figure 4.4: CBUTM logical Architecture

```
uint8 type
string model
float64 weight
float64 volume
float64 payload
```

```
float64 cruise_speed
float64 mtbf
float64 length

sensor_msgs/NavSatFix position
nav_msgs/Odometry odometry

bool has_goal
sensor_msgs/NavSatFix goal_position
uint8 mission_type
nav_msgs/Path global_path
nav_msgs/Path local_path

float64 fmax
float64 smin
float64 alfa

bool authorization_to_fly
float64 priority_level
bool registration_status
```

Once the drone's owner is in possession of all the information required for the registration (4.2), UAV node can starts running. Without entering in detail of all the parameters in 4.2, just let's classify them in macro-areas to understand their role:

Construtive parameter as weight, mean time between failure,ecc.

Path-related parameter as start position, global path,ecc.

Mission's requirement as maximum risk ($fmax$)

CBUTM' service parameter as the registration status, the priority level computed by the traffic manager,ecc.

Within the registration service routine (4.3), first the applicant is identified, and all the relevant informations are collected. Some of them are then used to compute its priority level, that in our tests has been supposed to be static and assigned at the beginning of the mission according to the mission's type and the vehicle characteristics. In a future moment, it could be turned in dynamic element, looking to the mission's priority in a specific time instant. Beside this, the registration service also awakes the Risk-Aware Map Manager, which firstly provides a model of the operative environment (working on point cloud, derived from .osm file), and then uses data contained in uav_id message to calculate all the layer of interest for the drone's mission. In following section, a detailed explanation of this procedure will be provided.
Last step of the registration service is to refresh the UAV list, that must contain an up-to-date database of the active vehicles, easy to check and fast to access for obtaining all the needed data. To do so, objects of class map and set are used, that are associative containers that store elements formed by a combination of a key value, used to uniquely identify the elements, and a value, that store the content associated to the relative key.
The insertion of the aicraft in the list is followed by the creation of a direct link between the CCS and the UAV nodes, aimed to the awareness of drone's disconnection, with the former checking if the latter's process has terminated, either cleanly or by crashing. Practically this means the presence of a specific topic in which the UAV node transmits a periodic signal (containing its id

reference) to the CCS, representing its heartbeat. Every time it's received, an initialized timer is restarted, while when the heart stop beating for a determined amount of time (more than 5 seconds reasonably) a timeout callback is run, advertising the crash and updating the UAV list.

Listing 4.3: Pseudo-code of the registration service

```
check_id()
{...}
compute_priority_level()
{...}
create_layer()
{...}
start_heartbeat_check()
{...}
```

The UAV insertion-removal from the list allows the CCS to know which and how many drones are flying in every moment, making it possible to build up an efficient tracking system. It is used to monitor the position of every drone, and check the distance between all the registered vehicle flying in the city airspace.
When the Map Manager ends its work, the registration phase is accomplished. At this time, the mission didn't start yet: a goal position has to be imposed, actually through Rviz's interface. Once start and goal are setted, the Path Planner node receives them thanks to a specific topic, and applies RRT* algorithm to find the less costly trajectory, working on the cost_layer that the Map Manager has built for the specific UAV.
Finally, Path Planner node calls the Path_Validation service, which aim is to guarantee the validity (respect to the mission's standards) of the trajectory, providing a boolean value equal to true they are satisfied. In this case, $authorization\_to\_fly = true$ is setted, and the mission can starts.
It's interesting to notice that, differently from all the others UTM, the CBUTM only needs an autopilot (i.e. Ardupilot) on-board, while all the in-flight computations are done on the cloud system itself. In particular, when the mission starts, the UAV node calls its function Follow_Path(), which is actually the core of both the trajectory following and collisions avoidance and is in charge of the control of the mission.
Its role is double:

1. To receive the path of the mission computed by the Path Planner,scanning the path in accordance with the progress of the mission and providing the position of the next goal to be reached. All this work is done by "Follow_Path()" function itself.

2. Optimizing the trajectory, returning the best one to reach the following waypoint. This function is done by "Find_Trajectory()", called by "Follow_Path()".

The optimization part is crucial to finally provide a Traffic Manager capable of controlling its agents avoiding risky situation or, in worst case, collisions. Practically, this led to the need of a proper solver which could be used to obtain the optimal control input according to a proper cost function: in CBUTM, we decided to use NLopt.
NLopt is an open-source library for non-linear optimization, callable from a variety of programming language among which C++, which provides a common interface for a number of different optimization routines [58]. They consist in a standard way to define all aspects relative to an optimization problem, as for example: bound constraints, non-linear constraints, stopping criteria,ecc. In this way, it is possible to impose an higher cost to area of the map as no fly zone (red from the corresponding map layer) or which are in proximity of vehicles with higher priority.
Finally, Nlopt provides the "Optimize()" function, which solves the optimization problems finding the corrects control input for each drone.

With this concepts, the implementation overview ends, since every part of the Cloud-Based UASs Traffic Manager has been depicted from a low level point of view. In the rest of this chapter, only the Map Manager and Path Validator will be further analyzed.

4.4 Risk-Aware Map Manager

4.4.1 Environment-Related Layer

In this section, the ROS nodes related to the map managing will be discussed. Although the code is freely available to everyone, it seemed uselees to report it here in its totality. A better way to explain what's going on inside the Map Manager is instead a mix of pseudo-code and textual reference, that can guarantee still a low level view but without the need of knowing Python or C++.
The first node we developed was "Create_Map_Service" (CMS), in order to answer the demand of a real urban mapping to make the drone fly safe. The aim of this node is to implement a service that, when triggered by the CBUTM, is capable to provide a geographical map at a given height level h. Practically, given a 3D environment, CMS has to operate a cut at the proper value of z-axe, in order to obtain a planar map. Inside this node, two function are implemented: "CreateNoFlyZone" and "CreateCoveredArea". The first's output is the map of all the buildings taller than h, that have to be considered by the UTM has no fly zone to prevent collisions. The second instead works in the opposite way, since it returns a grid map object containing all and just the structures with height less than h. The utility of this object emerges when evaluating the sheltering factor of a given area, since flying over a concrete palace is many times safer than over a street. In pseudo-code, refer to 1. The input of the function are the geographical coordinates of the

Function 1 Create_No_Fly_Zone

Require: Minimum and maximum longitude and latitude, map's resolution, flight altitude h and .osm file's path
Ensure: Creates a layer of a grid_map containing no fly zones
```
GPStoMeter(minlat,minlon,maxlon,maxlat)
loadPolygonFileOBJ(mapdirectory, mesh)
cloud ← mesh.cloud
for i ≤ cloud.point.size do
    if cloud.point.z ≥ h then
        AddPointToMap(cloud.point,NFZmap)
    end if
end for
SaveMap(NFZmap)
```

vertex of the area considered, the directory where is contained the .osm file with the environment in 3D (freely available on OpenStreetMap.org), the resolution and theflight altitude of the drone. First, we have to convert the coordinates from GPS to meter, with the proper function. Then, the .osm file have to be converted in point cloud, in order to be able to manage it. Once this operations are done, every point's z is compared with the flight altitude and, if it's bigger then it, eventually added to the final NFZmap. This object, is a GridMap element initialized with the resolution passed as parameter of the function. Finally, SaveMap provides a .png file of the map in the ROS directory. For what concerns "CreateCoveredArea", it works in the same way but obviously inverting the if condition (see Function 2). It is interesting to notice that both this functions are not drone-related, but instead can be considered general for all the aircraft joining the

Function 2 Create_Covered_Area

Require: Minimum and maximum longitude and latitude, map's resolution, flight altitude h and .osm file's path

Ensure: Creates a layer of a grid_map with the buildings smaller then h

```
GPStoMeter(minlat,minlon,maxlon,maxlat)
loadPolygonFileOBJ(mapdirectory, mesh)
cloud ← mesh.cloud
for i ≤ cloud.point.size do
    if cloud.point.z ≤ h then
        AddPointToMap(cloud.point,NFZmap)
    end if
end for
SaveMap(NFZmap)
```

CBUTM: the maps we obtained (as .png images) are only function of the geography of the scenario. The services provided by "Create_Map_Service" are invoked by the node called "Risk_Map_Node", which is without doubt the core of the map manager: its main aim is to handle the layer of the grid map, providing services to add/ remove them, and to calculate particular kind of layer (as for example the risk one). As said before, each layer of the grid map contains particular information that can be interesting for the risk assessment procedure. In detail, the ones we are considering at the moment in CBUTM's Map Manager are:

no_fly_zone is the layer that contains all the no fly zone of the scenario. Is loaded as a layer from the .png image produced by "Create_Map_Service" node.

covered_area is the layer that contains all thecovered areas of the scenario. Is loaded as a layer from the .png image produced by "Create_Map_Service" node.

pop_density is the layer that contains the number of $\frac{ab}{m^2}$ of each sub-area of the map. For this reason, its upper-bound has been set to 0.0073. We supposed to have an image containing the density distribution, so as before this layer is loaded from a .png file- However, it's possible to take it in a different way, for example starting from a matrix of values.

sheltering_factor is the layer that contains the sheltering factor's value of each sub-area of the map. Its range is within [0,10]. In order to produce this layer,we have to merge the information about the sheltering capabilities of each structure of the environment (contained in a .png file or matrix called sheltering_map), with the covered areas. The result is a simple if piece of code, that to each area associates its proper sheltering value, or the maximum one if there is a building on it:

if $covered\_area$ **then**
  $shletering\_factor = 10$
else
  $shletering\_factor = sheltering\_map$
end if

drone.uav_name +"/signal_layer" this layer contains the signal's QoS for a specific drone, whose name is contained in the label of the layer itself. For our purpose, a generic signal map has been used, but in real operative scenarios is crucial to have this kind of information from the cloud service provider.

drone.uav_name +"/risk_layer" is probably the most important layer of the map, since it contains the risk information of each sub-area for a specific UAV (whose name is contained in the label of the layer). How to build this layer, and the next one, will be described later on more in detail.

drone.uav_name +"/cost_layer" It's the last layer of the grid_map. The map manager weights the two previous ones in order to build the "cost_layer" of the drone, with the techniques exposed in chapter 3. Upon this layer, the Path Planner works to find the best trajectory.

The framework is however expandable if some kind of new data should become interesting in future.
Once the Map Manager is launched, "Risk_Map_Node" starts adding the layers. Firstly, the service provided by "Create_Map_Service" are called and both "no_fly_zone" and "covered_area" begin to form the map. Then, "pop_density" and "sheltering_factor" are added with the logics previously exposed, looking to the images that correctly describe this information.
This part of the work is done every time the ros node is launched, no matter if there are no drones

registered to CBUTM, since this four layer will be the same of everyone. In next section, how to build the remaining layers will be discussed.

4.4.2 UAS-Related Layers

At this point, three different services are defined inside "Risk_Map_Node" in order to provide the instruments for the CBUTM to evaluate the last three layers. In particular, when a registration request is advanced by an Unmanned System to the traffic manager, the "Control Station" node will call in sequence this functions, that will build signal, risk and cost layer for the specific drone. Once again, it is important to focus on this point: the layer which are independent from the drone's parameter are included in the map automatically, without any service and with the procedure exposed before. The ones we are going to see now, instead, are added only when a specific request comes from the "Control Station" node.
As it emerges from the name, this layers are deeply linked with the constructive (and not only) parameters of the UAS: practically, they describes the quality of signal, the risk and the overall cost for an UAV to fly over every area of the map. Upon each layer of a grid map an iterator can be constructed, which moves from a cell to the following one: for every of them the proper analysis must be done.
The first layer built is "drone.uav_name + /signal_layer" through the service "add_cost_layer". This is the simplest among the three UAS-Aware Layers, since no real operation have to be done to calculate it. As said before, it will be mandatory for the cloud operator to provide such a map, that however should be adapted to each drone according to it capabilities (for example antenna's gain) in receiving signals. The range of value is within [0,100], where 0 is lack of signal and 100 maximum quality, in order to be easily compared with risk and cost.
In our case, since there were no map available, a simple scenario has been simulated. Thanks to the "circular_iterator", which is a particular kind of iterator provided by "grid_map" library, we deployed a map with 100% QoS exept for a circle of radius equal to 80 where the Qos starts decrasing till the center which has zero value. Obviously, this is only intended for simulation purpose, and doesn't reflect real TIM's map.
The second layer construction triggered by the UAV's registration process is "drone.uav_name + /risk_layer", which is the core of the whole Map Managing procedure. Inside the service which create it("add_uav_layer"), many technical tricks have been used. In the following list, the most interesting ones are underlined, in chronological order:

1. The layer is firstly initialized with the map dimensions, equal to the one of the previous layers. At this time, each cell is imposed equal to NAN.

2. Evaluation of impact kinetic energy E, using two different formulas according to the type (quadrotor or fixed wing) of drone: $E = m \times g \times h$ for free fall and $E = 0.5 * g * v^2$ where v is the impact speed evaluated with glide angle equal to 45.

3. Calculation of impact area with eq. 2.5

At this point for each sub-area, identified with a classical iterator:

1. Evaluation of sheltering factor and population density. Thanks to the same iterator that access "drone.uav_name + /risk_layer" it is possible to access also the corresponding cell in "pop_density" and "sheltering_factor". Furthermore, we still improved the estimate of this two parameter, introducing a mean between the value of the cell considered and the surrounding ones, in a circular range of radius equal to 10 meters. In this way, when calculating the risk of having a failure in a given area, we also consider the eventuality in which the drone falls in a zone that is near the one where the fail happened.

2. Calculation of cell's risk, with equation 2.2. If "no_fly_zone" layer contains a value different from 0 in the corresponding cell, the risk is imposed equal to f_{max}, according to what discussed in chapter 3.

Finally, the last service called by the registration process is "add_cost_layer, which aim is to implement the concepts about the cost function seen in chapter 3 and build "drone.uav_name + /cost_layer" . It have obviously to be the last layer to add because of its nature: it have to aggregate the numbers contained in signal and cost layer, that for this reason must already exists before start working on it. The first operation performed is this check, thanks to the grid_map function's "exists". Then, on each cell:

1. the corresponding normalized ([0,100]) risk is evaluated, using formula 3.7.

2. Cost value is inserted, using eq. 3.29.

The last thing to do is to make the map readable for the "utm_path_planning" node, so it is published on topic "/risk_map_node/risk_map". That's said, the map managing procedure it's over. Many (the biggest part!) of the implementation issues have not been presented in order to make easier the reading of this part.

4.5 Path Validator

As discussed in chapter 3, the Path Validation is a crucial feature of CBUTM, since it has to measure the overall risk (and not only) of the trajectory and compare it with the required standard (imposed by National Flight Authority, as explained in the flowchart 3.1).
According to figure 4.4, in our framework the node deputed to the Path Planning is called "utm_path_planning", which receives the grid_map on topic "/risk_map_node/risk_map". How the P.P. procedure has been implemented will not treated here, since this is not the scope of this particular thesis. The only things important to know are that the RRT* algorithm (see chapter 3) is the one chosen as Path Planner, and when it ends its task the trajectory is contained in a message of the type nav_msgs::Path: this particular ROS object has been developed precisely for mobile robotics, and contains an array of pose in space that the drone is supposed to follow. In our case, both the z-position and the orientation can be neglected. It's fundamental to notice that not all the poses (that will be infinite!) are included in this message, but only the ones where a change of direction in drone's flight is needed.
Finally, the Path Planner's starting point is contained inside the uav_id message that starts the registration process, while the goal is received through RViz's interface: this solution (that can be easily changed) allowed us to makes a lot of different simulation without changing manually the target position every time.
Once start and goal are known and the map available, the Path Planner finds the lowest cost path that links this two points, working on the cost layer (which values' range is from 0 to 100) of the particular drone that demands the authorization to fly. This last task is the core of the path validation.
Due to its intrinsic characteristics, we decided to develop "path_validation" as a service, which returns true if the path satisfies all its constrains (so, it is valid) and false vice versa. The service is called by the "utm_path_planning" node when the path has been chosen, and before returning to the drone the authorization to fly: for this reason, the path validation is the last part of the registration process.
The work's flow of this service can be divided in three macro areas, which have to be faced in this particular order:

Path Decoder which starts from the Path message and provide as output all the cells where the drone is supposed to fly.

Signal Validation check if the Signal's QoS of that path satisfies the mission's requirements

Risk Validation check if the path's risk is below the threshold.

Although in real code they are in the same function (path_validation), it's easier for us to explain in pseudo code this three parts as divided.
For what concern the path analysis, a particular iterator of grid_map, called LineIterator, is used. Practically, given two points this object automatically interpolates them with a straight line, finding all the cells between them. Once the path has been unwrapped, some information useful for the following parts are collected, as for example the distances between two adjacent cells, the time needed to pass from one to the following and the overall time's length of the flight. In this

Function 3 Path_Decoder

Require: Path's waypoint vector $Path$
Ensure: A set of arrays containing data on all the cells crossed by UAV
newarrays $signal, risk, distances, times$
loadPolygonFileOBJ(mapdirectory, mesh)
$cloud \leftarrow mesh.cloud$
for $i = 0, i < path.length - 1, i++$ **do**
$initial\_pose \leftarrow Path(i)$
$final\_pose \leftarrow Path(i+1)$
for LineIterator li(initial_pose,final_pose), !li.isPastEnd() **do**
$risk \leftarrow risk\_layer(li)$
$signal \leftarrow signal\_layer(li)$
$disances \leftarrow getDistance(cost\_layer(li), cost\_layer(li+1))$
$times \leftarrow distances/cruise\_speed$
$mission\_time \leftarrow distances.sum/cruise_speed$
end for
end for

way, the problem of finding out which cells are overflown is solved and the information that will be used by next functions are provided as vectors, easy to be handled and accessed. Pseudo code in funciton 3
The second "sub-function" we developed is "Signal_Validation", which works on the "signal" vector obained as output of "Path_Decoder". As already discussed in chapter 3, many other function like this one can be developed (using almost the same paradigm) according to the number of parameters of the unmanned mission. The output is just a boolean variable "s_check", which will be true of false if the path satisfies or not this requirement. Pseudo code in function 4
Finally, the risk validation have to be addressed. Once again, it is important to remember that this function must be the last one, since only the risk assessment can decide whether or not the mission is feasible: the output will be a boolean variable, call "validation", which can be considered as the final output of the registration procedure. As discussed in chapter 3, in this function the theory of crash probability have to be implemented (equation 3.41), in order to pass from the risk value in people per hour of flight to expected victims. Moreover, it has to check if a no fly zone is crossed and calculate the overall mission's risk, called "Ntraj". Then, boolean validation is returned.
That said, the function provided by the Map Manager are over. We implemented a part of the CBUTM that now is capable of handling .osm file (real open source map), doing a risk analysis on it and finally finding an optimal trajectory and compare it with the requirements. This conclude

Function 4 Signal_Validation

Require: The trajectory to validate, the signal lower-bound $signal\_lowerbound$
Ensure: A boolean value $check$
$s\_check \leftarrow true$
for $i = 0, i < signal.length, i++$ **do**
if $signal(i) < signal\_lowerbound$ **then**
$s\_check \leftarrow false$
end if
end for

Function 5 Risk_Validation

Require: The trajectory to validate, the risk upper-bound $risk\_upperbound$
Ensure: A boolean value $validation$
$Ntraj \leftarrow 0$
for $i = 0, i < risk.length, i++$ **do**
if $risk(i) > risk\_upperbound$ **then**
$validation \leftarrow false$
end if
$Ntraj \leftarrow Ntraj + risk(i) * MTBF * Pc$
end for
if $Ntraj \leq risk\_upperbound$ **then**
if $check$ **then**
$validation \leftarrow true$
else
$validation \leftarrow false$
end if
else if $Ntraj > risk\_upperbound$ **then**
$validation \leftarrow false$
end if
return validation

the implementation section: in other thesis, also the trajectory following problem and collision avoidance will be discussed deeper.

Chapter 5

Simulations

5.1 Simulation Environment

In this chapter, a complete simulation of a typical Traffic Management scenario will be provided: we will show how the processes described in previous chapters really works, considering not only the Map Managing but also all the other components of the Cloud-Based Traffic Manager, from the registration to the in-flight control.
Before starting the simulation, let's have a quick presentation of the main tools we used in it.
Simulation, testing and debugging have been performed in Gazebo 7, a robot simulation software that will be presented more in detail at the end of this section, running on a ROS Kinetic distribution. Our needing was to have a simulation environment, in which vehicles as much as some on-board sensors (like the GPS transmitter) were simulated, and could send and receive messages and commands through the MAVLink protocol: this one is a communication protocol, very common for Micro Aerial Vehicles and already presents in most of the ones actually commercialized.
For this purpose, Unmanned Capture the Flag (U-CTF, [65]) turned out to be very useful. It is actually nothing more than a game developed in the ROS environment, in which swarms of drones are flown in an Unmanned Capture The Flag match, that can be played both in the real world and in the simulator. It consists of a complex system in which the simulator, the autopilot software and a mission management interface are linked together, built up as provided by 3D Robotics in a software-in-the-loop (SITL) environment.

Drones have been inserted by means of their Unified Robot Description Format (URDF) model, that properly describes their dynamics. This is what has been done from UCTF developers, that used a 3DR Iris quadcopter as simulated drone, together with a scaled down Cessna fixed wing, that has not been used in our work, since for technical characteristics and flight dynamics, does not fit with the purposes of this project. Both these vehicles are equipped with a autopilot software, that, managing all the technical aspects of the flight, allows high-level interaction with the drone. It could be chosen between a PX4 or an ArduPilot-copter, two of the most common autopilot software in commerce: we decided for the latter for convenience. It is a full-featured, open-source multi-copter UAV controller capable of a wide range of flight requirements, which can be programmed through a number of compatible software ground stations. It uses the MAVLink communication protocol ([60]), that is used for transmitting commands and informations between vehicles and the control station. At this point, a Gazebo simulation environment is set up, with certain world and environment models: Gazebo node then, through its plugins, simulates the behavior of inertial measurements units (IMU) and GPS sensors, publishing messages about IMU, GPS position and GPS velocity value on specific topics. This SITL can be spawned in multiple instances, modelling multiple different copters to exist at once, allowing us to run at least three simulated drones on a

reasonably powered laptop.
We then used as interface the QGroundControl[61] software, a powerful Ground Control Station which provides full flight control and mission planning for any MAVLink-enabled PX4- and ArduPilot-powered UAVs. It has been configured according to the Software-In-The-Loop configuration, following the general guidelines, making it read from port 14000, where the simulated drones are communicating via MAVLink. Its usage has been fundamental for a preliminary understanding of how the autopilot works and how to interact with it, while, once understood the communication's dynamics, it has been put aside, and the system extended to allow communication between our autonomy package and the autopilot, creating proper nodes to send and receive messages and manage missions.
Last software involved in the simulation environment setup is Rviz, a 3D visualizer for the Robot Operating System framework, already presented in 4.

5.1.1 Gazebo

Gazebo is an essential tool for developing robotics applications. It is free and supported by a great community, making of gazebo one of the primary tools used by the ROS developers [59]. It consists of a robust physics engine, high-quality graphics, and convenient programmatic and graphical interface, offering the ability to accurately and efficiently simulate robots in a complex scenario, with everything being customizable according to the needing. By means of proper world files is, in fact, possible to describe all the elements making up the scenario, while with model files is possible to model any kind of robot. This is done in a specific format, called SDF, Simulation Description File, even tough it is still possible to find robot model described in the old URDF (Universal RObot Descriptio File) file, left apart because of the evolving needs of robotics.
SDF is a complete description for everything from the world level down to the robot level. It is described using XML, is scalable, self-descriptive and makes it easy to add and modify elements.
The Gazebo server, the engine of the simulator, parses the description files given to it as input, building up the simulation environment accordingly and simulating dynamics thanks to multiple high-performance physics engines. It is shown thanks to the Gazebo graphical client, that connects to a running gazebo server allowing its visualization. It utilizes OGRE's graphics rendering engine, providing realistic rendering of 3D scenarios. Gazebo uses then a number of enviroment variables to locate files, and set up the communications between servers and clients.
Finally, many plugins offered within the Gazebo community provide a simple and convenient mechanism to interface and interact with it, having direct access to Gazebo's API and all its functionalities. They are self-contained routines that can be added or removed from running systems, allowing to include new features like, for example, the simulated GPS sensors used in the present work.

5.1.2 Distributed Architecture

Introducing ROS we said how it is explicitly designed for distributed computing. Here we will see how this possibility can actually be put in place, allowing us at the same time to show a simulative context more coherent with the Cloud infrastructure we designed in this project.
The way in which the processes are managed by means of the nodes, and the methods of communication between them, analyzed in section 4, make it possible not only to allocate these processes on different calculation units, but it is also possible to relocate the same processes at run-time to match the available resources in the network.
This last possibility is very interesting as it is suitable for a new emerging technological paradigm: edge computing.
This concept basically consists in revising the cloud infrastructures considering the changes that

the internet of things (IoT) implies on the way of perceiving the internet itself. In fact, for years now, the internet has ceased to be a virtual and intangible entity. In the age of IoT, we already see it, Internet is not only present and directly affects many of the activities that each one performs every day (appliances, the way we communicate or move or prepare food) but ends up generating the knowledge itself, starting from the information that such devices collect to work.
This in some way shifts the "center of gravity" of internet itself: from large data centers thousands of kilometers away, the measurements and the actuations of internet take place and will take place practically everywhere.
Edge computing is exactly the paradigm for which the computation moves close to the source of the data it collects and close to the actions it determines.
This paradigm is perfectly suited to being associated with 5G, which is "a collective name for technologies and methods that would go into the future networks" [62]. Given the importance of cloud services also on mobile networks, the concept of Mobile Edge Computing (MEC) is recognized as one of the key emerging technologies for 5G networks and it is therefore something that we will deal with in the future.
Distributing the computation with ROS is very easy. It is required that each pair of machine have a complete and bi-directional connectivity on all the port then, assuming that we want to distribute the calculation only on two machines, we proceed to configure them as shown below:

```
Machine1:
$ export ROS_HOSTNAME=Machine1
$ export ROS_MASTER_URI=http://Machine1:11311
$ export ROS_IP=machine_1_ip

Machine2:
$ export ROS_HOSTNAME=Machine1
$ export ROS_MASTER_URI=http://machine_1_ip:11311
$ export ROS_IP=machine_2_ip
```

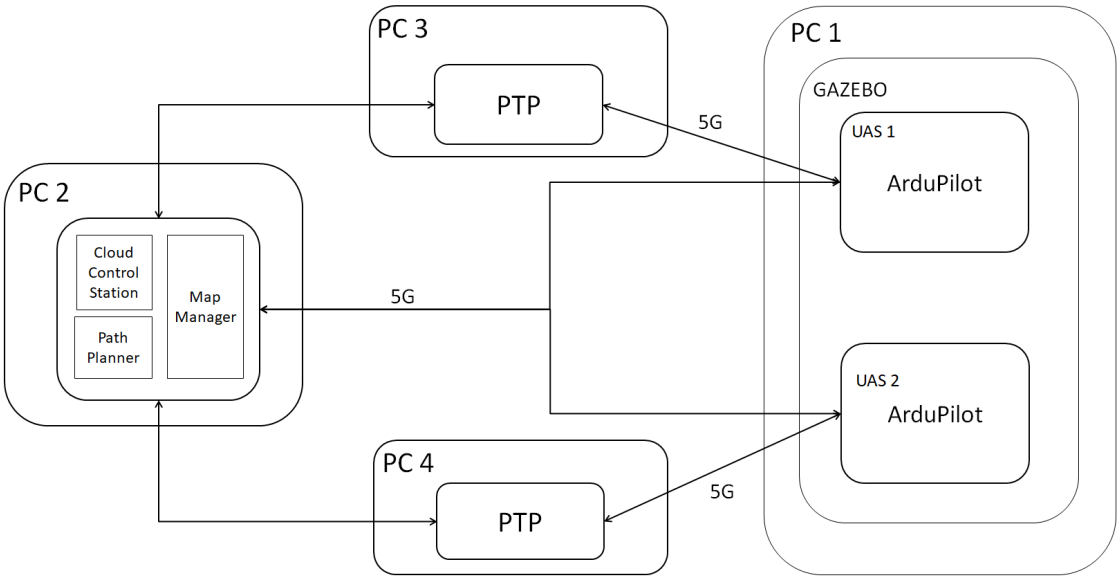


Figure 5.1: Architecture of the distributed simulation environment

Using an architecture like the one in figure 5.1, it is possible to perform simulations more coherent with the real context of application. With such a configuration, it is possible to experiment the distributed architecture and above all it is possible to introduce a certain latency in the communication between the nodes. The latter was not considered in the course of the project and of the simulations but it is possible to make however some considerations.
First we look at the criterion with which we assigned the nodes to the machines. PC 1 simulates the real world: inside we have Gazebo and therefore two simulated drones with ArduPilot on-board. Notice how here we refer to only two UASs, but clearly the discourse can be extended and then actually applied to an arbitrary number of unmanned systems. All connections with PC 1 simulate 5G connections between vehicles and the cloud. The latency specifications of the next generation of mobile network are not yet perfectly known, but we know that they should be in the order of milliseconds.
It can be assumed that in general the latency of a network varies according to a normal distribution. Let's see, with the following command, how we can add to PC 1 a normally distributed delay with a mean value of 50ms and a variance of 20ms

```
# tc qdisc change dev wlan0 root netem delay 50ms
20ms distribution normal
```

These latency values should not affect the functioning of the architecture proposed here. In fact, the frequency of the nodes involved in CBUTM never exceeds 5Hz.
PCs 3 and 4, as seen, execute the nodes related to the trajectory following; PC 2 all other CBUTM's nodes.
A consideration of the connections between PC 2 and PCs 3 and 4 can be traced back to the previous discussion on edge computing. If we assume CBUTM as a centralized entity, PCs 3 and 4 are nothing more than virtual machines within which the nodes are executed. In this case the connection will simulate the connection between the virtual machines, with extremely low latency. On the other hand, if we assume to apply the paradigm of edge computing, PCs 3 and 4 can be considered as independent units of calculation allocated near the mission area. In this case the latency will depend on the way in which these machines are connected, and we can then use the previous command on these machines too, assuming here also a communication based on 5G technology.

5.2 Map Manager

5.2.1 Environmental Modelling

According to the concepts exposed in previous chapters, the first output that the Risk-Aware Map Manager has to produce its a coherent model of the environment in which the mission takes place. In particular, it has to provide two UAV-independent layers, which will be the same for every drone that will fly (at the same height h, obviously) in this area: "no_fly_zone" and "covered_area". Their role, already explained in chapter 4, is to describe two crucial features of the "world", that for a planar robot (as the drone) is a simple "cut" of the geographical map at the flight's altitude. On one hand, there are all the buildings taller than this value, that for this reasons can be a danger for the Unmanned Aircraft and have to be precisely identified. On the other, all the structures lower than h that can offer a very good sheltering for people on ground, in case of UAV's fall.
De to the high computational cost that such simulation requires, we decided to not consider an entire city, but just a part of it, that can provides some particular features to show the capability of CBUTM. The area we chose is the one contained in the rectangle defined by:

$$[min\ latitude, max\ latitude] = [45.0645000, 45.0741000] \tag{5.1}$$

$$[min\ longitude, max\ longitude] = [7.6537000, 7.6732000] \tag{5.2}$$

which is the part of Turin around the San Paolo's skyscraper. In figure 5.2, the view from osm-buildings.com is depicted. Obviously, the choice of this area is not casual, but has some criticisms

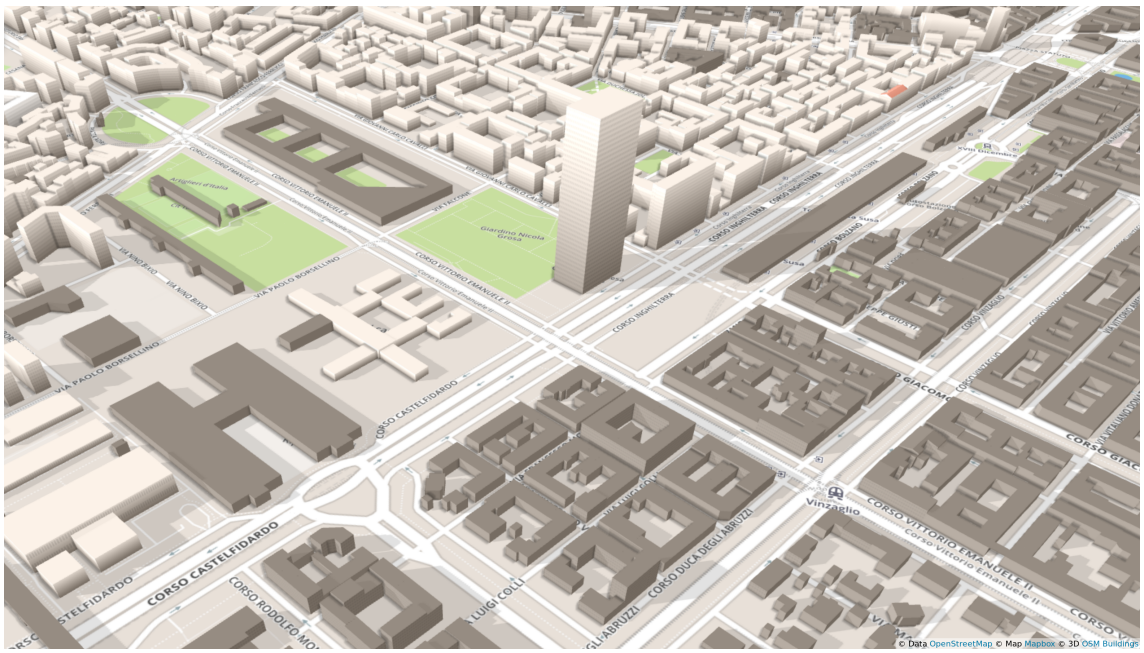


Figure 5.2: Open Street Map Buildings View

due to the presence of an over 100 meter building.
Starting from the corresponding .osm file, the input of the "Create_Map" service is a point cloud, which can be obtained through specific open source softwares. The result is the one in figure 5.3.
Saving it as .obj file, it is finally the proper input for th modeling environment function. Besides this, also other information are needed to compute the layers: height of flight and map's resolution. In this simulation case, we decided to impose $h = 25m$ and a resolution of $25m^2$, which both are in line we real operative situation. Converting the geographical coordinates in pure lengths, we obtain a map of $1535 \times 1070m$, that whit this resolution's value correspond to 307×214 cells.
Applying a cut at $h = 25m$ we obtain the first two layers of the map (see fig 5.4 and 5.5). What is interesting of this two layers is their complementarity: merging the two figure we obtain the complete representation of the operative environment. In particular, in the first one, only the skyscraper and the building next to it are depicted. As already said in this thesis, it's not very common to find tall palaces in city center. For the same reason, the covered areas layer is very dense, since there are a lot of "small" structures, especially in high densely populated zone as this one.
Two other layers are needed to complete the mapping of the environment: population density and sheltering factor. For what concerns the first, we set a constant value of $0.0073\frac{ab}{m^2}$, that means a flat layer (that for this reason will not be reported here). Finally, the sheltering has been modeled as maximum (10) in the zones corresponding to the covered areas, or a generic mean value (6) in all the other parts. In figure, it has the same shape of 5.5, but with different values.
Finally, the environment's model is completed. It is structured in layers and published every 2 seconds on "risk_map_node/risk_map" topic.

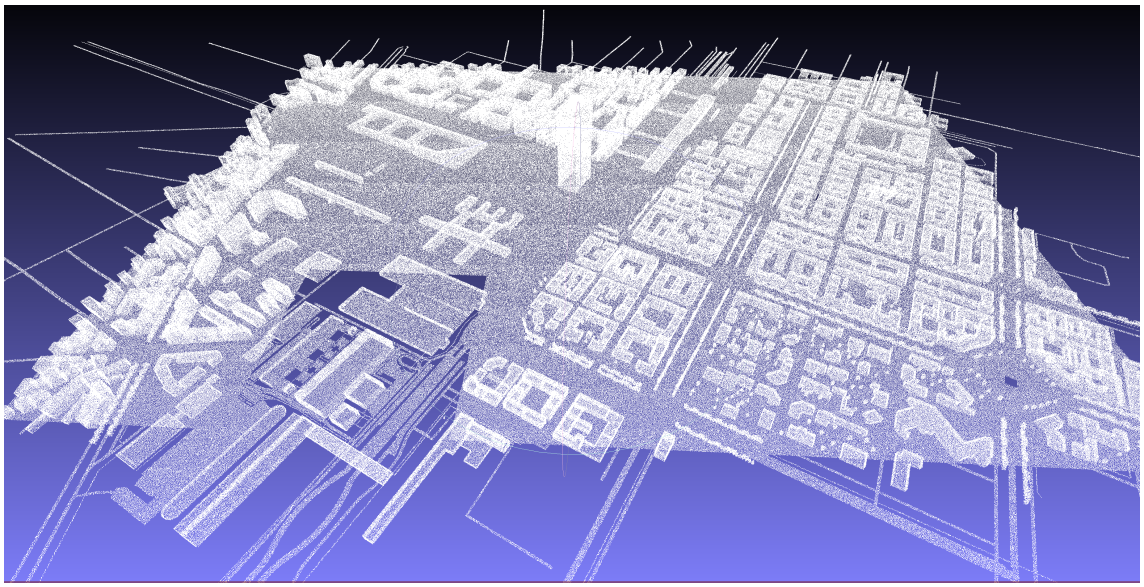


Figure 5.3: Point Cloud Model

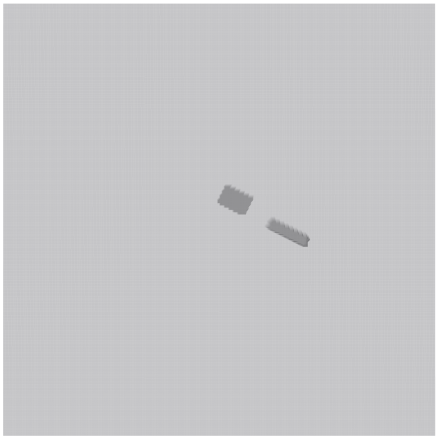


Figure 5.4: No Fly Zone Layer

5.2.2 UAV-Aware Risk Map

Once the model of the environment is available, is possible to accomplish to the registering procedure advanced by the UAVs. In order to correctly do this, its necessary to merge the information present in the first four layers with the drone's characteristics. As discussed previously in this thesis, three layers are UAV-Aware:

- Signal Layer, which represents the capability of the drone to receive Internet connection, in

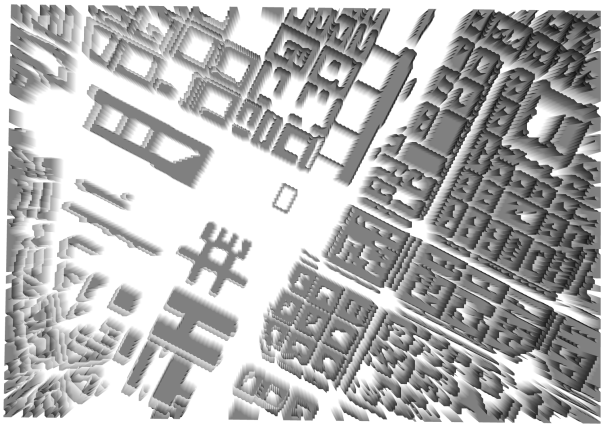


Figure 5.5: Covered Areas Layer

every area.

- Risk Layer, which represents the risk for the UAV, evaluated with equation 2.18.
- Cost Layer, which merges the first two according to proper weights.

In this simulation, the drones are supposed to have a length equal to 0.7 meters and a weight of 0.5 kilograms.
For what concerns the first layer, it is a generic measure of the Quality of Service of the Internet Connection, and is used just as example to show the potentiality of the cost layer. In any case, during this simulation we assume every drone has on board a 5G compliant antenna, to receive TIM's signal. Furthermore, in order to have some criticism, we supposed to have an optimal signal reception, except for a circular area, as in figure 5.6. This is not strictly true, in particular in urban environment where this kind of infrastructure are very powerful. However, it's useful for a better result of the simulation. According to the metric, the layer's value is always 100 except for the cone, where it goes down till 0.
Beside, also the Risk Layer of the drone can be computed. According to what we said in chapter 3, it follows the rule (for every cell):

if no_fly_zone **then**
  $Risk \leftarrow upperbound$
else
  $Risk \leftarrow (N \times P(f|e) \times \lambda)$
end if

In this case, as for the rest of the work, we impose $upperbound = f_{max} = 10^{-5} \frac{victims}{h}$. This layer is probably the most important one, since the risk assessment is a crucial procedure in our framework. It merges all the information present in the other layers, and the results have to be consistent wit real empirical results. Finally, normalizing the risk in range [0,100], we obtain the risk map in figure 5.7.

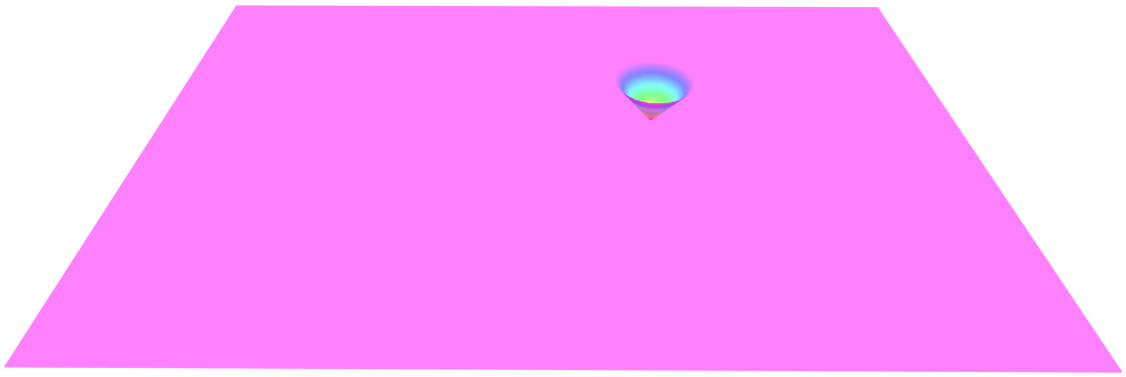


Figure 5.6: Signal QoS Layer

Three kind of areas emerges clearly from the figure:

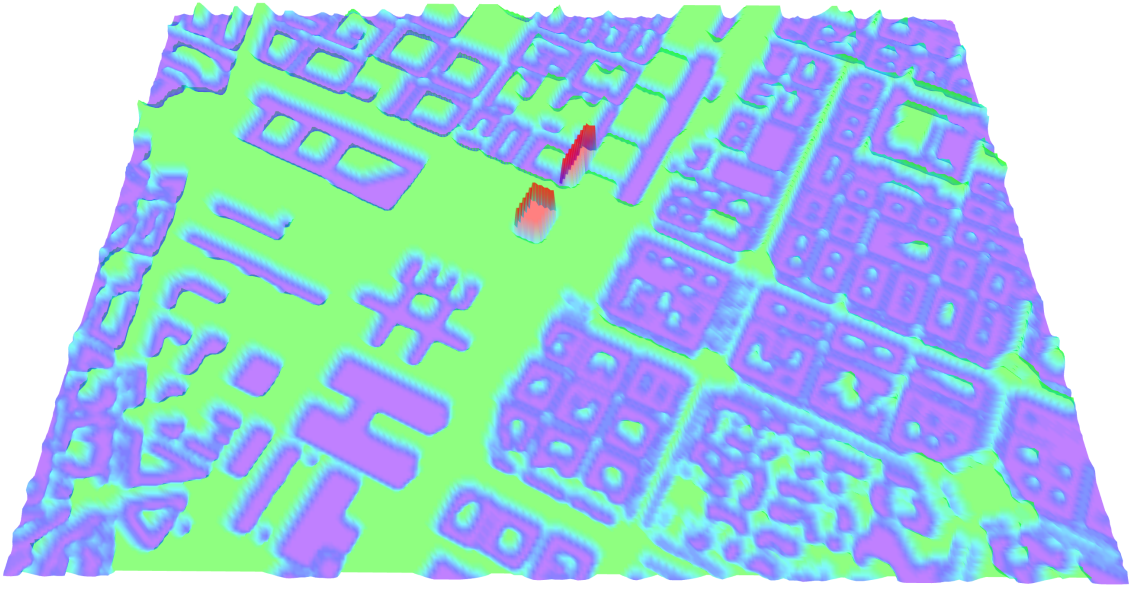


Figure 5.7: Risk Layer

- Areas with the lowest value of sheltering (6), which has a green color in figure. Their risk is equal to 7×10^{-6}, which normalized is 54.
- Areas with the maximum sheltering possible (10). They have a purple color ad the lowest risk $f_F = 5 \times 10^{-6}$, that normalized become 19
- No-Fly Zone, which are red and have the higher risk $f_F = 1 \times 10^{-5}$ that corresponds to 100.

Obviously, many other values are present in the risk maps, but in a rough view they can be summarized in this three categories.

Since both risk and signal layer are in range [0,100], they can be summed in a final layer, called Cost Layer. According to the theoretical concept exposed about it, this last map can be seen as a weighted sum of the other two layers, obtained using coefficients α and $\beta = 1 - \alpha$. In particular, when $\alpha = 1$, only the risk is considered, and the resulting map is the same as in figure 5.7. Instead, when $\alpha = 0$, only signal's QoS affects the cost, and the layer turns out to be the opposite of the one in figure 5.6. An example with $\alpha = 0.5$ is provided in figure 5.8.

In particular, it's possible to notice the no-fly zones and the area of no signal coverage, which in

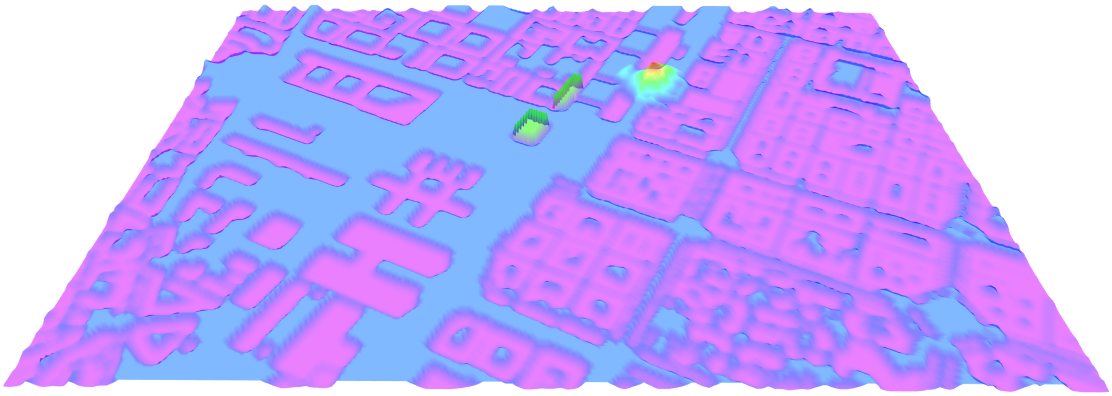


Figure 5.8: Cost Layer

this case is no more an "hole" but instead a cone. This behavior is due to the fact that a low level of signal corresponds to an high cost.

Finally, the Risk-Aware Map Manager ends its work, providing a set of layers common for every drone, and three different layers specific for each aircraft. Upon the last one, the Path Planner will evaluate the better trajectory to perform.

5.3 Path Planning and Validation

The map, understood as a collection of layer, is published on topic "/risk_map_node/risk_map" with a frequency of 0.5 Hz. Every time a new layer is available, it is simply added above the others. In particular, when the Cost Layer of a drone is completed, it is possible for the CBUTM to start the mission planning phase. In this part, we suppose to have a known start position, and to receive the goal of the mission as Rviz input. Each UAV registered to the cloud traffic manager can provide its own end position, different from the other ones and on its specific layer.

Main aim of the Path Planner is to choose the less costly path, from point A to B, and then to validate it through the Path Validation procedure already described in previous section. Since for now we are not considering collisions with other aircrafts, its possible to show the capabilities of this two algorithms just on a single drone, since it will have the same behavior on the others. In figure 5.9 a typical path planning situation is shown, including also some criticisms due to the presence of the skyscraper. The circle in red its the Unmanned Aerial System in its starting pose, while the three axis frame is its final pose. In between, both the no-fly zones and the "signal hole" are present, and easily identifiable.

What emerges its the Path Planner capability in working also in very small spaces: for example, the drone's trajectory is very close to the buildings but never hit them. Since we are also considering

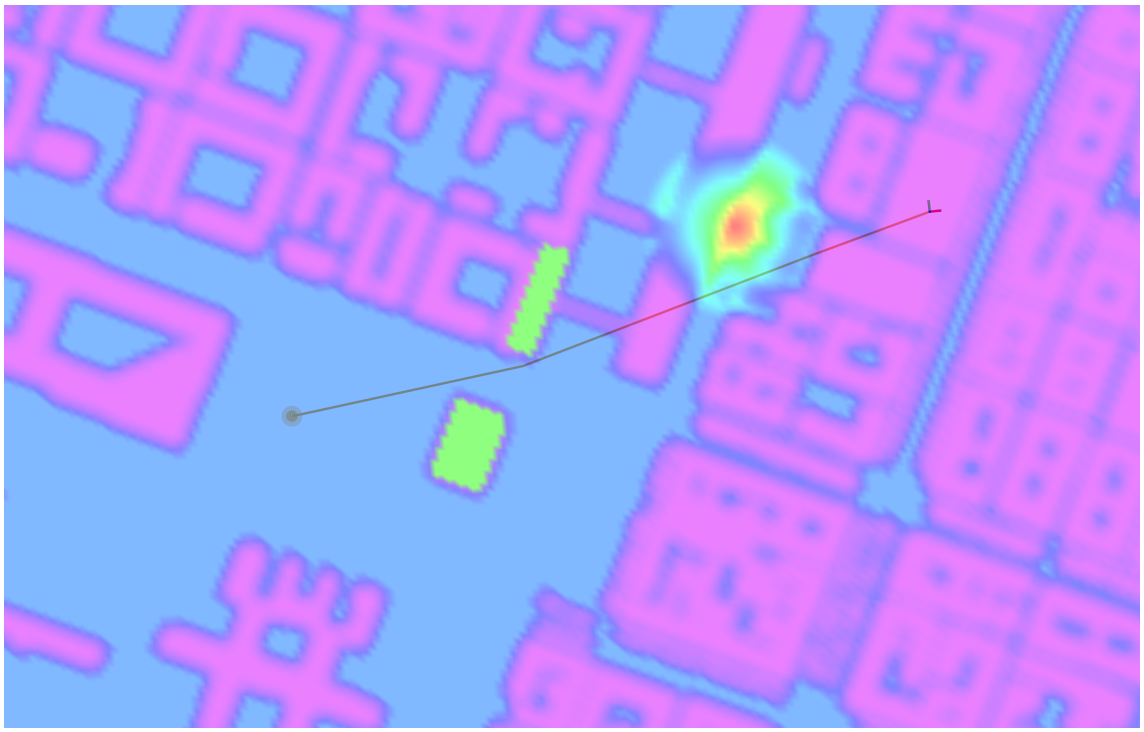


Figure 5.9: Path Planning Example

an inflation area around them, we are almost sure that no collision will happen. Furthermore, the feasibility of the path for what concerns the aircraft's dimensions it's guaranteed by the cells area, which is big enough to allow any kind of maneuvers.
Once the path has been found, the Path Validation Service is called.
The first step to perform is the path decoding, where some useful information are extracted. For this particular simulation, we are assuming a constant cruise speed for the drone equal to $3m/s$ ($10km/h$). Founding out all the cells that belong to the trajectory, its overall length can be measured and is equal to $1008m$. From this value, also the time length of the mission can be calculated, as: $\tau_{miss} = \frac{length}{speed} = 336s$.
At this point, the signal's QOS validation can starts.
As we seen previously, once the path's cells are known this is a quite simple operation, since the validator only has to verify that each cell has a QOS value bigger than the lower-bound, which in this case is set to $S_{min} = 50$. From image 5.9, we can see that the path covers for almost all the time areas with signal's QOS maximum value. Zooming around the hole, it can be verified that a little part of it has a lower value, which however is never below 50. For this reason, the signal quality validation can be considered successfully completed.
Finally, a risk-aware validation have to be performed. First of all, it have to check the presence along the path of no-fly zone: as it's possible to see in figure, none of them is crossed during the flight. Furthermore, it have to ensure that the predicted number of victims of the trajectory N_{traj} is not greater than the maximum acceptable N_{Max}. Since:

$$N_{Max} = f_{F,Max} \times \tau_{miss} = 0.00336 \geq N_{traj} = 0.001239 \tag{5.3}$$

The risk validation procedure is successfully completed.
Since the path is valid from all the point of views, the drone is authorized to start the mission,

setting to true the boolean flag $authorization\_to\_fly$.

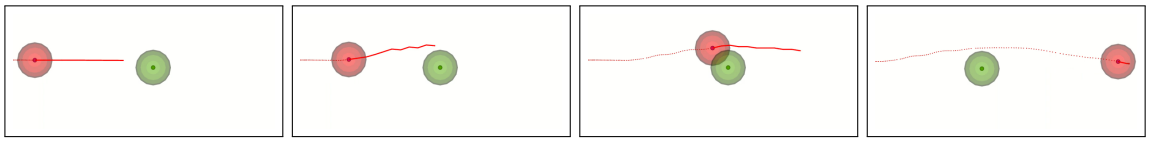
5.4 Path Following and Collision Avoidance

A set of tests and simulations has been performed to analyse the behaviour of the cloud-based traffic management during the flight of the aircraft, in order to validate our approach. For this reason, CBUTM has been seen in action in several operational situations, and the obtained results will be here presented. Since our collision avoidance algorithm is priority-based, meaning that it is only and always the lower priority drone avoiding the higher priority one, for an easier understanding the following colours convention will be applied:

- red: lowest priority level drone
- orange: middle priority level drone
- green: highest priority level drone

The following sequences of screen-shots come from Rviz, where vehicles are represented with a circle (a $geometry\_msgs :: PoinStamped$), surrounded by their own safety area. This is the forbidden area around every drone, where other drones are not allowed to fly, and it is to be considered as a safety distance marker of 10 metres. Output of the collision avoidance algorithm is the local trajectory the autopilot will follow, that is shown with a thicker line coming out of the drone, while the dashed line the UAV leaves behind is its odometry trace. Buildings and fixed obstacles are represented in a certain colour scale on the map: in the sequences below, the map used is built on the no_fly_zone layer.

The first test is a basic one, where 2 unmanned vehicles, regularly registered within the network, are flying in the same area, too close one to the other. In fig 5.10, the first test sequence is shown. In this case, only one drone is flying - the *red* one - while the other, with higher priority level, is standing in its position, hovering. As supposed, the red drone is the one forced to move from its original path, as soon as the other drone is detected in 5.10b. The *red* drone starts slightly moving out of its path, going around the *green* one, and finally going back to its original path, and reaching its goal.

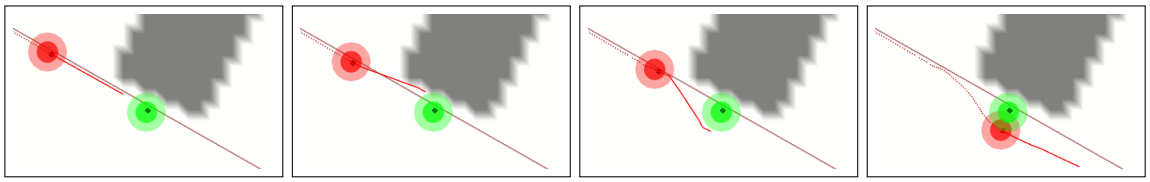


(a) (b) (c) (d)

Figure 5.10: Test 1: 2 UAVs, only one moving

The same situation in the neighbourhood of a $noflyzone$ represented by a building is shown in Figure 5.11. In this case, the *red* US, when looking for a way to escape from collision, tries to go left-ward first, being that one the shortest way to she goal (5.11b). When the presence of the building is detected (5.11b), it immediately changes direction, overcoming the *green* obstacle going right (5.11c), and then coming back to the original path.

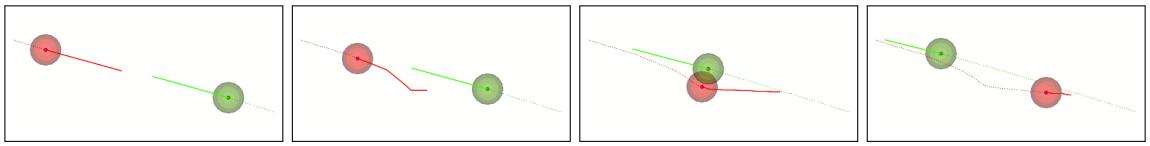
From now on, at least two vehicles will be in motion at the same time. The first, basic case involves two drones flying one against the other, shown in the sequence in Figure 5.12. The time requirements of this case are a bit more tight with respect to the previous one, since both the drones are flying at their cruise speed of about $4m/s$. The *red* drone then, once detected the *green* one in 5.12b, must be faster than before in changing trajectory. This is done in the optimal



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

Figure 5.11: Test 2: 2 drones, only 1 moving. Presence of a fixed obstacle

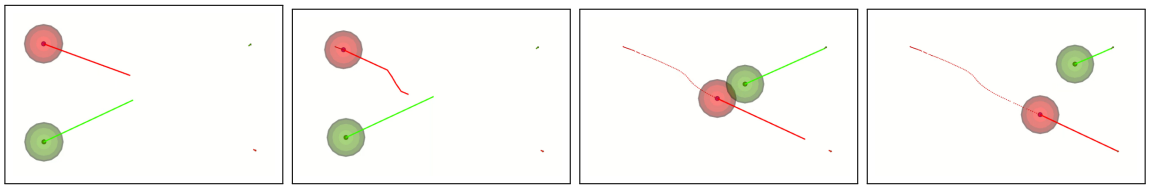
way shown in 5.12c, so that both the vehicles can safely accomplish their mission according to their priorities.



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

Figure 5.12: Test 3: 2 drones, both moving. Presence of a fixed obstacle

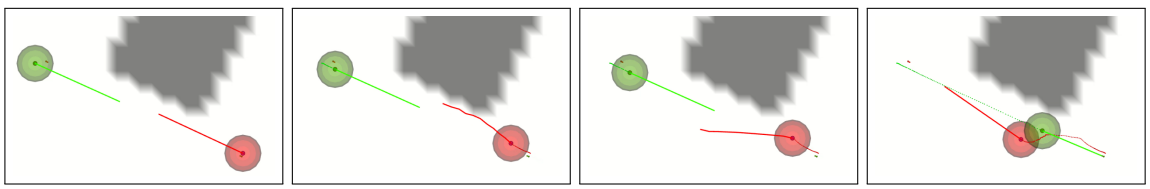
A slightly different circumstance is presented in the sequence 5.13 , where the two vehicles are crossing their paths in a diagonal way.



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

Figure 5.13: Test 3: 2 drones, both moving. Presence of a fixed obstacle

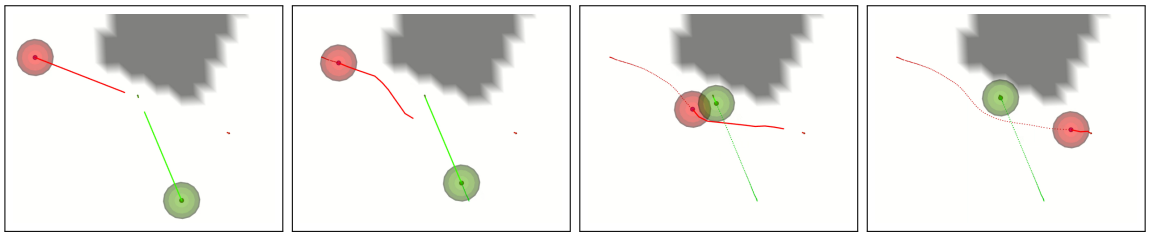
A bit more complex is the situation presented in 5.14, where both the drone are completing their own mission, and their paths cross in front of a building. The *green* one, having an higher priority level, keeps going in its direction, will the *red* one must again fly away from its path. The operation is not performed along the shortest way because of the building, detected in 5.14b, forcing the vehicle to go in the other direction, as can be seen in figure 5.14c.



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

Figure 5.14: Test 5: 2 drones, both moving one toward the other, in presence of a fixed obstacle

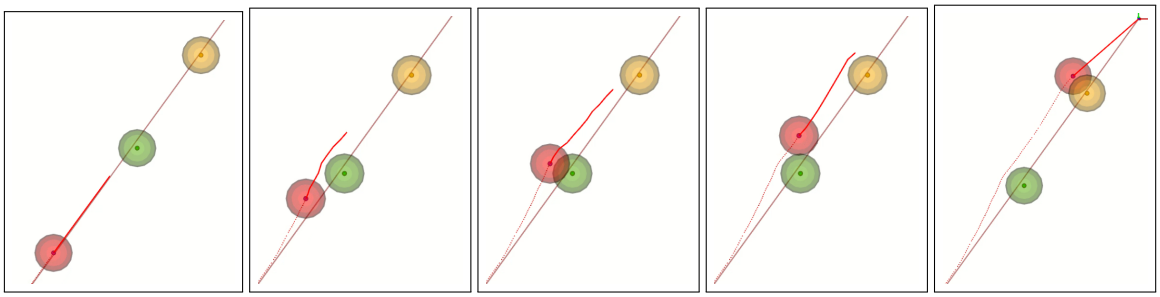
Sequence 5.15 shows a similar situation, with different dynamics: here, the *green* vehicle is approaching the building, and its mission consists in hovering on a fixed position in front of it. In the same moment, the *red* drone is coming from the left-hand side, passing next to the building without stopping by. When it detects the higher priority UAV arriving, the Predictive Trajectory Planner (PTP) computes the optimal deviation from its original path, predicting to go all around the safety area of the *green* one (5.15b).



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

Figure 5.15: Test 6: 2 drones, both moving, crossing their path, in presence of a fixed obstacle

Adding one more vehicle, the situation gets a bit more tricky, but same well managed by the CBUTM system. In Figure5.16, the *red* vehicle coming from below finds two standing vehicles on its path. When the first one has been detected in 5.16b, the vehicle starts moving left to avoid it. Once it has been overcome, while trying to get back on its original path (5.16c), the *red* drone comes across the second higher priority vehicle, and changes again its trajectory to avoid that one too, as can be seen in 5.16d.



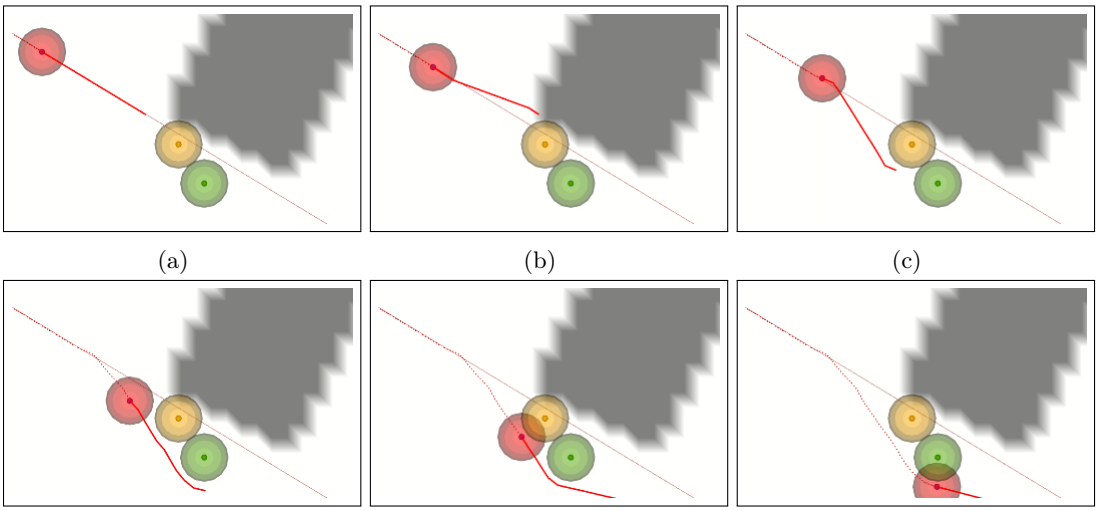
(a) (b) (c) (d) (e)

Figure 5.16: Test 7: 3 drones, one moving

Similar situation, with the only one moving drone meting the other two standing, is shown next (fig:5.17) this time in presence of a fixed obstacle. The same considerations done so far can be applied here.

In the next tests, both the *red* and the *yellow* drones are moving, while the green is standing in its position. A first case is represented in the sequence 5.18, where the *red* and *yellow* vehicles are moving one against the other, while the *green* one is hovering in the middle.

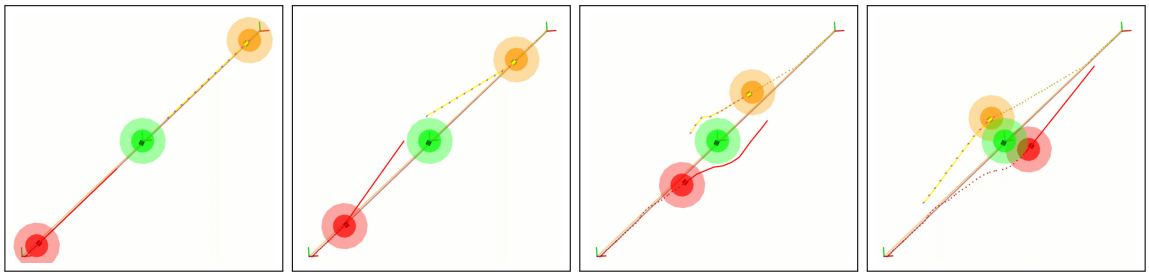
The first two vehicles are flying in different directions (5.19a), and both will meet someone with higher priority on their path, forcing them to change trajectory (5.19b). While doing it, the *red* will also come across the *yellow*, being forced to find another way. It solves the situation computing the path in a way that some way-points are in the same position or close, resulting in a deceleration of the vehicle, as can be noticed looking at its trajectory projection, that becomes



(a) (b) (c)

(d) (e) (f)

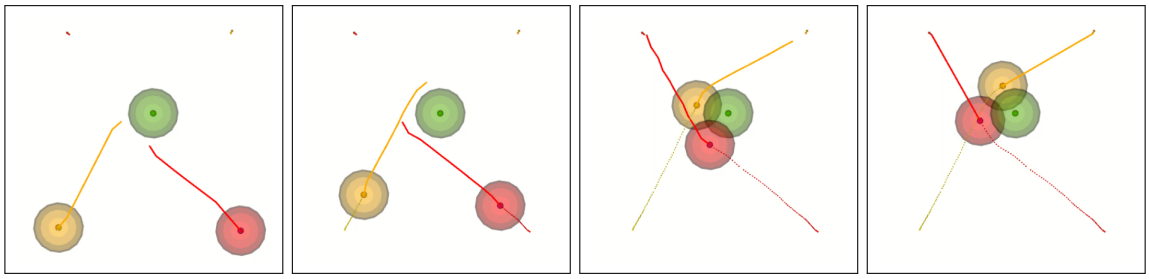
Figure 5.17: Test 8: 3 drones, one moving, in presence of an obstacle



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

Figure 5.18: Test 9: 3 drones, two moving, 1 standing in the middle

shorter than usual (5.19c). After the *yellow* vehicle will be passed by, the way will be free for the *red* one, that keeps going, knowing the path is safe.

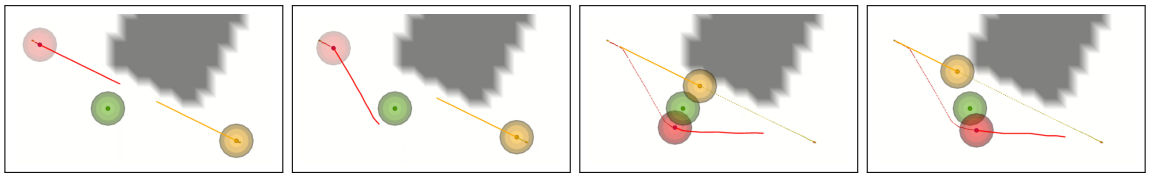


(a) (b) (c) (d)

Figure 5.19: Test 10: 3 drones, two moving in different directions, one hovering

Again, we tested this intersection between three vehicles putting them close to a building.

The result is shown in sequence 5.20. The two vehicles with lower priority are flying in opposite direction (5.20a), until they will get caught between the highest priority one and a fixed obstacle. When pointing toward the space in between the *green* drone and the building, not large enough for both of them, the *yellow* has the precedence, while the *red* one searches for another way, finding at its right-hand side (5.20b). They keep then flying their missions.

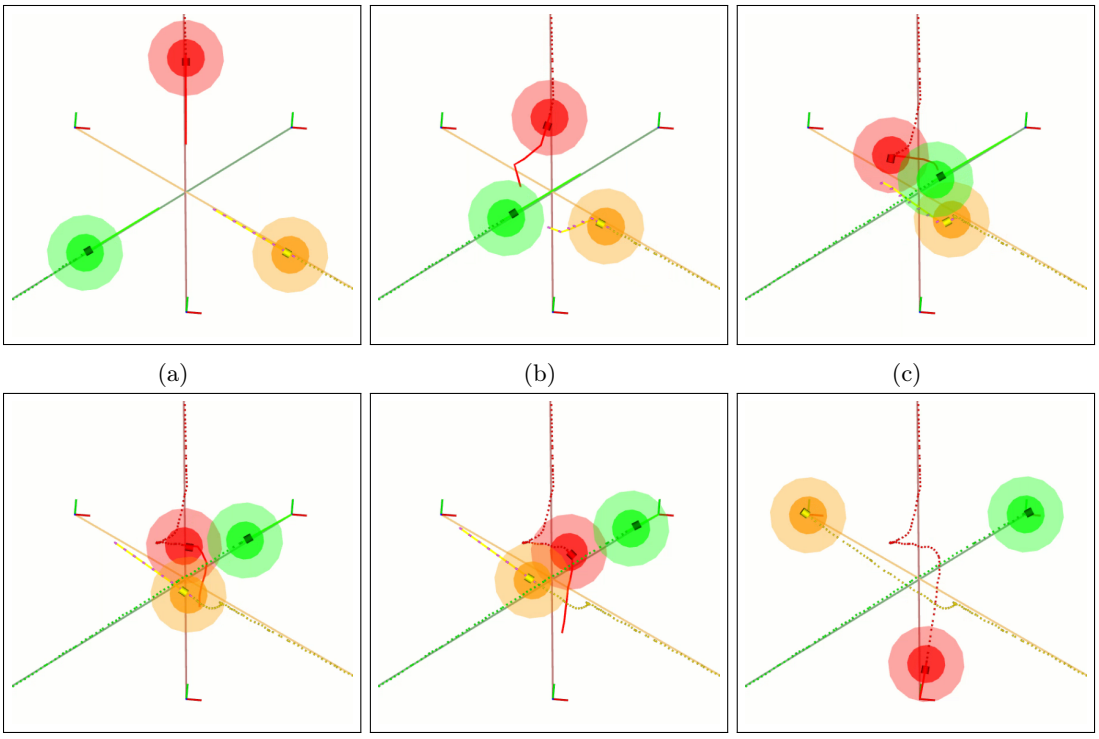


(a) (b) (c) (d)

Figure 5.20: Test 11: 3 drones, two moving, 1 standing, in presence of a fixed obstacle

Last interesting test is shown in the sequence 5.21. Two vehicles with lower priority meet the *green* one, and start computing a different trajectory, that would bring them to move in the same direction (5.21b). When the *red* one detects the *yellow*, it understands it must re-change the plan, and so it does, after tending towards the opposite direction to avoid both the *yellow* and the *green* vehicle (5.21c). In this operation, the predicted trajectory of the *red* drone is made up of way-points which are closer one to each other, resulting in a deceleration of the motion, that allows it to fly just behind the *green* one maintaining the safety distance (5.21d). The priority order is then respected during the crossing, and the safety distance maintained, allowing all the drones to reach their goal with a small deviation.

That's said, the simulations end. In this chapter we showed the software environment we used, and then the all the most important functionalities of the Cloud-Based Traffic Management, treating not only the Risk-Aware Map Manager, but also the Path Planner, the Path Validator and the Collision Avoidance algorithms. What emerges is that the framework is already operative, and it is able to correctly manage not only an ideal operation, but also criticisms during flight.



(a) (b) (c)

(d) (e) (f)

Figure 5.21: Test 12: 3 drones moving in different directions

Chapter 6

Conclusions and Future Works

The need of a completely new UASs Traffic Manager to ensure safety for people on ground has been discussed in deep during this thesis, underlining the features it must guarantee and the issues that arises during the design procedure. Moreover, due to the peculiar characteristics of the urban environment and its high degree of complexity, also the cloud-based technology has been treated, showing how and why it can be used to realize a more effective Cloud-Based UASs Traffic Management.
In particular in this work some remarkable results has been obtained, that are on one hand innovative, while on the other provide for the first time a complete picture of the state of the art of the risk analysis for Unmanned Mission, together with a possible way to use it inside a Traffic Manager. For the first time, a Risk-Aware Map Manager has been designed, capable to produce coherent risk maps, which are without doubt crucial for the future development of the Unmanned Aviation.
Instead of proposing new methods for risk assessment of the vehicle itself, that would impose a study of the aircraft in deep starting from its smaller components, we assumed this part of the work already done by the UAVs producers. In this way, we moved to a risk assessment of a geographical area more than an aircraft, really evaluating the risk for one or more people to be killed or seriously injured by a drone crash (chapter 2). The results has been used by the Risk-Aware Map Manager to produce a completely new idea of map (the risk map), which no longer takes into account only the geographical elements (and obstacles), but all the characteristics that can affects the risk itself. Given a drone, with its peculiar constructive parameters, it will be authorized to fly if and only if a safe enough trajectory can be found for it.
To realize this, we developed in parallel a theoretical background and its implementation (chapter 4), realized in an open source way (ROS) and freely available, modifiable and expandable by everyone. In detail, a new risk metric has been proposed, merging together different researches in this fields with new concepts, as for example the sheltering evaluation. Moreover, the risk assessment procedure has been inserted in a more complex scenario, where also environmental and legislative parameters had to be consider. The result of this operation is a Map Manager capable of merging information, updating and finally building a risk map, with a given resolution function of the cloud computing capability but also of the manoeuvrability of the drone itself (chapter 3). The framework proposed is also modular and multi purpose, in the sense that it can be used in any scenario, for any kind of drone: the more accurate the information it receives are, the more performing the results.
Upon the risk map, handled and update by the RAMM, also a set of Path Planning algorithm has been studied and presented, with a particular attention to Optimal RRT (or RRT*) which is the one implemented actually in CBUTM. Finally, we moved from the risk assessment of a cell of

the risk map, to propose a completely new metric for the risk of a mission (or a trajectory). The importance of this step is crucial, since it allows the Traffic Manager to choose between paths, evaluating which one is the best and if it can be considered compliant with the standard imposed by the national flight authority. This part of the work, called Path Validation, has been discussed deeply and then implemented (chapters 3 and 4).
Inside the same project, all the other aspects that characterize a UASs Traffic Manager, that in this thesis have only been presented, has been deepened and discussed. In particular, according to [63] and [64], a complete registering and monitoring procedure has been developed, to guarantee a constant tracking of all the aircrafts flying in a given moment and to ensure the capability of the Traffic Manager to detect malicious drones. Furthermore a Predictive Trajectory Planner, minimizing a proper cost function, is in charge of handling each drone's dynamic, making it capable to follow the pre-calculated waypoints and, if needed, to avoid collisions both with static and dynamic obstacles.
All this results have been not only discussed, but also tested and simulated, as in chapter 5.
Once again, I want to stress the importance of team work to realize a so complex infrastructure. Everything here presented was made possible through a continuous exchange of knowledge, methods and support, both scientific and human.
Since this is a thesis work, it wasn't possible to complete everything and we really hope someone next will finish it. In particular, some improvement can be done to the Risk-Aware Traffic Manager to better exploit the cloud network. First of all, it is crucial to study and implement a set of algorithms to make the risk map really dynamic (as discussed in chapter 3). In this sense, sensors on board and a communication link between drones and cloud have to be studied, implementing also a filtering on data and possibly a way to reduce battery consumption of all this "dynamic structure", that can be really demanding. Then, to finally produce risk-coherent risk map, two different features have to be implemented: on one hand the environmental map have to be provided by the municipal authority, since they must be more accurate than they actually are. On the other, a new set of algorithms, maybe through neural networks, have to be implemented to evaluate the sheltering factor from satellite images.
In any case, we developed an already operative UASs Traffic Manager, with the hope that in future the regulation of this new kind of air transport will meet the economical and social requirements, always guaranteeing safety and wellness for people.

Bibliography

[1] Divya Joshi (2016), *Commercial Unmanned Aerial Vehicle (UAV) Market Analysis – Industry trends, companies and what you should know*, Business Insider

[2] National Institute of Standards and Technology (2004), *Autonomy Levels for Unmanned Systems(ALFUS) Framework*, Volume 1: Terminology

[3] ENAC (2016), *Mezzi Aerei a Pilotaggio Remoto*

[4] Kehoe, Ben; Patil, Sachin; Abbeel, Pieter; Goldburg, Ken (13 September 2014), *A Survey of Research on Cloud Robotics and Automation*, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.

[5] Robotics-vo, (2014), *A Roadmap for U.S. Robotics From Internet to Robotics 2013 Edition*

[6] K. Dalamagkidis, K. Valavanis, L.A Piegl (2012),*On integrating unmanned aircraft systems into the national airspace system*, Springer

[7] US Department of Defense (2007), *Unmanned systems safety guide for DoD acquisition*

[8] R. Clothier, R. Walker, N. Fulton, D. Campbell (2007), *A casuality risk analysis for UAS operations over inhabited areas*

[9] Range Safety Group, Range Commanders Council (2007), *Common risk criteria standards for national test ranges:Supplement. Supplement to document 321-07*

[10] Range Safety Group, Range Commanders Council (1999), *Range safety criteria for UAV: Rational and Methodology Supplement. Supplement to document 323-99*

[11] Range Safety Group, Range Commanders Council (1999), *Range safety criteria for UAV. Document 323-99*

[12] National Transortation Safety Board (2008), *Accidents database and synopses*

[13] Weibel,Hansman (2004), emphSafety considerations for operation of small UAV in civil airspace, Master's Thesis, Massachusetts Institute of Technology

[14] Neades DN, Rudolph RR (1984) *Head injury prediction capability of the hic, hip, simon and ulp criteria. Accident Analysis and Prevention*

[15] Haber JM, Linn AM (2005) *Practical models of human vulnerability to impacting debris*

[16] Walker SH, Duncan DB (1967) *Estimation of the probability of an event as a function of several independent variables*

[17] G. Guglieri, G. Ristorto, *Safety Assessment for Light Remotely Piloted Aircraft Systems*

[18] G. Guglieri, F. Quaqliotti (2014), *Analisi di rischio: metodologie e considerazioni*, Workshop "mezzi aerei a pilotaggio remoto"

[19] European Aviation Safety Agency (EASA) (2005), *Policy for UAV cerification*, A-NPA,No 16/2005

[20] Sheldon M Ross, *Probabilità e Statistica per l'ingegneria e le scienze*

[21] Clothier, Reece A. and Walker, Rodney A. (2006), *Determination and Evaluation of UAV Safety Objectives*, In Proceedings 21st International Unmanned Air Vehicle Systems Conference, pages 18.1-18.16, Bristol, United Kingdom

[22] Weibel,Hansman (2004), *Safety considerations for operation of different classes of UAV in civil airspace*

[23] Primatesta, Capello, Antonini, Gaspardone, Guglieri, Rizzo (2017), *A cloud-based framework for risk aware intelligent navigation in urban environment*
[24] Sara Giammusso, *Cloud Robotics in real time application*
[25] ROS.org, *grid map package summary*
[26] Fankhauser, Hutter (2016), *A Universal Grid Map Library: Implementation and Use case for rough terrain navigation*
[27] *https://www.thoughtco.com/population-density-overview-1435467*
[28] La Cour-Harbo, Anders (2017),*Mass threshold for harmless drones*
[29] Centre for Telecommunications and Information Engineering, Monash University (2016) *Remote Piloted Aerial Vehicles: An Anthology*
[30] Angela Schoellig (2014) *The Role of Unmanned Aerial Vehicles in Future Urban Environments*
[31] NCAR / EOL Workshop (2017) *Unmanned Aircraft Systems for Atmospheric Research*
[32] Charles A. Poynton (2003). *Digital Video and HDTV: Algorithms and Interfaces.*
[33] Oxford Staff (January 26, 2017). *Waypoint—Definition in English* OxfordDictionaries.com. Retrieved January 26, 2017.
[34] *https://en.wikipedia.org/wiki/MotionPlanning*
[35] S. Scheggi, S. Misra (1994) *An experimental comparison of path planning techniques applied to micro sized magnetic agents.*
[36] Hart, Nilsson and Raphael (1968) *A formal basis for the heuristic determination of minimum cost path.*
[37] A. Stentz (1994) *Optimal and efficient path planning for partially known environment*
[38] A. Stentz et al.(1995) *The focussed D* algorithm for real time re planning*
[39] M. Nosrati, R. Karimi, H.A. Hasanvand (2012) *Investigation of the star search algorithms*
[40] A. Nash, K. Daniel, S. Koenig, A. Felner (2007) *Theta*: Any angle path planning on a grid.*
[41] G. Guglieri, A. Lombardi, G. Ristorto (2015) *Operation oriented path planning strategies for rpas*
[42] *https://en.wikipedia.org/wiki/Randomizedalgorithm*
[43] L.E. Kavraki, P. Svestka, J.C. Latombe, M.H. Overmars (1996) *Probabilistic roadmaps for path planning in high dimensional configuration spaces* IEEE transactions on Robotics and Automation
[44] S.M. LaValle (1998) *Rapidly Exploring Random Tree: a new tool for path planning*
[45] Maniezzo (1992) *Distributed optimization by ant colonies*
[46] S. Karaman, E. Frazzoli (2010) *Sampling-based algorithms for optimal motion path planning*
[47] ROS.org, *"grid_ map" Package Summary*
[48] Fankhauser, Hutter (2016)*A Universal Grid Map Library: Implementation and Use case for rough terrain navigation*
[49] *http://www.ros.org/about-ros/*
[50] *https://en.wikipedia.org/wiki/Robot_ Operating_ System*
[51] *http://wiki.ros.org/Nodes*
[52] *http://wiki.ros.org/Topics*
[53] *http://wiki.ros.org/Services*
[54] *https://www.faa.gov/uas/research/utm*
[55] *https://utm.arc.nasa.gov/index.shtml*
[56] European Commission - Press release (2017) *Aviation: Commission is taking the European drone sector to new heights*
[57] *http://www.sesarju.eu/U-Space*
[58] Peter E Hart, Nils J Nilsson, and Bertram Raphael *A formal basis for theheuristic determination of minimum cost paths.* IEEE transactions on SystemsScience and Cybernetics, 4(2):100–107, 1968
[59] *http://www.gazebosim.org/*

[60] *https://github.com/mavlink/mavlink*
[61] Dronecode *QGroundControl - Ground Control Station for PX4 and ArduPilot UAVs*
[62] *https://sdn.ieee.org/newsletter/march-2016/mobile-edge-computing-an-important-ingredient-of-5g-networks*
[63] E. Stabile (2018) *Cloud-Based UASs Traffic Management:General Architecture*
[64] F. Polia (2018) *Cloud-Based UASs Traffic Management: Trajectory Tracking and Collision Avoidance*
[65] *https://github.com/osrf/uctf*