POLITECNICO DI TORINO

Master degree course in Mechatronic Engineering

Master Degree Thesis

Sheltering factor map generation via machine learning tool

Supervisor
Ing. Alessandro Rizzo

Candidate
Armando Lanzaro

Internship Tutor
Ing. Carlos Norberto Perez Montenegro

ACADEMIC YEAR 2017-2018

Abstract

In the last years the usage of Unmanned Aerial Vehicles (UAVs) has increased, especially in civil applications. With respect to the traditional manned aviation, these are usually characterized by lower experience and ability of the operator, an inferior build quality of the aircraft and the fact that UAVs move in the complex urban environment, where accidents are more likely to happen and can be harmful for inhabitants, causing them injuries or even fatalities. Despite this, there is still no common regulation on unmanned flights, not only among different countries but sometimes not even in the same national airspace and this leads to a confused situation in which it is almost impossible to apply safety criteria for unmanned aviation.
In this context there is a need to have a coherent risk definition, assessment and to find a way to mitigate risk. This thesis focuses on the sheltering factor, a parameter that expresses quantitatively the sheltering offered by structures to protect people from UAV falls or from debris following an accident. The purpose of this work is to develop a tool that, given a satellite photograph of an area, is able to identify its characteristics and to generate a corresponding sheltering map. The innovation introduced is the use of machine learning and in particular of a neural network that exploits supervised learning to compute the sheltering factor starting from the analysis of the original picture.
The first step to reach the goal is to have a coherent, exhaustive and standardized risk metric and to do that, the state of art is presented and discussed in its strengths and limits. Then a complete description of the concept of risk is introduced to explain risk assessment and provide the instruments to quantitatively perform it, through a definition of risk as the victims' rate per hour of flight. Current safety standards are then analysed to find an upper-bound for the risk that could be used to have a unified regulation for different national flight authorities. Moreover a special focus is put on the sheltering factor, giving its definition, some examples and scales, including the one used in this work.
Once defined the risk metric, an initial data set of satellite pictures to work on is chosen and an algorithm is introduced to perform image analysis and processing on it, to extract information on colours and edges. In parallel the same pictures are divided in sub-parts and evaluated by inspection to assign each one a sheltering factor. At this point, the results of image analysis and the desired, labelled, sheltering factors, are used respectively as input and targets to perform supervised learning with a two-layer feed-forward neural network. The choice of the best network is discussed with reference to its performance and

parameters, like the number of neurons and the training algorithm. The network is, at this point, exploited to find a function correlating image analysis data and the corresponding sheltering factor. Then the function developed in this way is integrated in an algorithm that, when given a satellite image, finally generates a sheltering map.
The last part of the work explains how the sheltering map can be used for path planning of the UAV flight, to choose, among all the possible ones, the trajectory that presents the lowest risk for humans. Moreover, a cloud-based traffic manager architecture is introduced to integrate map generation, path planning and validation and advantages of using cloud communication and computing, like high computational power and real-time update of information, are discussed.
The procedure to automate the definition of the sheltering factor from a satellite picture and the capabilities offered by cloud computing, will give, in the future the possibility to remove the human element from the control loop, allowing the cloud-based traffic manager to handle every part of the flight of a UAV, from the planning to the landing, guaranteeing, at the same time an high safety level for humans.

Contents

List of Tables v

List of Figures vi

1 Introduction 1
1.1 Context definition . . . 1
1.2 Structure of the thesis . . . 3

2 Ground Impact Risk Modelling 5
2.1 Unmanned Missions Risk: Introduction . . . 5
2.1.1 Main Concepts . . . 5
2.1.2 UAVs' Types of Accidents and Equivalent Level Of Safety . . . 6
2.2 Early Flight Termination Risk Modelling . . . 11
2.2.1 Evaluation of Exposed Inhabitants: N_{exp} . . . 13
2.2.2 Estimation of fatality probability for people exposed to crash: $P(f|e)$ 15
2.2.3 Evaluation of Impact Kinetic Energy: E_{imp} . . . 18
2.2.4 Estimation of Sheltering Factor: P_S . . . 19
2.3 Mid-Air Collisions Risk Modelling . . . 22
2.3.1 Estimate of the Mid Air Collision's Rate: f_{MAC} . . . 23
2.3.2 Evaluation of the Collisions Rate Between UAVs: f_{UAV} . . . 23
2.4 Ground Impact Risk Modelling: A Complete Model . . . 25

3 Image processing and analysis 27
3.1 Image processing . . . 29
3.1.1 Sheltering factor assignment . . . 29
3.1.2 Image Analysis . . . 32

4 Linear Regression 40

5 Neural network 45
5.1 The neural network toolbox . . . 45
5.2 Simulation . . . 50
5.2.1 Levenberg-Marquardt algorithm . . . 50

5.2.2 Bayesian Regularization algorithm 53
5.2.3 Discussion of the results and choice of the neural network 55

6 Map generation 58
6.1 Sheltering factor extaction 58
6.2 A second approach: adding a filter 61
6.3 Testing the algorithm 62
6.3.1 Best cases 64
6.3.2 Worst Cases 84
6.3.3 Overall results 99

7 Map Manager and Path Planning 103
7.1 Map Manager 104
7.2 Path Planner 105
7.3 Path validation and risk acceptance 107
7.4 A cloud based architecture 111

8 Conclusions and future developments 113

Bibliography 115

List of Tables

2.1 $P(fatality|exposure)$ evolution in function of P_S. [27] 18
2.2 Sheltering Coefficient scale proposed by [19] and [20] 20
2.3 Sheltering coefficient scale used 21
4.1 Results of Linear Regression 41
4.2 Results of Linear Regression 44
5.1 Levenberg-Marquardt algorithm - 70/15/15 division 51
5.2 Levenberg-Marquardt algorithm - 70/10/20 division 51
5.3 Levenberg-Marquardt algorithm - 70/5/25 division 52
5.4 Levenberg-Marquardt algorithm - 75/10/15 division 52
5.5 Levenberg-Marquardt algorithm - 75/5/20 division 52
5.6 Levenberg-Marquardt algorithm - 80/5/15 division 53
5.7 Bayesian regularization algorithm - 70/5/25 division 54
5.8 Bayesian regularization algorithm - 75/5/20 division 54
5.9 Bayesian regularization algorithm - 80/5/15 division 54
6.1 Results of Test One 68
6.2 Results of Test Two 73
6.3 Results of Test Three 78
6.4 Results of Test Four 83
6.5 Results of Test Five 89
6.6 Results of Test Six 94
6.7 Results of Test Seven 98
6.8 Overall results obtained on the whole data set 101

List of Figures

2.1 Primary and secondary accidents in UAV operation and possible outcomes 8
2.2 Risk reference systems for large manned aircraft (the grayed areas indicate unacceptable risk) [14] . . . 10
2.3 NATO UAVs classification . . . 10
2.4 $P(fatality|exposure)$ evolution with respect to Kinetic Energy and Sheltering [27] . . . 17
2.5 Gas model application[27] . . . 25
3.1 Example of image division and definition of the sheltering matrix $\mathbf{M}$ by inspection . . . 31
3.2 Colour wheel with screens' RGB pixels . . . 33
3.3 Edge detection . . . 35
3.4 Image processing scheme . . . 36
3.5 . . . 37
4.1 Error distribution . . . 41
4.2 Aerial Picture 1 . . . 42
4.3 Sheltering map assigned by inspection of Figure 4.2 . . . 43
4.4 Sheltering map obtained with linear regression applied to Figure 4.2 . . . 43
4.5 Distribution of the errors . . . 44
5.1 The neural network . . . 46
5.2 Neural Network Training Tool . . . 49
5.3 Error histogram of the chosen neural network . . . 56
5.4 Image processing scheme . . . 57
6.1 Graphic representation of the sheltering extraction and map generation . . . 60
6.2 Supervised Training working principle . . . 61
6.3 Aerial Picture 1 . . . 64
6.4 Sheltering map assigned by inspection of Figure 6.3 . . . 65
6.5 Sheltering map obtained with neural network function applied to Figure 6.3 65
6.6 Sheltering map obtained with neural network function and filter applied to Figure 6.3 . . . 66
6.7 Distribution of the errors between sheltering maps 6.5 and 6.4 . . . 67
6.8 Distribution of the errors between sheltering maps 6.6 and 6.4 . . . 67
6.9 Effect of the filter in reducing the noise on the original error signal . . . 68

6.10 Aerial Picture 2 69
6.11 Sheltering map assigned by inspection of Figure 6.10 70
6.12 Sheltering map obtained with neural network function applied to Figure 6.10 70
6.13 Sheltering map obtained with neural network function and filter applied to Figure 6.10 71
6.14 Distribution of the errors between sheltering maps 6.12 and 6.11 72
6.15 Distribution of the errors between sheltering maps 6.13 and 6.11 72
6.16 Effect of the filter in reducing the noise on the original error signal 73
6.17 Aerial Picture 3 74
6.18 Sheltering map assigned by inspection of Figure 6.17 75
6.19 Sheltering map obtained with neural network function applied to Figure 6.17 75
6.20 Sheltering map obtained with neural network function and filter applied to Figure 6.17 76
6.21 Distribution of the errors between sheltering maps 6.19 and 6.18 77
6.22 Distribution of the errors between sheltering maps 6.20 and 6.18 77
6.23 Effect of the filter in reducing the noise on the original error signal 78
6.24 Aerial Picture 4 79
6.25 Sheltering map assigned by inspection of Figure 6.24 80
6.26 Sheltering map obtained with neural network function applied to Figure 6.24 80
6.27 Sheltering map obtained with neural network function and filter applied to Figure 6.24 81
6.28 Distribution of the errors between sheltering maps 6.26 and 6.25 82
6.29 Distribution of the errors between sheltering maps 6.27 and 6.25 82
6.30 Effect of the filter in reducing the noise on the original error signal 83
6.31 Aerial Picture 5 84
6.32 Sheltering map assigned by inspection of Figure 6.31 85
6.33 Sheltering map obtained with neural network function applied to Figure 6.31 85
6.34 Sheltering map obtained with neural network function and filter applied to Figure 6.31 86
6.35 Distribution of the errors between sheltering maps 6.33 and 6.32 87
6.36 Distribution of the errors between sheltering maps 6.34 and 6.32 87
6.37 Effect of the filter in reducing the noise on the original error signal 88
6.38 Aerial Picture 6 89
6.39 Sheltering map assigned by inspection of Figure 6.38 90
6.40 Sheltering map obtained with neural network function applied to Figure 6.38 90
6.41 Sheltering map obtained with neural network function and filter applied to Figure 6.38 91
6.42 Distribution of the errors between sheltering maps 6.40 and 6.39 92
6.43 Distribution of the errors between sheltering maps 6.41 and 6.39 92
6.44 Effect of the filter in reducing the noise on the original error signal 93
6.45 Aerial Picture 7 94
6.46 Sheltering map assigned by inspection of Figure 6.45 95

6.47 Sheltering map obtained with neural network function applied to Figure 6.45 95
6.48 Sheltering map obtained with neural network function and filter applied to Figure 6.45 96
6.49 Distribution of the errors between sheltering maps 6.47 and 6.46 97
6.50 Distribution of the errors between sheltering maps 6.48 and 6.46 97
6.51 Effect of the filter in reducing the noise on the original error signal 99
6.52 Distribution of the errors between the output of the unfiltered algorithm and the desired values of sheltering factor, computed on the whole data set of 200 pictures 100
6.53 Distribution of the errors between the output of the filtered algorithm and the desired values of sheltering factor, computed on the whole data set of 200 pictures 100
7.1 Mission risk management and acceptance 109
7.2 Example of mission from a point A to B 110
7.3 Path planning based on the sheltering map 110
7.4 Path followed by the UAV to minimize the risk 111
7.5 Scheme of the communication between UAVs and CBUTM 112

Acronyms

ANN Artificial Neural Network

CCS Cloud Control Station

CBUTM Cloud-Based UASs Traffic Manager

ELOS Equivalent Level of Safety

GPS Global Positioning System

GUI Graphical User Interface

NAS National Airspace System

NFZ No-Fly Zones

RAMM Risk-Aware Map Manager

TLS Target Level of Safety

UAS Unmanned Aerial System

UAV Unmanned Aerial Vehicle

UTM UAVs Traffic Management

Chapter 1

Introduction

1.1 Context definition

In the last ten years Unmanned Aerial Vehicles have been rapidly growing in popularity and they have been used in a large variety of applications, spanning from the more traditional military field to civil applications. This development is in agreement with the transformation that should take place in the next decade of the urban areas in smart cities that will be based on the most advanced information technologies, from Data Science to Internet of Things, from artificial intelligence to robotics, with the purpose of increasing the efficiency of services and improve the quality of life of the inhabitants, while at the same time reducing the costs through policies of optimization of the communication between people and services, development or improvement of a sustainable mobility system and the achievement of energy self-sufficiency [18][10]. In this context Unmanned Aerial Vehicles will play an important role thanks to the sensors they may be equipped with, to their agility and the possibility to connect them to the cloud to manage information and their flights. The possible applications may include traffic management, pollution and environmental hazard monitoring and surveillance [28]. According to [24], consumer drones market will grow by a compound annual growth rate (CAGR) of 31.3%, reaching 29 millions pieces by 2021, while enterprise UAVs, with a CAGR of 51% in the same period. However, this increase takes place in a current confused situation due to the lack of common regulation and laws on UAV flight, in which every country has a different approach, with rules that are sometimes too restrictive, sometimes, on the contrary, too relaxed.
Before entering in details, some concepts have to be introduced to better understand the environment on which this work is based. The formal definition of Unmanned System given by [29] is:

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 is focused on Unmanned Aerial Vehicles (UAVs). The main difference between manned and unmanned aircraft is the presence of humans on board. In particular, the lack of passengers and of a pilot aboard, allow UAVs to be smaller, lighter and to perform missions that are considered too dangerous or too repetitive for a human operator [39]. A common definition of unmanned mission is:

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.

UAVs can be classified based on their Level of Autonomy (LOA), according to [29], in:

Remote Controlled: the human operator, without benefit of video or other sensory feedback, directly controls the actuators of the UAV on a continuous basis and via a tethered or radio linked control device, using visual line-of-sight.

Teleoperated: the human operator, using video and/or other sensory feedback, either directly controls the actuators or assigns incremental goals, like waypoints in mobility situations, on a continuous basis, via a tethered or radio linked control device.

Semi-Autonomous: the human operator and/or the UAV itself plans and conducts a mission.

Fully Autonomous: the UAV is expected to accomplish its mission, within a defined goal and without any human intervention.

The current state of art allows to imagine that, in the future, missions will have to be accomplished in urban scenarios, flying at quite low altitude and in fully autonomous mode, with no human in the loop, neither driving nor supervising. Since the level of safety to be guaranteed must be yet the same as in the case with an human operator involved, an accurate analysis of the risks and of the way to mitigate them is needed.
This work is focused the sheltering factor, a parameter that expresses the ability of structures and objects to shelter people, protecting them from the fall of a UAV or of debris following a mid-air collision. To automatically derive the sheltering factor from a satellite picture, we have used machine learning techniques, in particular neural networks. We have decided to use this approach because in the last decades artificial intelligence has become more and more used in many operational fields. In fact algorithms have improved so much that currently they are considered excellent tools to face a large variety of problems. ANNs are modular, they do not need to be provided with a specific algorithm for a certain task because they learn by example. In fact the layers of neurons are able to receive as input huge quantities of data and to automatically elaborate them to find the

outputs, with a learning ability inspired by the human brain.
This structure makes neural networks perfect for identifying patterns or trends in data, so they are well suited in general for prediction and forecasting problems. Currently ANNs provide the best solutions in problems like:

- Speech recognition
- Image recognition
- Classification
- Predictive analysis

In fact their ability to handle quantities of data that can be too big or too complex for a traditional solution, makes ANNs perfect where there is a great variability of input data. The most widely used applications in fact are related to speech recognition, that are exploited by digital devices, especially smartphones, and services like Google Assistant or Siri. Financial firms as well have been using neural networks for years in the investment decision making process [41]: these tasks involve intuitive judgement or require the detection of data patterns which elude conventional analytic techniques. ANNs can draw conclusions on incomplete data and they are even able to learn from past mistakes, so they are used as traders and investors. Neural networks are gaining a central role in medical diagnosis as well. They are used to define a disease based on the symptoms and observation of data distribution like heart rate, for example. In the last years they have been exploited for their image recognition ability and now neural networks have better results than humans in finding skin cancer, for example.
The fields of application of ANNs are so many that can not even be listed all, including clustering, profiling, data denoising, electromagnetic analysis and many more, but in this thesis we are particularly interested in image recognition.
The convolutional neural networks, or ConvNet, are able to extract features from input images, dividing them in smaller squares of input data and analysing them singularly. The are used for face recognition and in general object recognition by cameras.
The aim of this project is to use for the first time a convolutional neural network to analyse an image coming from a drone camera and, based on that, to automatise the definition of the sheltering factor and therefore of the sheltering map of that specific region. This can be exploited as a risk map by the path planner or it can be used to generate a further risk map if more variables are involved.

1.2 Structure of the thesis

The overall purpose of this work is to guarantee safety during the flight in an urban environment of fully autonomous UAVs. Therefore in an initial phase it is fundamental to understand that when referring to safety, we are addressing to the one of people on the

ground that can be affected by the consequences of a UAV crash. This concept is crucial, since it underlines that the centre of this work are the people, not the UAVs themselves. The first operation to be made, in Chapter 2, is to find an exact definition of risk as the frequency of fatalities in terms of victims per hour of flight of a UAV. All the accident types are listed and a special attention is put on the ones that can cause ground impact, Early Flight Termination and Mid-Air Collision. The metric to compute the risk in these two cases is given and we introduce an overall Ground Impact model of risk to take into account both of them. Moreover, in this chapter the sheltering factor is defined starting from its state of art and introducing the new classification that is used in this work.
Chapter 3 introduces the actual activity: extraction of a map based on the sheltering factor, starting from a satellite picture of an area. To do that, a set of sample images is introduced and we define an algorithm to divide them in portions and analyse each of them based on colours and the presence of edges. At the same time, every picture is explored by inspection to define a desired sheltering factor value for each portion. Using the latter values and the ones coming from image analysis, in Chapter 4, a first approach is tried to associate image information to desired sheltering factor: the technique exploited is linear regression and an example is made to check the performance. However it is not satisfying and therefore, a second approach is adopted and presented in Chapter 5, where the idea of using a neural network to perform the aforementioned association is introduced, to create a fitting function between the image analysis data and the target sheltering factors. At first, the MATLAB Neural Network Toolboox that we have used is presented, in its potentialities and settings. Then it is used to perform supervised learning between the two data sets and many simulations are made with different algorithms and parameters to find the best network for the purpose. Once defined the latter, it is integrated, in Chapter 6, in a complete map generation algorithm that, given a satellite picture is able to associate a corresponding sheltering map. The algorithm performs image analysis, extracting information on colours and edges and it feeds them as input to the neural network function. The output is then re-arranged in a matrix form to be visualized as a map.
In Chapter 7 a wider picture is given, metaphorically: we present the context in which the sheltering map is used, introducing the Path Planner, that, given a mission between a starting and an ending point and the aforementioned map, find among all the possible trajectories, the one that minimizes the risk for humans, imposing, basically, to the UAV to overfly highly sheltered areas. The Path Validator is the last step in the flight management process: it verifies if the chosen trajectory respects the safety criteria imposed by the standards and, based on that, it gives the permission or denial to the mission. These three elements, Map Generator, Path Planner and Path Validator, are part of a cloud-based architecture that manages every aspect of the flight of UAVs and presents many advantages that will be introduced.
Finally, all the conclusions and the considerations on possible future developments are exposed in Chapter 8, with reference to the current situation and the future possibilities.

Chapter 2

Ground Impact Risk Modelling

2.1 Unmanned Missions Risk: Introduction

2.1.1 Main Concepts

In agreement with thesis organization given previously, this chapter is going to analyse the concept of risk for an unmanned mission, introducing some definitions and computational methods.
First of all, when discussing about risk we refer to the time frequency in which the drone causes fatal or very serious injuries to one ore more people on ground [8]. It is important to underline the fact that we are not dealing with a risk for the drone, like a crash or a in-flight collision, but exclusively for the people on the ground that can be directly or indirectly affected by a UAV-caused accident. Risk analysis is aimed at avoiding hazard for the human beings, not at avoiding the accident itself.
Safety has always been one of the main issues in the avionic industry and development, therefore many risk models have been created, each one with a specific level of accuracy [8]. When building a risk model, it is important to not underestimate the risk, and instead always prefer a more conservative choice with respect to another that could not take into account all the available elements.
The final purpose of risk analysis is to establish if an Unmanned Aerial Vehicle (UAV) is able to guarantee a required Level of Safety (LOS) when overflying a certain geographical area in a given time window. In case this analysis gives a negative answer, the system must be capable of suggesting one or more countermeasures to be taken in order to increase the overall safety of the mission. This last part is called risk management [8].
The procedure to follow is generally organized in four steps [36]:

Mission Definition and Hazard Identification A description of the mission has to be provided and, as a consequence, the definition for safety bounds is derived. In the case treated by this work, that is civil operations in an urban environment, the risk limit is imposed by each national flight authority and it can be considered

constant. In a nutshell it means that the flight agencies decide a maximum number of victim per hour of flight, which must be used as upper-bound for every mission. Finally, all the possible hazards that may appear during the flight have to be listed.

Risk Assessment Evaluation of the risk value corresponding to every hazard has to be made.

Risk Reduction and Managment The overall risk is compared with the safety bound. If it turns out to be higher, some countermeasures have to be taken in order to reduce it.

Risk Acceptance Once verified that the risk is below the limit, the mission is approved and it can start.

Hazards are divided in two categories: due to internal failures and due to external causes. In general hazard analysis can be very time consuming, especially for what concerns the UAV internal failures. Many tools to perform it have been already developed but, identifying all the possible hazards is not the purpose of this work that, as a matter of fact, is rather aimed at using them to compute a coherent risk value. This concept will be further clarified in next chapters.
Finally, once the risk analysis is completed the Risk-Aware Map Manager (RAMM) is in charge of producing a corresponding risk map. At this point it is important to highlight that, despite appearing similar, risk assessment and risk map generation are two different processes: the first one is more theoretical and it must be followed by the second, definitely more practical. In fact, a Risk-Aware Map Manager is a crucial block of the Cloud-Based UASs Traffic Management system [27] and it controls and merges all the available information on risk for UAVs, producing, as a consequence, a coherent Risk Map for the drone's mission. The RAMM has to be able to work in real-time, updating its parameters and computation when something changes.
The creation of such a map allows the path planner algorithm to find the path that presents the lowest risk, for the UAV to accomplish the assigned mission.

2.1.2 UAVs' Types of Accidents and Equivalent Level Of Safety

The preliminary step before performing a quantitative analysis of the risk is to briefly identify and introduce the main UAVs types of accident.
During their mission UASs face different types of hazard that can cause three distinct kinds of accidents [8]:

Unintended or Abnormal Mobility Operation This category includes all the accidents that happen when the drone has not taken off yet and it is still on ground. They are due to unexpected movements of the UAV that results in a potential danger for the ground crew members. This kind of accident are normally caused by

an operator error, like not having a correct view of the UAS or not checking if the area is cleared and therefore they can mitigated with a stricter observation of the standard operating procedures and security protocols [42].
However this accident type is not in the field of interest of this thesis, since it involves UAVs in a phase preceding the take off and, as a concequence, it will not be analysed any further.

Mid-air Collisions This category includes all the accidents than happen during the flight: collisions between two UAVs or between an Unmanned Vehicle and a manned aircraft, or even a crash on a building, a situation very likely to happen in an urban scenario.
In this case the primary accident is constituted by possible victims in the crash, like the passengers of a line flight, but given the very different operating altitudes and size of planes and UAVs, the collision between the two is very unlikely to happen in an urban environment.
However a secondary accident is constituted by the ground impact of debris due to collision, that can cause serious injuries or damage properties [7]. This aspect is crucial and it will therefore be part of the analysis in this thesis.

Early Flight Termination This category includes all the accidents due to a loss of control and a subsequent anticipated landing of the UAV on ground or water.
The early flight termination can be either controlled or uncontrolled. In the first case the operator is able to select a point of impact and possibly a speed in order to reduce the chance of fatal events or property damage [7]. In the second case the operator can not perform any emergency control action and the UAV crashes. Of course in this case as well the accident is constituted by the ground impact of the drone.

These accident types represent the three main ways in which an unmanned aerial vehicle can, directly or indirectly, cause serious or fatal injuries. It is important to underline that when talking about injuries, we refer only to people involved in the accident and not to damages to the drone itself. In some cases potential collateral damages to property can as well be taken into account [9].
In Figure 2.1 all primary and secondary accidents and their possible outcomes are resumed.
However possible indirect negative effects may occur, especially regarding environmental pollution due to fuel leakage, fire following the accident or even the diffusion of an harmful or toxic payload, like in the case of UASs used in agricultural application. Furthermore a possible secondary damage is social rejection and blame: an high accident rate, even without fatal events, could cause in people a sense of repulsion toward UAVs and their missions, possibly leading to limitations. This may arise also if the areas involved in accidents are cultural-/societal-sensitive, like parks, monuments, schools and churches [8]. In this scenario it is crucial to be able to assure an adequate level of safety to the people directly involved or indirectly affected and therefore a framework to define, quantify and

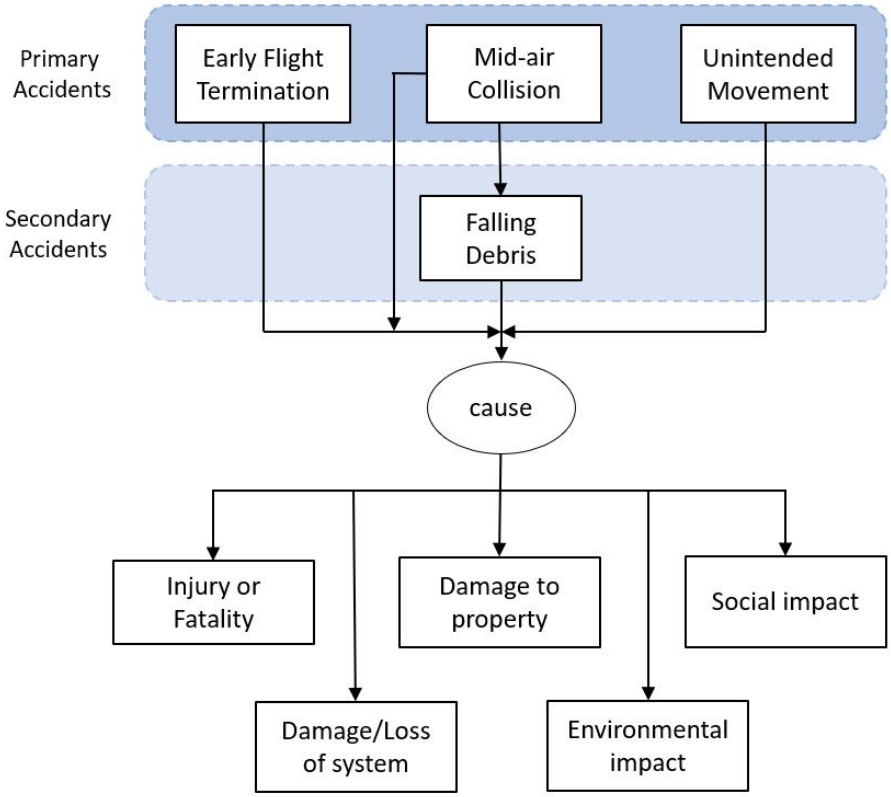


Figure 2.1: Primary and secondary accidents in UAV operation and possible outcomes

reduce the risk is needed.
The concept of risk does not have a commonly accepted definition, since it changes with different operative scenarios and according to the target to be achieved. However, in the aviation environment the most commonly used is the following:

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.

What stems clearly from this definition is that risk considered a frequency, so it has a direct connection with time t. Therefore to find the risk it is important to evaluate the number of fatalities per hour of flight that a UAS in a certain mission can cause [8].
This must be done for both the accident types that may produce ground victims. The risk evaluation of mid-air collision and early flight termination will be treated separately in the next pages, in order to give each of them a proper metric. At the end, they will be merged to provide an overall ground impact risk model.
To be considered safe, a mission must have an overall risk value lower than the objective one imposed by the national authority that controls flights.
A common practice is to compare unmanned with the more traditional manned aviation.

In general the main difference between them lies in the presence of a human pilot on board and on the level of automation involved. Anyway, despite UASs may share design and even mechanical components with the military or general aviation aircraft, the difference between the two introduces new failure modes and, accordingly, a higher perceived risk, that needs to be mitigated.
As stated by many national aviation agencies the target to achieve is at minimum an Equivalent Level of Safety to the one assured by manned aviation. This is also known as ELOS principle.
According to the Range Commanders Council and the European Aviation Safety Agency, respectively:

> Any UAV operation or test must show a level of risk to human life no greater than that for an operation or test of a piloted aircraft [38]

> Airworthiness standards should be set to be no less demanding than those currently applied to comparable manned aircraft nor should they penalise UAS by requiring compliance with higher standards simply because technology permits. [13]

The manned aviation is regulated by requirements and standards that define a Target Level of Safety (TLS) to be compliant with. These risk reference systems typically classify events based on severity, in terms of injuries and/or fatalities, and maximum rate of occurrence.
Nevertheless there is a lot of criticism on the usefulness of the ELOS principle in general. In fact derivating the same risk reference systems for UASs is not trivial since, on one hand they introduce new failure modes, but on the other they do not carry passengers and this decreases inevitably the number of people exposed to risk.
At the same time, classical aviation vehicles can rely not only on control systems but also on human ability and expertise to avoid a possible aircraft fall. In general it is very difficult to compare vehicles that are so different in terms of size, control mechanism and functionalities.
A better idea to evaluate the correct maximum risk value is based on statistics [8][30] taking into consideration only on ground victims and not the flight passengers.
In this way the typical risk lower-bound of fatality rate for classical manned aviation, which is $10^{-7}h^{-1}$ (figure 2.2), can be increased to find the maximum for a UAV mission in the following range:

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

Basically it means that one fatal event every 100.000 hour of flight.
The last step before introducing a quantitative risk analysis is to define the type of Unmanned Vehicle to be taken into account in this thesis. Today UAVs are available with different size and characteristics: they can be equipped with fixed or rotary wings and they have various speed and size depending on their functionality [31].
Figure 2.3 shows a table with the classification of UAVs produced by NATO, with a clear

		Catastrophic	Hazardous	Major	Minor	No safety effect
Probable	$> 10^{-5}$ h^{-1}					
Remote	$< 10^{-5}$ h^{-1}					
Extremely remote	$< 10^{-7}$ h^{-1}					
Extremely Improbable	$< 10^{-9}$ h^{-1}					

Figure 2.2: Risk reference systems for large manned aircraft (the grayed areas indicate unacceptable risk) [14]

division in categories and classes depending on weight, employment and flight characteristics.

Despite the map generation that is the purpose of this thesis does not depend on the

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 2.3: NATO UAVs classification

type of vehicle used, it is appropriate to define one. To move safely in an urban scenario Micro and Mini multi purpose UAVs can be chosen: they are the most widespread due to

ease of operation and lower cost and they guarantee a satisfying mission radius (up to 25 km), while keeping a low weight (under two kilograms). Moreover, a great advantage is that high-resolution measurements can be carried out thanks to the ability to fly multiple platforms simultaneously. Lastly it is important to underline that it will be assumed that the UAV does not carry any harmful or toxic payload.

2.2 Early Flight Termination Risk Modelling

The first model to be discussed is Early Flight Termination. The purpose of this section is to find an analytic expression that, known the flight conditions, is able to evaluate correctly the number of victims per hour due to an uncontrolled landing of the UAV.
The unexpected flight ending may be due to different causes, both internal and external (that do not include anyway mid-air collisions).
According to [8], in these cases the risk model can be obtained as:

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

where:
$f_{F,EFT}$ = fatal events frequency due to an Early Flight Termination. Unit of measurement: $[h^{-1}]$
N_{exp} = number of people exposed to the accident.
$P(fatality|exposure)$ = probability that a person involved in the UAV crash will suffer fatal injuries
f_{EFT} = frequency of failures that cause an unexpected ending of the flight. Unit of measurement: $[h^{-1}]$

The parameters that appear in this formula are the ones that characterize and affect a UAS mission. They will be further developed in the next pages to detail the risk evaluation analysis. Equation 2.2 combines them to have a risk analysis that is:

- Coherent with the available statistical data
- Not expensive from a computational point of view
- Easy to customize or extend in case new interesting parameters emerge
- Independent from the dimension of the area taken into consideration

Some considerations about this formula and its parameters have to be done.
$P(fatality|exposure)$ it's the crucial point of the equation. It takes into consideration drone kinetic energy, human body vulnerability and the geographical sheltering factor to obtain the probability of having fatal injuries after the collision of a UAV with people. It has been introduced by [8] and then further investigated and extended by [19][20].
The number of exposed people N_{exp} can be easily evaluated starting from the knowledge

of population density and of drone's impact area. An exact formulation will be given in the next section.
Finally, f_{EFT} is another key element to understand, since it expresses the frequency of the ground impacts of the drone, introducing the time parameter in the equation. The first idea of [27] to obtain an estimation was to use probabilistic studies, combining the external causes that may affect the drone (bad weather, for example) and the likelihood of internal failures. However this approach presents two reasons for criticism that can not be ignored:

1. An evaluation of ground impact frequency starting from the hazards that can be the cause of it strictly implies the complete knowledge of all the possible threats and of the quantitative measure of the impact that each of them has on the probability of having a crash. Ignoring even one hazard only or underestimating its impact would definitely distort all the risk analysis, resulting in a risk that would be under or overestimated.
   An example can be obtained when trying to provide the probability of having a crash due to the high wind speed: it is possible to use the maximum wind speed predicted by the weather report for the period in which the mission is performed, but even using the most detailed weather forecast it is impossible to predict and so take into account exactly the wind gusts, which are actually real cause of crashes.

2. Trying to give an estimate of f_{EFT} means working with the probability that each hazard can happens, and it will eventually cause a crash of the UAV. The trustworthiness of the final result of the analysis has been discussed in the previous point, but it is important to notice that this result is a probability as well, without any unit of measurement. The lack of time element is a critical issue, since as it has been cleared up that risk must be measured in fatal injuries per hour according to the current standards. The importance of time in the equation 2.2 will be studied in deep in next chapters, when a method to evaluate the overall risk of the mission will be provided.

Given that, a pure probabilistic hazard study in the evaluation of f_{EFT} turns out to be hard to perform and to give meaningless results, so, in agreement with [8] and [37], a statistical study of risks must be preferred. In fact a risk analysis in which the ground impact frequency is assumed to be known may seem a simplification, but the purpose of this thesis is not to make a pure hazard analysis, but instead it is the evaluation of risks and their mitigation in order to guarantee safe missions.
The idea is to establish the number of ground impact per flight hour referring to the history of the UAS: each manufacturer in fact have to make a high number of tests in order to have the most correct possible evaluation of the f_{EFT} of its drone. Sometimes these data are not provided by the producers, therefore the value of f_{EFT} must be estimated after a proper number of flight hours, using as initial value a very high failure frequency. Moreover whether a new drone model is designed starting from an already tested one, it

is be possible to initially assume f_{EFT} of the new drone equal to the one of the old one and use it for the risk analysis. Obviously, in this case as well it will be possible to update the data after the observation of some hours of flight.
According to [19] and [20], a realistic and usable f_{EFT} value for light and cheap UAVs can be $f_{EFT} = 0.01\frac{impact}{h}$.
An interesting modification in equation 2.2 can be introduced in case there is very detailed information about the sheltering of an area and the number of people that can be considered sheltered from a possible fall of the UAV.
An estimation of the number of people in an area and on the possibility they are covered can be made for example with the combination of statistical and/or historical data with mobile phone signals, that can be collected by antennas and analyzed by a cloud system. Once identified the different groups of people and the level of sheltering associated to each of them, the equation 2.2 can be reformulated [8]:

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

where $N_{i,exp}$ and P_i refers to the i-th group of people.
In the following subsection, using a top-down approach from the higher level expression 2.2, all its parameters will be discussed in deep, giving a theoretical definition and a method to compute them.

2.2.1 Evaluation of Exposed Inhabitants: N_{exp}

For the risk evaluation of an unmanned mission, it is necessary to estimate in a correct way the number of people possibly affected by the crash. It is important to underline, once again, that the purpose of this work is to avoid human victims, not UAVs crashes, so the measure of the risk is inevitably connected to the number of inhabitants present in the overflown area.
Given a certain geographical area, the number of people exposed to an accident is:

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

where:
N_{exp} = number of people exposed to the accident.
ρ = population density in given area. Unit of measurement: $[\frac{people}{m^2}]$
A_{exp} = exposed/impact area. Unit of measurement: $[m^2]$

First of all we can suppose population density(ρ) in the given area is uniform. This assumption can be done because, as explained in the next chapters, the resolution of the map created by a drone in an urban scenario is about $25m^2$. Considering a uniform population density in such a small area is reasonable and does not impact badly the risk assessment.
A_{exp} is defined as the lethal area of the drone and as a consequence it has to be evaluated taking into account features like UAV's glide angle, dimensions but also the human mean

height and radius [46]. A large number of studies have been published about this topic: there are several methods to compute the injuries-space of a falling drone and specific aircrafts or scenarios can lead to quite different results.
According to [19], one optimal way to evaluate the impact area is the one presented by the Federal Aviation Administration [15]:

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

where:
$r_p = 0.23m$ = mean value of human body radius
R_{UAV} = maximum linear dimension of the UAS
$d = \frac{Hu}{\tan\gamma}$ = horizontal distance travelled by UAV during the fall. Unit of measurement: $[m]$
$Hu = 1.75m$ = mean value of human height
γ = glide angle

It is important to notice that if the UAV is a quadcopter, then it is possible to suppose the fall to be vertical. This would put the term representing the horizontal component of the fall, $2 \times (r_p + R_{UAV}) \times d$ equal to zero.
Differently, for the fixed wing drones, a glide angle $\gamma = 45°$ should be used. This implies that quadctopters can be consider safer then fixed wing UAVs, at least for what concerns the impact area.
The values to be chosen both for Hu and r_p are intended to modify the version of 2.5 proposed in [15] in order to make it more coherent with real statistical values, as exposed in [20].
Once again, it is important to have a right estimation of the population density ρ, since people, and not drones, are the key point of this work for what concerns risk analysis. A wrong assumption would inevitably end up falsifying the whole study.
In older works about safety, like [8] and [13], it is recommended to suppose ρ constant and equal, in urban environment, to $200\frac{people}{km^2}$. This number emerged trying to consider all typical scenarios of civil aviation, so it is a mediation among moments in which the UAS flies over very crowded areas and ones in which it does it over sparsely populated ones instead.
However, if the operative scenario is a city, as in the case we are considering, this approach is inaccurate, since it passes from being too restrictive when the drone is flying over parks, rivers, etc to being not conservative enough in all the other situations, instead.
An idea to obtain a more realistic value of ρ is to extract from the web the population density for each area. In fact since the drone we are considering is connected in cloud with the Cloud-Based UAV Traffic Manager [27], this information can be obtained via internet with a level of detail that can reach the district dimension, instead of an a priori estimate ($200\frac{people}{km^2}$). The values obtained in this way, however, are still static and therefore in some cases not realistic.
The main problems of such an approach are:

1. The values of popolation density are usually measured taking into account only the people living in a certain area. This excludes from the risk evaluation all the people that could be in the area for other reasons.

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

3. It does not take into account the time element. The hour of the day in which the mission takes place is an important element, that drastically changes the number of people that the drone would affect with a crash. Moreover, this approach does not take into consideration all the possible special events, like concerts or sports matches, that cannot statistically be predicted but end up creating big crowds that an UAV must avoid.

According to [27], to overcome these problems it is possible to further exploit the possibilities given by the internet network. In fact, thanks to the cloud, it is possible to connect to the web to acquire more data about areas that the drone is going to overfly and, for example, it is possible to have a knowledge of events and their estimated participation. Even historical or statistical information of the number of people usually present in a given zone at a given time or in case of events, could be extracted.
Finally, to obtain an higher level of precision, one possibility is to estimate the number of mobile phones connected to the antenna responsible for the communications in the area to overfly. All these information can be merged to obtain an estimation in real time of the variation of people present in that area.
This approach is not new, since it is used by the most common maps provider to give an approximate value of traffic on the streets.
Being able to insert the time dependency to this model is important because it allows the cloud framework, that controls the drone through the cloud, to be capable of updating itself even during the drone flight, giving the path planner the possibility to find the best path in real time [27].

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

The term $P(fatality|exposure)$ in the risk equation 2.2 represents the probability[1] of one or more people of having very serious injuries after the impact of the falling drone. In order to better define this probability, in the last years many studies have been made to provide a model of human vulnerability taking into account factors like position of the body, age and physical conformation to estimate the damage that body would suffer after the impact with a UAV at a given speed [32] [21] [44]. However, these models,

[1] Being a probability, $P(fatality|exposure)$ is an adimensional number in range [0,1]

despite being interesting and even inspirational for this thesis, are too detailed and the information they rely on is seldom available, so they will not be exploited in this work.
To evaluate $P(fatality|exposure)$, it is necessary to suppose it to be function both of the kinetic energy of the falling UAV 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 presented by [43] and [36] and modernized by [19] and [20], it is possible to compute $P(fatality|exposure)$ as:

$$P(fatality|exposure) = \frac{1-k}{1-2k+\sqrt{\frac{\alpha}{\beta}} \times (\frac{\beta}{E_{imp}})^{\frac{3}{P_S}}} \tag{2.6}$$

where

P_S is the sheltering factor, defined in the interval $[0, +\inf]$. It is used to determine how exposed is the population to an impact of a falling UAV. It is function of all the natural and artificial shelters in the area, that can be considered obstacles in the crash trajectory, since they can absorb all or part of the impact energy and even deflect debris [43]. The mean value of P_S is equal to 1 and obviously its increase corresponds to a decrease of the $P(fatality|exposure)$. In the following lines a detailed description of the sheltering factor and of the method to compute it will be given.

$k = min(1, (\frac{\beta}{E_{imp}})^{\frac{3}{P_S}})$ is a term properly studied in [43] 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 $P_S = 6$ [8].

β is the lowest level of energy needed to have a fatal event in absence of sheltering ($P_S = 0$). According to the fatality limit of [36], the value can be considered constant and equal to 34J

Figure 2.4 shows the trend of $P(fatality|exposure)$ with different values of kinetic energy and sheltering factor: generally an increase of P_S corresponds to a decrease of the $P(fatality|exposure)$.

The main limit of this kind of formulation is the lack of a coherent metric to describe the sheltering factor. Actually, in fact, there is no tested method to deal with it and this possible problems in correctly evaluating $P(fatality|exposure)$.
According to [8], that is the most important work for what concerns 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.

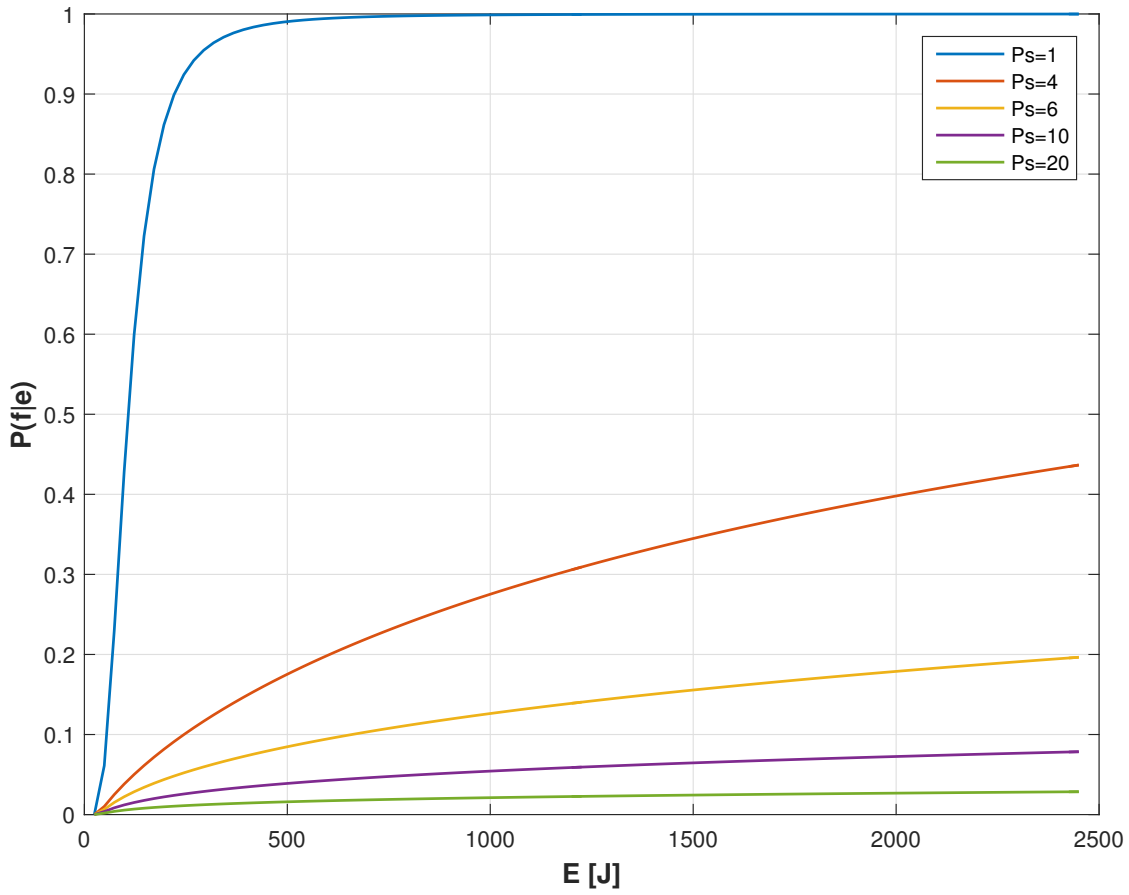


Figure 2.4: $P(fatality|exposure)$ evolution with respect to Kinetic Energy and Sheltering [27]

However this definition is more qualitative then quantitative and, in general, it leads to a quite confused situation. In fact P_S is known to belong to the range $[0, +\text{inf}]$ and therefore it cannot be expressed as a probability, in the range [0,1], for example for people on ground to be protected from the UAV's fall. Moreover there are no bounds in the definition of P_S, so it seems impossible to quantify it just by looking at the operative scenario.
There is the need to define a metric to evaluate P_S according to some specific features. To do that, a good starting point is to observe the behaviour of $P(fatality|exposure)$ when P_S varies.
First of all, when $P_S = 0$, that is when people are not protected at all, $P(fatality|exposure)$ behaves like a threshold: recalling the definition of β, when the E_{imp} is lower than 34J [36] the fatal event has zero probability, while when it is over 34J then $P(fatality|exposure)$ is basically equal to 1. However, an operative scenario with $P_S = 0$ is considered to be very unrealistic, especially in an urban environment, and practically no UAV would guarantee the safety standards in such a situation.

Many tests can be performed to have a clearer idea in the magnitudes of these parameters. For example, when $P_S = 1$, $E_{imp} = 108J$ is requested to have $P(fatality|exposure) = 0.5$ while 1000J would guarantee the certainty of the fatal event. Obviously, in this case as well, for $E_{imp} < 34J$ the $P(fatality|exposure)$ turns out to be null. Some tests for different values of sheltering factor, kinetic energy and fatality probability are shown in Table 2.1.

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. [27]

The purpose of these tests is to show that, even though P_S is defined in the range $[0, +\text{inf}]$, in a real scale $P_S = 10$ is already an upper-bound for the computation of $P(fatality|exposure)$. In fact, when the sheltering factor is equal to 10, a kinetic energy of 3.4MJ is required to have 50% probability of having the fatal event. However, as previously said, in this thesis we are considering to work on a UAV belonging to the category Micro or Mini[2], with a weight that can be supposed to be more or less 1kg. To obtain a value of kinetic energy equal to 3.4MJ, the drone speed should be around $2600\frac{m}{s}$. Such a value is obviously preposterous, so $P_S = 10$ can be supposed as the upper-bound of P_S This means that an area with $P_S = 10$ have to be considered as a complete covered area, that presents no risk at all for people.
As a conclusion, in this thesis we are supposing $P_S \in$ [0,10].

2.2.3 Evaluation of Impact Kinetic Energy: E_{imp}

Equation 2.6 has highlighted the need of an evaluation of the kinetic energy of the impact. To do that, many different solutions have been proposed, from the most accurate ones, taking into account factors like air density or dragging, to the ones that simplify the computation using, instead, only the UAS speed and mass [8]. Following this approach, kinetic energy is defined in [19] as:

$$E_{imp} = \frac{1}{2} \times MV^2 \tag{2.7}$$

where:
M is the UAV mass

[2]Based on the classification made by NATO (Figure 2.3)

$V = \sqrt{2 \times g \times h}$ is the free fall speed from height h

2.2.4 Estimation of Sheltering Factor: P_S

Sheltering factor P_S is a term used to evaluate the protection from a UAS fall offered by natural or artificial elements of an area. Roofs and trees, for example, are obstacles in the crash trajectory of a drone and they can absorb all or part of its kinetic energy and shield from the debris.
The history and state of art on this parameter is interesting because the definition of a metric for it has been a problem since the very beginning of risk analysis.
The first approach developed in [37] did not include this parameter, supposing, as a matter of fact, that no protection was ever available. This idea is due to the fact that the first risk models for unmanned flight were created looking at the aerospace field: space debris had such huge dimensions that it was practically useless to even consider environment protection. However this model turned out to be too conservative when transferred in the drone field: without any shielding possibility, even light UAVs are dangerous and therefore their flight would be limited by law.
Then sheltering factor was introduced by [8] as a term in the range $[0, +\inf]$. This was a major change of perspective, since for the first time the characteristics of an area were considered in the risk evaluation.
In high density areas, like cities for example, it is crucial to find a way to reduce risk, otherwise it would be impossible to satisfy the risk bounds imposed by the national authorities. Buildings, trees but also vehicles and other sheltering objects are able to absorb, partially or in its entirety, the kinetic energy of a falling UAV, reducing drastically the probability of having a fatal event. Therefore an analysis of the kinetic energy required to penetrate a structure is needed, taking into account the different materials as well. After that, a way to map the sheltering ability of an area in the range [0,10] has to be found.
First of all, we can define the sheltering coefficient C_S of a structure as its ability, once impacted, to reduce UAV's kinetic energy. It is important to highlight that it does not consider only the ability of a structure to stop the drone, but also, in case it does not stop, the amount of kinetic energy reduced by the structure. This analysis cannot leave aside some considerations about different materials when evaluating the amount of kinetic energy needed to penetrate a structure. As reported by [43], this energy can be as low as 23J for a roof made by a sheet of 24 gage corrugated aluminum, while it can exceed 506J for roofs made of light concrete. A complete analysis in this sense can be found in [36], but many works proposed their own version. A simple and yet interesting one is the one defined in [19] and [20]: the results take into account the most common features of an urban scenario, as it can be seen in the following table:
It is easy to notice that $C_S = 1$ indicates a building that is able to completely stop the fall of a UAV, on the contrary $C_S = 0$ is used for the completely unprotected areas.
Of course the resistance to an impact of a falling drone also depends on its size and kinetic

C_S	**Structure 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 Coefficient scale proposed by [19] and [20]

energy, but since this thesis and the state of art it refers to ([43][8][19][20]) only consider fairly small drones, we can suppose that they cannot penetrate reinforced concrete buildings.
In this thesis, we actually developed a new scale for the sheltering coefficient identification. It is a 10 levels scale, where C_S, that is to say the ability of a certain zone or structure to reduce the kinetic energy of a falling drone, is expressed ad a percentage. In addition to that, the classification introduces a differentiation in considering the different kinds of buildings: it takes into account also the probability of causing a fatal event in that specific context. Residential buildings, in this framework, present a lower sheltering coefficient with respect to the one offered by educational building, which are as densely populated but only for a limited number of hours during the day. Same for industrial building, that, in addition are less populated than the previous ones. In this framework $C_S = 100\%$ indicates the maximum of protection, that is reached in particular by military buildings, while $C_S = 0\%$ characterises free areas, where there is no structure to offer a shelter from a possible UAV fall.
The table is completed by a third column, the one of levels that are used to represent in a more efficient and compact way the corresponding classes and sheltering coefficients. In fact, in Chapter 3, it will be explained how these have to be assigned by inspection to the various portions of an image; to do that, using a single value is more convenient and immediate than using percentages and it makes computation simpler. Therefore we have decided to introduce these 10 levels and work with them instead, always with reference to the corresponding class and C_S, obviously.

C_S	**Class**	**Level**
0%	No obstacles	0
10%	Small trees and tents	1
25%	Sparse trees	2
30%	Houses	3
50%	Trees	4
60%	Low buildings	5
75%	Residential buildings	6
80%	Educational buildings	7
90%	Industrial buildings	8
100%	Military buildings	9

Table 2.3: Sheltering coefficient scale used

Once C_S of a certain zone is known, the sheltering factor value can be computed as:

$$P_S = C_S \times \frac{A_C}{A_{tot}} \times 10 \tag{2.8}$$

where:
C_S = Sheltering Coefficient.
A_C = Sheltered surface of the given area. Unit of measurement: $[m^2]$
A_{tot} = Total surface of the given area. Its dimensions depend on the map resolution. Unit of measurement: $[m^2]$

Basically, the term $C_S \times \frac{A_C}{A_{tot}}$ evaluates the probability that the UAV falls in a covered area multiplied by the capacity of the area itself to reduce the drone kinetic energy. The multiplicative factor 10 is used to scale P_S to make it compatible with the other terms of equation 2.6.
Obviously the equation 2.8 can be modified to take into account different values of the sheltering coefficient in the same area. In this case the probability of the union is equivalent to the union of the probabilities [35] weighted by the corresponding C_S factor. Then:

$$P_S = (\sum_{i=1}^{n} C_{S,i} \times \frac{A_{C,i}}{A_{tot}}) \times 10 \tag{2.9}$$

where $C_{S,i}$ is the sheltering coefficient of i-th sub-part with area $A_{C,i}$, and P_S is the overall sheltering factor.
Once found both the impact kinetic energy and the sheltering factor, $P(fatality|exposure)$ can be computed to finally evaluate the risk for people on the ground to suffer deadly injuries from an unexpected end of the flight due to internal failure of the UAV or to external causes, excluding mid-air collisions.

2.3 Mid-Air Collisions Risk Modelling

This section is aimed at providing a way to model the risk for people on the ground to suffer deadly injuries due to fall and subsequent impact of a UAV, following a mid-air collision.
As explained previously, in this thesis a simplification has been made: we are not treating the case of collisions between unmanned and manned vehicles, such as line flights or general aviation aircraft, because their difference in size and operating altitude makes a crash very unlikely to happen in an urban scenario. Moreover, without this consideration, such a situation would cause many more deaths among the passengers of the flights, leading to incoherent analysis with respect to the one carried out in the other cases.
That said, National Transportation Safety Board ([30]) divides mid-air collision in three categories, based on who is involved in the crash:

- Impact between two flying UAVs
- Impact with a building
- Impact with other obstacles, such as trees, birds or power lines

Risk modelling of this kind of situations has always presented many complications, especially when aiming at a quantitative analysis, since it requires the knowledge of a lot of information very often impossible to know a priori, such as, for example, the exact trajectory that will be performed by all the other UAVs during their flights [8].
In this analysis, after a mid-air collision, the possible victims are the people on ground involved in the crash, particularly the ones that are hit by debris following the accident [7], so the number of people in the zone of the crash has to be considered in the risk evaluation. We can suppose that the latter has the same probability to cause deadly events as the whole UAV. In fact it is impossible to predict if and how it would break and in how many pieces and therefore the worst case scenario must be considered: a main debris with practically all the initial momentum of the drone. In addiction, if we suppose that the kinetic energy does not change after the mid-air collision, $P(fatality|exposure)$ does not vary as well.
According to [8], the frequency of fatalities due to in-flight accidents can be modelled as:

$$f_{F,MAC} = N_{exp} \times P(fatality|exposure) \times f_{MAC} \tag{2.10}$$

where:
$f_{F,MAC}$ = frequency of fatalities due to mid-air collisions. Unit of measurement: $[h^{-1}]$
N_{exp} = number of people exposed to the accident
$P(fatality|exposure)$ = probability that a person affected by the UAV crash will suffer fatal injuries
f_{MAC} = frequency of UAV mid-air collisions. Unit of measurement: $[h^{-1}]$

It is important to keep in mind that, in this accident type as well, we are still considering on-ground victims, so f_{Max}, coming from previous considerations remains valid as the maximum acceptable value of $f_{F,MAC}$ and it has to be evaluated with the same main criteria.
Since three out of four parameters of equation 2.10 have already been explained when referring to the previous formula 2.2, the next lines will be focused 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} is needed to have an estimation of the number of collisions that can happen during a mission. Once again, this parameter considers only crashes with other UAVs, buildings and others unpredictable obstacles, like trees, bird or power lines.
The most common technique to estimate the accidents rate is to use statistical methods based on a huge database of accidents happened in the last years.
The results is a very small number, in the order of $10^{-7} \frac{accidents}{h}$ [8][30]. An equation can be defined:

$$f_{MAC} = f_b + f_{UAV} + f_{TBE} \tag{2.11}$$

where:
f_b is the collisions rate with buildings. Unit of measurement: $[h^{-1}]$
f_{UAV} is the rate of collisions with other UASs flying in the same area. Unit of measurement: $[h^{-1}]$
f_{TBE} is a term to parametrize all the other unpredictable collisions. Unit of measurement: $[h^{-1}]$

The only way to estimate both f_b and f_{TBE} is referring to the statistics created after many hours of flight on the same area. The evaluation of f_{UAV} is the object of analysis of the next section.

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

Algorithms for collision avoidance are real time procedures implemented to allow two or more UAVs, whose trajectories are going to collide, to avoid each other [27]. Nevertheless, accidents are possible and it is important to model the frequency of the impacts between UASs, f_{UAV}.
To evaluate the number of collisions, the main difficulty is to be able to known all the trajectories of the UAVs over a given area, instant by instant.
Many different techniques have been developed to obtain a correct measure of f_{UAV}. According to [8] and [45] it can be expressed as:

$$f_{UAV} = E(CT) \times P(collision|CT) \tag{2.12}$$

where:
$E(CT)$ is the expected value of conflicting trajectories per hour in a given zone. Unit of measurement: $[\frac{trajectories}{h}]$
$P(collision|CT)$ 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 so called Gas Model [8] [45], that considers a volume of airspace V around the drone. In this model, the starting point is the knowledge of the exposed surface of every other aircraft that can impact the UAV. Startig from it, the volume that it will occupy during its trajectory is evaluated.

$$E(CT) = \frac{A_{exp} \times d}{V \times t} \tag{2.13}$$

where:
A_{exp} is the exposed surface of the drone. Unit of measurement: m^2
d is the distance that the aircraft will cover in the volume V. Unit of measurement: m
V is volume of airspace around the UAV. Unit of measurement: m^3
t is the time that the other aircraft will spend in the volume V. Unit of measurement: h

An easier way to understand this concept is to visualize it in image 2.5.
Nevertheless the Gas Model requires a precise knowledge of the exact trajectories of all the other UASs in the same area. This is almost impossible because, even if a centralized approach with a single Traffic Manager is used to plan all the trajectories, it still can not forecast all the possible real-time changes in the path, due, for example, to external unexpected events. Therefore the Gas Model turns out to be ineffective in the vast majority of cases [8].
Another approach to evaluate $E(CT)$ is based on the "statistical worst case analysis", and it is simple and conservative. It is based on the knowledge of how busy a given area in the time window of the mission is and it allows to provide a coherent value for $E(CT)$. Simulations ([8]) have defined a typical value for highly busy areas to be $E(CT) = 4 \times 10^{-5} \frac{CT}{h}$. However, this value may change a lot in function of the dimensions both of the considered airspace and of the UAS.
The other term of the equation 2.12, $P(collision|CT)$, was originally introduced to take into account the possibility that one ore both the conflicting UAVs exploit on board collision avoidance algorithms [8]. However, from our point of view, it is difficult to know which algorithms are implemented on the other UASs and the lone possibility is to focus only on the UAV that is performing the mission whose risk we want to evaluate. We could assume $P(collision|CT) = 0$ when the anti collision systems are implemented and $P(collision|CT) = 1$ otherwise [27].

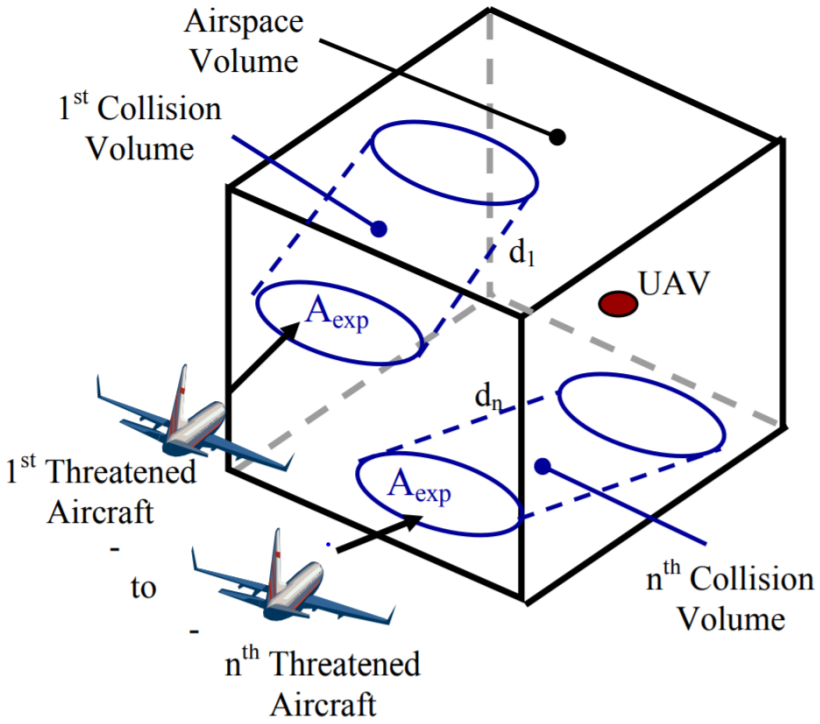


Figure 2.5: Gas model application[27]

2.4 Ground Impact Risk Modelling: A Complete Model

So far in this chapter, we have analysed the two accident types that can lead to a ground crash of a UAV: mid-air collision and early flight termination. The crash of the vehicle is a threat for people possibly affected, and it can result in injuries or fatalities.
For each of the two hazards, the risk assessment procedure is able to compute a value that represents the risk for people, depending on the characteristics of the area and on the map resolution.
The last step to complete the risk assessment process consists in merging the two metrics to provide just one value able to describe completely the ground impact risk. This process is fairly simple, provided that we make a specific assumption: in every area, only one of the two kinds of accident can happen. This basically means that the ground impact can be either caused by an in-flight lost of control or by a flight's collision, not by both of them. In order to build a conservative risk analysis, the higher risk value between the two must be chosen:

$$f_F = \max(f_{F,EFT}, f_{F,MAC}) \tag{2.14}$$

With reference to the specific equations 2.2 and 2.10 this can be written as:

$$f_F = \max(N_{exp} \times P(fatality|exposure) \times f_{EFT}, N_{exp} \times P(fatality|exposure) \times f_{MAC}) \tag{2.15}$$

and finally, since the first two terms are in common, it becomes:

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

Equation 2.16 can be considered the final ground impact risk formula, since it contains all the parameters of interest of both the scenarios. For what concerns our operative environment however, some considerations must be done, that strictly follow all the risk's discussion.
In this equation, f_{EFT} and f_{MAC} are the only two parameters that differentiate one hazard from the other. At this point, some final considerations with reference to the state of art have to be made about them:

- f_{EFT}, once again, is the frequency of ground impacts due to internal failure of the UAS, or in general to anything that can causes an unexpected end of the flight in an area not suitable for landing. Unfortunately this kind of events can be fairly common for small and cheap consumer drones like the ones that fly over cities. According to [19] the order of magnitude is around $f_{EFT} = 100^{-2}h^{-1}$, that means one accident every 100 hours of flight.

- f_{MAC} is the frequency of ground impact due to an in-flight collision between two UASs, or ,in general, between the aircraft we are considering and any kind of static or dynamic object. In fact, the main threats, as discussed in the dedicated section 2.3, are unexpected static obstacles on the trajectory or other UASs (dynamic elements). According to what previously stated, given a generic time instant, the order of magnitude of f_{MAC} is around $10^{-7}h^{-1}$.

As a consequence, since $f_{EFT} \gg f_{MAC}$ always, equation 2.16 can be further simplified omitting f_{MAC}:

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

This final equation, allows to provide an operative and quantitative definition of risk, in terms of number of victims per hour of flight, that respects all the standards of the most important national flight agencies. It is interesting to notice that, since the risk is defined as a frequency, it varies in the range $(0, +\inf)$.
Given such a theoretical background, the Cloud-Based UASs Traffic Manager[27] is able develop a complete risk assessment procedure, with the path planner that is finally in charge of guaranteeing the safety throughout the whole mission. In conclusion, as stated earlier in the chapter, it is possible to consider as safe all that missions that provides a number of fatal events per hour lower than $f_{F,Max} = 10^{-5}\frac{1}{h}$.

Chapter 3

Image processing and analysis

At the beginning of their history UAVs were mainly used for military purposes, as in the case, for example, of Queen Bee, the first returnable and reusable unmanned vehicle created in 1935 by the United Kingdom Royal Air Force. It was a radio-controlled biplane used as an aerial target during the training missions.
However, with the advancement of technology, UASs have become smaller and cheaper and now they are used in a large variety of fields. Certainly this usage expansion is also due to the increase of reliability and trustworthiness of the unmanned systems, that now are equipped with sensors for safety or to add new functionalities.
Some of these sensors are nowadays present on practically any drone to observe and control parameters that are crucial for the flight. It is the case of accelerometers, gyroscopes, GPS modules, inertial and tilt sensors, that are used to determine position, orientation, direction and altitude of flight. This data are helpful to monitor the conditions of drones in case they are piloted or to schedule the path of a mission if they are controlled by a machine.
UAVs may also have on board sensors aimed at checking self functionality, like current sensors to monitor and optimize the power drain, and, in the case of drones with gas engines, also an intake flow sensor to observe the air flow in the motors and to determine the proper fuel-to-air ratio at a specified engine speed, in order to improve power and efficiency and to reduce emissions [47].
With the development of new technologies for sensors, the customization of UASs is becoming cheaper and in the next few years this will lead to a lot of completely new possibilities and functionalities in many niche spaces. Nowadays drones can already be equipped, for example, with thermal cameras that can be used for for different tasks, like building diagnostics, security and rescue, surveillance, firefighting and wildlife conservation. On the other hand, multispectral cameras are able to capture images in a variety of wavelengths, including Green, Red, Red-Edge and Near-Infrared, and can be exploited in precision agriculture to collect data on plant and vegetation, that can be useful for farmers to optimize the use of irrigation and pesticides, for example. The number of different sensors and their corresponding possible benefits is huge and it is going to increase in

the next years. This customizability of the UASs, in fact, will probably lead, in the near future, to their exploitation in a large variety of fields. According to [2], the impact of commercial drones could reach 82 billions of dollars and give a 100.000 jobs boost to the U.S. economy by 2025.
The most widespread sensors for drones nowadays are certainly obstacle avoidance ones and cameras. The formers, quite self explaining, are used to avoid obstacles and crashes. This feature is so important for safety that it even reduces the insurance costs for drone's owners. In fact, crashing a drone on a tree or a building could damage or destroy it, but doing it on a person could have catastrophic consequences, as seen in the previous chapters. This kind of sensors may exploit different technologies like:

- Stereo Vision
- Ultrasonic (Sonar)
- Time-of-Flight
- Lidar
- Infrared
- Monocular Vision

All these use different approaches for the same purpose of detecting possible obstacles to avoid collisions. Data from different sensors can be merged by the flight controller, that actually runs obstacle detection software and algorithms, to improve the accuracy. As a result, drones equipped with this kind of sensors can fly indoors or even in situations in which safety is critical, like concerts or events where a lot of people is exposed.
The development in this field is fast and extremely important, since many companies, like Amazon and UPS, are planning to use in the next years unmanned aerial vehicles to autonomously deliver parcels to their customers' doors. As a matter of fact, UASs could increase work efficiency and productivity, improve accuracy while decreasing, at the same time, the workload for humans [23]. In order to achieve this result, once again, safety is crucial and obstacle avoidance is central in this sense.
The vast majority of UAVs is equipped with cameras as well. In fact in the last decades technology has allowed to build smaller and lighter cameras, keeping or even improving image quality. Their strength is the compactness that makes them extremely easy to transport without exceeding UAS payload.
The development of advanced algorithms of image analysis lead to the use of cameras for many different purposes. One of the most interesting is orthophotography, thanks to whom it is possible to create extremely detailed and accurate maps. In fact, during its flight, a drone can take large quantities of images with a constant scale and reference to ground control points. These images can then be processed by photogrammetry software to produce high-resolution maps on which very accurate measurements can be made.
A monocular camera can even be used, as stated before, with size expansion algorithms

to perform obstacle identification and avoidance, differently from what can be made with the more traditional and human-like stereo vision [1].
However in consumer drones the camera is mainly used for aerial photography and videography. According to [23], 42.9% of UAV market is nowadays dedicated to this kind of imaginary and probably this percentage will increase once the technology will be settled and proper regulation standards will be issued.
Cameras to be mounted on drones, despite being small and lightweight, are able to capture an exceptional image quality, thanks to sensors with a definition up to 6K and the capability to shoot in the uncompressed RAW format. These cameras are usually mounted on gimbals, that are cardanic motorized joints able to compensate the fast movements of the drone to eliminate vibration and obtain smooth video or sharp pictures. UAVs like DJI Inspire even allows to change lenses to obtain different optical effects and therefore they are used in the cinematographic industry nowadays. In fact filming with a drone instead of a camera on an helicopter is definitely less demanding from an economical point of view, it is more flexible and it guarantees a precision and repeatability of the camera movements that cannot be obtained in any other way.
Moreover these cameras can be triggered remotely from the Ground Control Station or simply using a device like a tablet or a smartphone that also give a first person view thanks to a real time video streaming.
The aim of this thesis is to use satellite images and the cloud connection of the UAV to the CBUTM to create a tool able to derive the sheltering factor P_S from the images taken by the drone. However the presence of a on-board camera opens new possibilities for future developments where these pictures are taken by the UAV itself and sent to the CBUTM via cloud to have a real-time response.
The first step is to build an initial data set of aerial images and assign the sheltering factor values to each of them by inspection following the considerations made in Chapter 2. A comparison between these values and the ones obtained through image analysis, will allow to find a precise univocal function linking them. In particular this will be obtained through a neural network fitting process and it will finally lead to the definition of the sheltering factor of a certain area starting from an aerial image of it.

3.1 Image processing

3.1.1 Sheltering factor assignment

The first step to perform consists in building an initial data set of images to work on and, to do that, we have chosen to use satellite pictures of an urban scenario. This kind of environment is the most interesting and challenging for risk modelling since it offers various areas with different sheltering ability. In fact it is clear, from what stated in Chapter 2, that trees, parks, buildings and squares, for example, do not offer the same level of protection to the population. Cities are so densely populated and so interesting for UAVs'

missions that finding the sheltering offered by the elements of the environment is crucial to be compliant with the safety standards, and to have, as a consequence, the permission to fly and perform the assigned task.
Therefore the initial set of images we worked on is composed by 200 satellite pictures of an urban environment and in particular, the vast majority of them show areas of the city of Turin, Italy.
To identify the different zones in an image and to assign them a corresponding sheltering factor, we have thought to divide every picture in smaller sub-parts. In fact, working on little areas allows to have a more precise identification of the different zones, like it would happen to a camera increasing its resolution, for example. In theory the smaller the area, the more precise is this process.
We have divided each original image, from now on denoted as A, in 160 sub-parts, with a 10×16 grid. The surface in frame is around $4000m^2$ and the one of each sub-part is $25m^2$, that is a reasonable value that allows to have a good accuracy. In addition to that, according to [27], an area of $25m^2$ respects the typical flight characteristics of an urban UAV, guaranteeing its manoeuvrability.
As explained in Chapter 2, we have decided to use a scale for the sheltering factor formed by ten levels. It is shown in Table 2.3. Following what prescribed there, we assigned a sheltering factor value to each of the sub-parts by visual inspection. This process is long and, despite the small surface taken into account, it can lead to uncertainties in some situations, especially on the edges of the different zones, like the side of the buildings, for example. In this kind of cases a conservative decision could be taken [38], as we have done, but it is important too notice that it could eventually lead to overconservative results. Once again, the sheltering estimation is, in general, quite complex, especially in this case, where the choice to be made in this phase will have a big impact on the error computation and on the training of the neural network.
At the end of the assignment each satellite image has a 10×16 matrix associated, containing all the estimated sheltering values for each of its sub-parts.Figure 3.1 shows the definition of the sheltering matrix by inspection of an example image. In the next lines we are referring to these matrices as $\mathbf{M}$.
Each of these $\mathbf{M}$s is defined basically starting from the observation of the corresponding image A and it represents the sheltering values associated to that very image. Therefore we could write generally that $\mathbf{M}$ is a function of A:

$$\mathbf{M} = f(A) \tag{3.1}$$

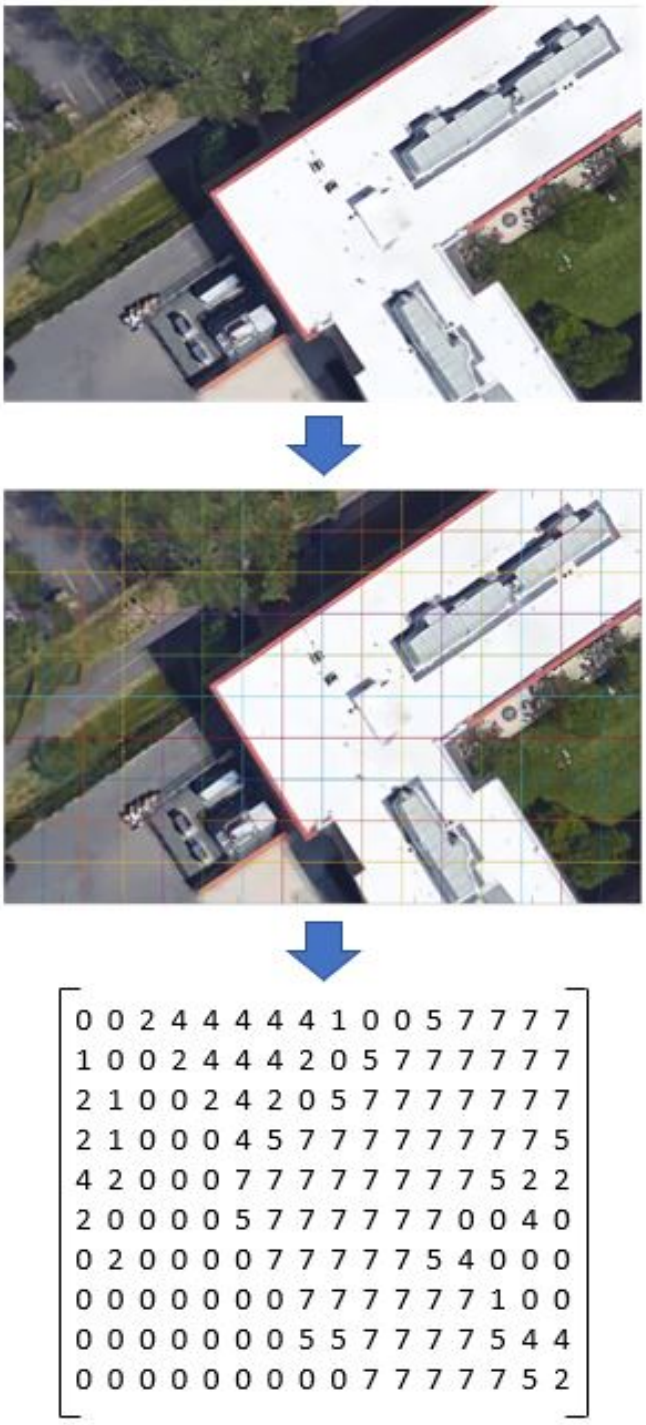


Figure 3.1: Example of image division and definition of the sheltering matrix **M** by inspection

3.1.2 Image Analysis

The RGB colour model

The purpose of this work is to build a tool able, given an aerial image, to recognize the zones with different sheltering ability and to assign each one a corresponding and coherent sheltering factor. In order to do that, pictures have to be explored to find specific features or characteristics that can identify those zones. To perform feature extraction we have thought to use colour analysis as a first step and so we have had the need to find a colour framework to be based on.
Colour models are abstract mathematical models describing how colours can be represented as values, typically three or four, that are the so called *components*. A certain quantity of each of these define precisely how the corresponding cromaticity looks and this is important to establish a unique definition for each colour. Many models have been proposed, and each one has a specific task and a different mechanism: some of them are based on the addiction of components, some on their subtraction, some on specific combinations of hue, saturation and brightness. The most important and used ones are:

- RGB
- CMYK
- L*a*b (or CIELAB)
- HSV
- HSL
- NCS

Since the most important devices that perform image sensing or displaying, like televisions, projectors, cameras, scanners and screens in general, are based on the RGB colour space we have decided to exploit this one, that is, in fact, the most famous and used colour model. RGB is based on the three chromaticities of red, green and blue (whose initials gives the name to the model itself), that are primaries and so can be added together to reproduce any colour. It exploits the trichromaticism of human eye and that is why it seems the most natural choice for sensing, representing and displaying images.
In particular, any chromaticity can be obtained by an RGB triplet, that means mixing in a certain quantity red, green and blue and it is said to have a *component* of each of the three primary colours. Each component can vary from zero to a maximum value: if all of the three are equal to zero, the resulting colour is black, while they are all to the maximum, the result is the brightest representable white. In the middle, with a combination of the three values, stand all the other colours. These range can be expressed in many ways but in general in informatics the components are integer numbers from 0 to 255, that is the scale that a single 8-bit byte offers. This is exploited, for example, by screens that, as a matter of fact, use special RGB pixels, which are formed by a red, a green and

a blue led side by side. Depending on which and how intensely each one of these lights up, the whole colour spectrum can be represented. A typical RGB pixel can be noticed in Figure 3.2.

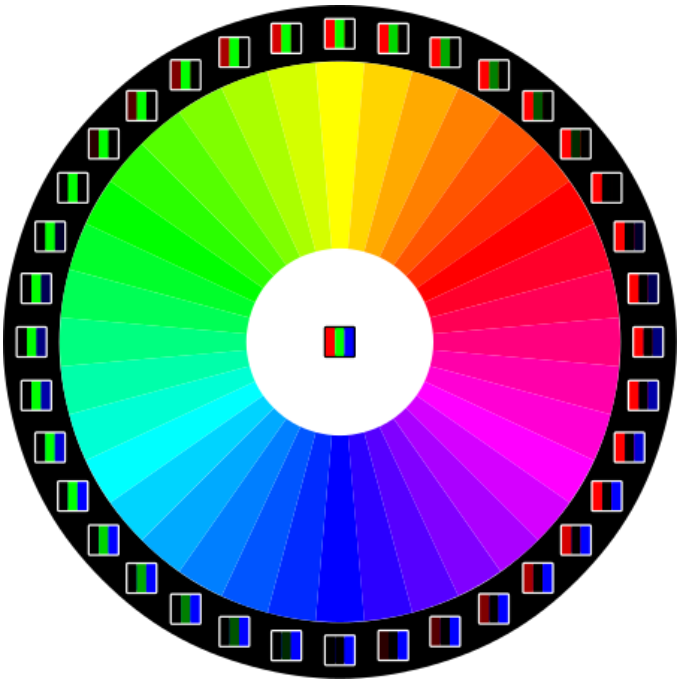


Figure 3.2: Colour wheel with screens' RGB pixels

Moreover the RGB colour model is very used also for image representation in particular by cameras like the one a UAV may be equipped with, and since the whole thesis is centred on aerial photography, using RGB colour model since the beginning could simplify a possible addition of new features and algorithms that exploit the on-board camera. Therefore, from now on, every colour will be classified and later identified by its RGB components.

Image processing

At this point we have 200 aerial images, divided in 160 sub-parts each, and 200 matrices containing a sheltering factor value for each sub part, given by visual inspection following the rules exposed in Table 2.3.
The purpose of the Image Processing phase is to decode the pictures and extract their features.
In order to do that, we have decided to exploit the possibilities given by MATLAB.
MATLAB (MATrix LABoratory) is a multi-paradigm numerical computing environment developed by MathWorks that exploits a matrix-based language to create functions, models, to analyse data and develop algorithms with great versatility. A different approach

or programming language, like C++ or Python could have been chosen to build the algorithm, but the big advantage MATLAB offers, with respect to other software, is indeed its versatility and modularity, with an high number of add-ons that expand its capabilities to nearly any type of analysis. Thanks to this additional packages, like Simulink, for example, it is possible to perform model-based design, as well as deal with robotics or control system problems. Machine learning and even deep learning algorithms can be developed in this environment, to deal with advanced problems, like computer vision or image recognition, or, as we are explaining in this work, with pattern recognition and function fitting. Moreover MATLAB algorithms can be automatically converted into other languages to be further developed or to be directly uploaded on embedded devices and this is an interesting feature for possible future developments in general. The software, an addition, offers a great scalability, allowing with only few modifications to size the analysis on a GPU, cluster or cloud level. Given the cloud-based nature of this project and the possibility of developing the whole of it in a single environment, MATLAB seems the most suitable tool to pursue the objective.

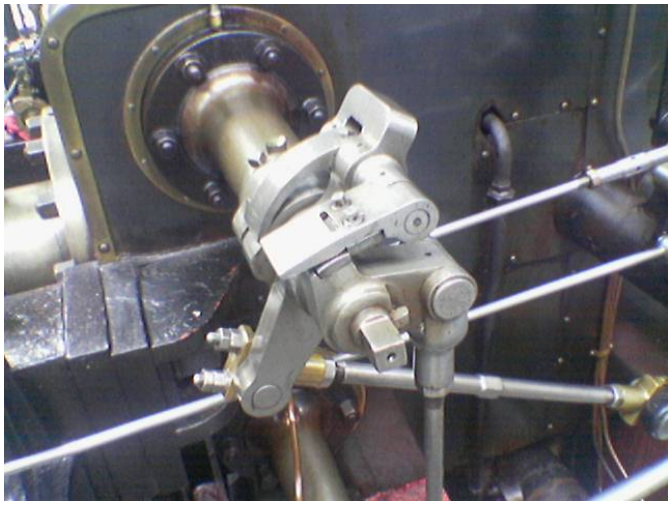
The purpose of this part of the work is to examine all the images of the data set to find some characteristic features that can be used in a quantitative way for classification. In fact the idea is that areas with the same, or very similar, features could identify the same kind of structure. For example a red area has an high probability of indicating the roof of a building, while a green one may suggest the presence of a meadow or woods. All these three example situations present different sheltering abilities, so being able to identify all the different zones in a picture is the first step to extract their sheltering factor from it.

To reach this result, a MATLAB script has been created to upload the aerial images and analyse them, one-by-one. We found as a matter of fact that the best way to identify the colour of each pixel is to know its three RGB components. In order to do that, each image A has been split up in three different ones, containing respectively only the information about the red, the green and the blue.

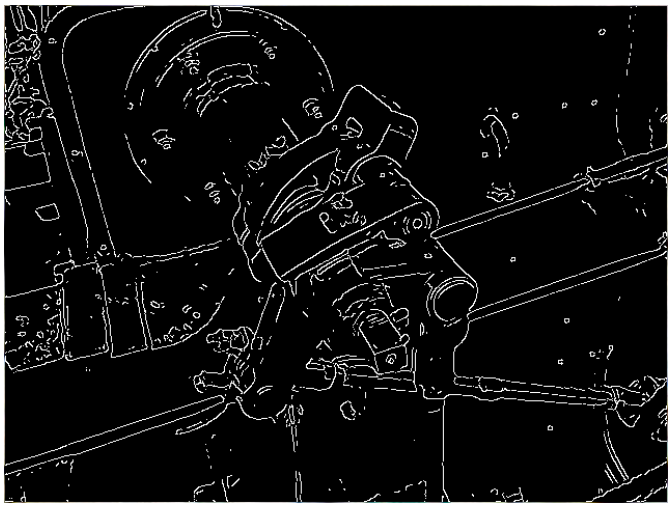
As previously said, every image is divided in 160 sub-parts; each of these is explored pixel by pixel to find the RGB components and a mean value of red, green and blue is computed for any region, to have approximately a representation of the quantity of those three colours in that specific sub-part. To keep track of them, the RGB mean values for each one of the original images (A) are collected in a tridimesional $10 \times 16 \times 3$ matrix, where the first layer (named R_mean) contains the information related to the red component, the second (G_mean) to the green and the third (B_mean) to the blue one.

To gather more information about the pictures we have used a MATLAB function, named *edge*. It uses the Canny approximation to find edges, that are points in an image at which the brightness has discontinuities. The specific algorithm we used applies to the image a gaussian filter to remove noise to prepare it for a more precise edge identification, it finds the intensity gradients using the Sobel approximation to the derivative, that finds edges where the gradient is maximum, then it performs non-maximum suppression, to mark as possible edges only the local maxima, discarding the other pixels. The last step the Canny algorithm performs is to actually track the edges through hysteresis thresholding:

given two threshold values, *min* and *max*, any edge with intensity gradient higher than *max* is labelled as "sure-edge", while the ones lower than *min* are marked as "non-edge" and discarted. All the values in between the two thresholds are labelled one way or the other according to their connectivity: if they are connected to “sure-edge” pixels, they are considered to be edges, otherwise they are discarded as well [5].
Edge detection algorithms are a fundamental tool in image processing and computer vision, since by reducing an image to its edges, it becomes much easier for algorithms to identify or process it. In fact an edge usually may correspond to a discontinuity in depth, illumination or surface orientation or colour, or even to a change in material property.
Figure 3.3b shows the effects of the Canny edge detector on image 3.3a.



(a) Original image



(b) Image with Canny edge detector applied

Figure 3.3: Edge detection

Practically, when applied to a picture, the MATLAB function *edge* returns a black and white binary image, with the same size of the original one, with 1's where it finds edges and 0 elsewhere. It is used, in this case, to identify objects, like trees or the edges of the roof of a building and therefore we have exploited it to add information that can be useful for the aforementioned classification of the areas of each image A. The analysis with the Canny edge detection algorithm is carried out as well pixel-by-pixel singularly on the layers of the matrix containing red, green and blue components. Just like we have done previously, the average value of edges for each sub-part is computed to obtain, at the end, 160 values for each colour of each image.
The whole process of image fragmentation, colour analysis and edge detection is schematised in Figure 3.4.

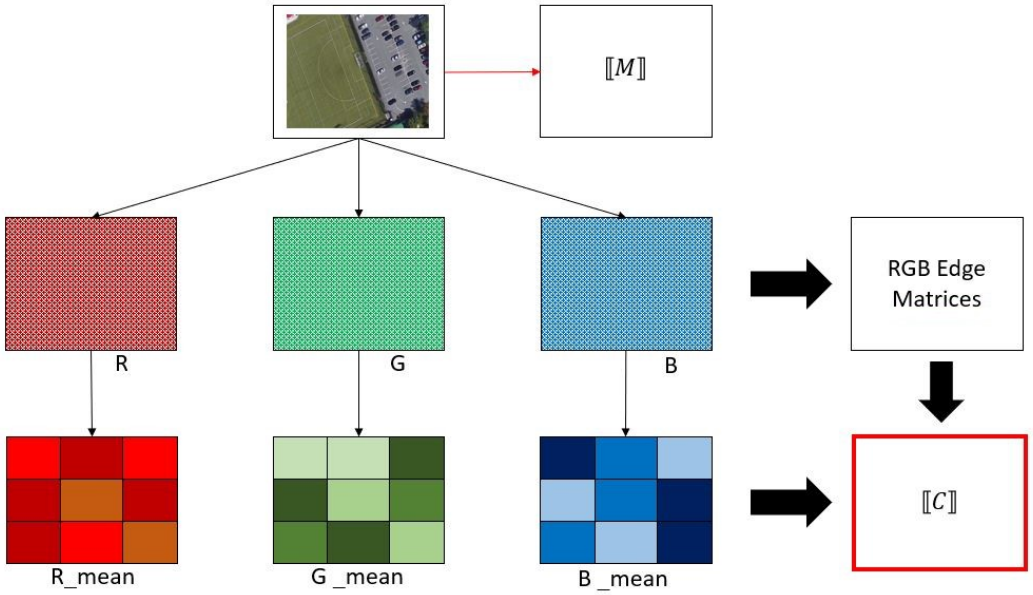
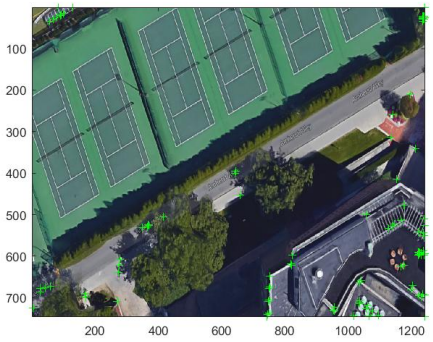


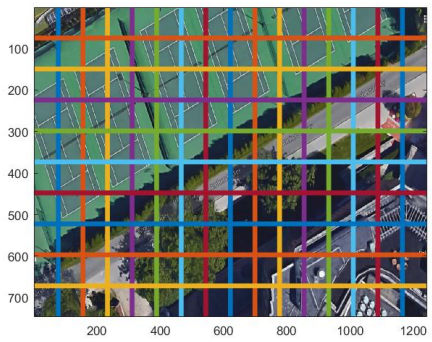
Figure 3.4: Image processing scheme

The steps made so far may seem confusing and difficult to visualize, so, to summarize them, an is shown in Figure 3.5. In this case the original picture A (3.5a) represents a typical urban scenario with a building, a street, tennis courts and few trees. At the beginning it is divided in 160 sub-parts (3.5b) with a 10×16 grid. Each of these sub-parts is examined by inspection to be assigned a sheltering factor in agreement with the scale chosen (Table 2.3) and all these values are collected in a matrix, called $\mathbf{M}$. In this example picture, only the building and trees have a sheltering ability, while the street and the tennis courts are free areas. In fact, if matrix $\mathbf{M}$ is represented in a colour scale for clarity, the obtained result is Figure 3.5c, where the different zones are clearly visible.

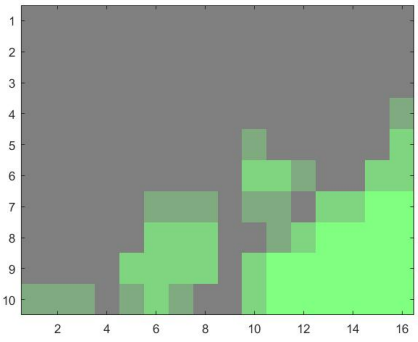
Figure 3.5d, on the other hand, shows, in a colour scale again, the mean values of edges found with Canny approximation, in this case, for the green component of image A (green is used just an example, the procedure for red and blue are exactly the same).



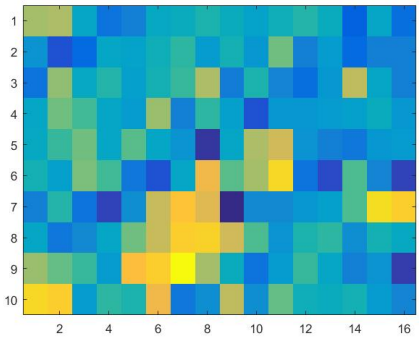
(a) Original image A



(b) Image division



(c) Sheltering matrix M



(d) Edge matrix for green

Figure 3.5

When analysing a satellite picture, shadows can be a problem, since they inevitably alter the colour of the area they are projected on and can even hide objects or structures whose associated sheltering factor has to be evaluated instead, for a complete risk map. Many image processing functions have been explored to limit these effects and to improve image analysis, but, despite using the uncompressed *.png* format for the pictures, no one of them was able to reduce the problem without completely modifying or even ruining the image. Therefore we have decided to use another approach. As a matter of fact, once again, it is important to underline that in a tool like the one we are making, based on the sheltering factor derivation starting from a picture, it is important that image analysis is not deceived by any unpredictable alteration of colour, due, as said, to shadows but also, on the contrary, to any object that could be interpreted as a shelter while it is not.
To limit these possible unwanted effects, the algorithm we have written computes the average values of red, green and blue in each sub-part, but in order to further improve the

analysis we have decided to extend the context even more. In fact we could assume that the sheltering factor matrix **M** is function of the RGB values not only of the single sub-part, but also of its neighbourhood (9 values per colour in total). Of course, as previously said, it is also function of the average edge values for red, green and blue of that specific portion of the image.
For example, if we consider a generic sub-part (x,y) of image A, with reference to equation 3.1 we can write:

$R_{x-1,y-1}$	$R_{x-1,y}$	$R_{x-1,y+1}$
$R_{x,y-1}$	$R_{x,y}$	$R_{x,y+1}$
$R_{x+1,y-1}$	$R_{x+1,y}$	$R_{x+1,y+1}$

$G_{x-1,y-1}$	$G_{x-1,y}$	$G_{x-1,y+1}$
$G_{x,y-1}$	$G_{x,y}$	$G_{x,y+1}$
$G_{x+1,y-1}$	$G_{x+1,y}$	$G_{x+1,y+1}$

$B_{x-1,y-1}$	$B_{x-1,y}$	$B_{x-1,y+1}$
$B_{x,y-1}$	$B_{x,y}$	$B_{x,y+1}$
$B_{x+1,y-1}$	$B_{x+1,y}$	$B_{x+1,y+1}$

$$M_{x,y} = f(R_{x-1,y-1}, R_{x-1,y}, R_{x-1,y+1}, R_{x,y-1}, R_{x,y}, R_{x,y+1,}, R_{x+1,y-1}, R_{x+1,y}, R_{x+1,y+1} \\ G_{x-1,y-1}, G_{x-1,y}, G_{x-1,y+1}, G_{x,y-1}, G_{x,y}, G_{x,y+1,}, G_{x+1,y-1}, G_{x+1,y}, G_{x+1,y+1}, \\ B_{x-1,y-1}, B_{x-1,y}, B_{x-1,y+1}, B_{x,y-1}, B_{x,y}, B_{x,y+1,}, B_{x+1,y-1}, B_{x+1,y}, B_{x+1,y+1}, \\ red\_edge_{x,y}, green\_edge_{x,y}, blue\_edge_{x,y}) \tag{3.2}$$

Equation 3.2 states that the sheltering factor of a generic sub-part of the original image A can be expressed as function of 30 parameters, 27 coming from the colour analysis of its neighbourhood and 3 from edge detection applied to its red, green and blue components. However, finding all the adjoining values for a sub-part located on the border of the image is impossible, since at least three elements would be located outside the picture and their data would consequently be missing. Therefore in order to have an even evaluation we have thought to expand the matrices containing the average RGB values with a specific function: the external layer has been "doubled" to have nine values per colour for each sub-part neighbourhood. This, of course, is an approximation since we are supposing that the part that is just outside the shot has the same characteristics as the one of the external layer of the picture. Another idea could have been simply not consider the outer layer, starting the analysis from the penultimate, but we have decided to use this approximation to avoid losing any information and to have a 100% coverage of the image.
We have then created a new matrix, denoted as **C**, containing all the RGB mean values for each sub-part and for its neighbourhood and the values coming from the edge analysis for each colour, in a specific order. Again, the overall scheme of the process is shown in Figure 3.4.
Therefore if we consider a generic sub-part (x,y) of image A , we can identify the corresponding row of the **C** matrix as:

$$\begin{aligned}\mathbf{C}_{x,y} = [&R_{x-1,y-1}, R_{x-1,y}, R_{x-1,y+1}, R_{x,y-1}, R_{x,y}, R_{x,y+1,}, R_{x+1,y-1}, R_{x+1,y}, R_{x+1,y+1},\\ &G_{x-1,y-1}, G_{x-1,y}, G_{x-1,y+1}, G_{x,y-1}, G_{x,y}, G_{x,y+1,}, G_{x+1,y-1}, G_{x+1,y}, G_{x+1,y+1},\\ &B_{x-1,y-1}, B_{x-1,y}, B_{x-1,y+1}, B_{x,y-1}, B_{x,y}, B_{x,y+1,}, B_{x+1,y-1}, B_{x+1,y}, B_{x+1,y+1},\\ &red\_edge_{x,y}, green\_edge_{x,y}, blue\_edge_{x,y}]\end{aligned} \tag{3.3}$$

To resume, at this point of the work, every original satellite image A has a corresponding $\mathbf{C}$ coefficient matrix, that contains all the information on edges and colours of any of its sub-parts, and a corresponding $\mathbf{M}$ matrix, that stores all the associated sheltering values given by inspection. Since all the original pictures have been divided with a 10×16 grid (see section 3.1.1), $\mathbf{M}$ is obviously a 10×16 matrix, while $\mathbf{C}$ has dimensions 160×30.

Chapter 4

Linear Regression

As seen in in the previous chapter in equation 3.2, the sheltering matrix $\mathbf{M}$ depends on the same parameters as $\mathbf{C}$, but while the former is composed by values we have assigned by inspection (see section 3.1.1), the latter carries the picture information related to colours and edge detection.
The idea is to compare these two kind of matrices and find the error. The first method we could think of is linear regression, a linear approach to model the relationship between a dependent and one or more independent variables. It is normally used for:

- Determination of the strength of predictors
- Forecasting of an effect
- Trend forecasting

We want to use regression analysis as a predictive model, to find the trend between the image analysis and the sheltering factor matrices and make predictions on new situations, based on the fitted model created in that way. Since, for each sub-part, we are associating 30 values coming from colour and edge analysis, hence predictors, to one value of sheltering factor, we are using multiple linear regression.
To build the predictive model we have decided to compare the whole set of pictures at once to have a more representative result. To do that we have created a matrix $\mathbf{C}_{tot}$ composed by all the $\mathbf{C}$ matrices associated to any image, in the specific order from 1 to 200. Since the dimension of every $\mathbf{C}$ is 160x30, $\mathbf{C}_{tot}$ is a 32000x30 matrix. Moreover, in a similar way, we have merged all the $\mathbf{M}$ matrices as well and combined the 32000 sheltering values in a $\mathbf{m}$ column vector, keeping the exact same order.
In this way, given a generic $i \in [1, 32000]$, the i-th value of $\mathbf{m}$ and the i-th row of $\mathbf{C}_{tot}$ would refer to the same sub-part of the same image, keeping track respectively of the sheltering value we have assigned and of the colour and edge information of that region.
At this point, since $\mathbf{C}_{tot}$ is not a square matrix, we have to use its pseudo-inverse to find:

$$\mathbf{m}{\mathbf{C}_{tot}}^{+} = \mathbf{p} \tag{4.1}$$

Using $\mathbf{p}$, we can now compute the error vector $\mathbf{e}$

$$\mathbf{m} - \mathbf{C}_{tot}\mathbf{p} = \mathbf{e} \tag{4.2}$$

The error vector is composed of 32000 elements and it represents the difference between our estimation and the one coming from image analysis for each sub-part of every picture. In the graph in figure 4.1 the distribution of the errors in the whole image set can be observed. To have a clearer eveluation of the performance of the method we have computed the mean error, the variance and the mean absolute error and the results are reported in Table 4.1.

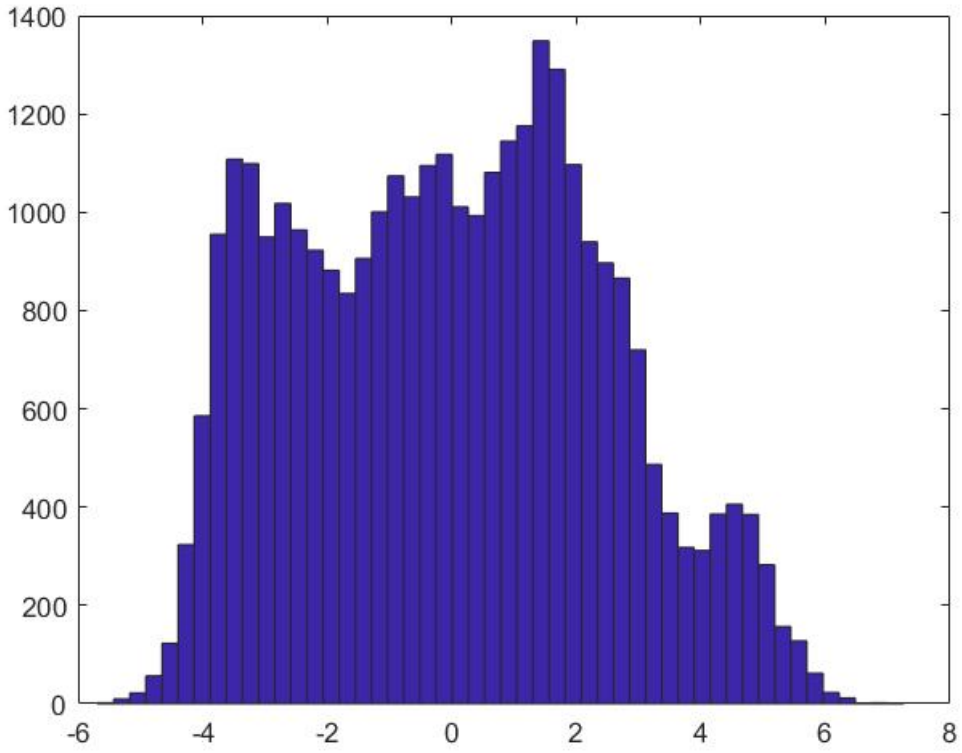


Figure 4.1: Error distribution

Sheltering Factor Mean Error	-0.0126
Variance	6.0734
Mean Absolute Error	2.0754

Table 4.1: Results of Linear Regression

This trend is more even and distributed than we could have expected, but it is approximately centred in zero, with local maxima in +2 and -3.
To make an example we could try to use this technique on one sample picture representing a typical situation in an urban scenario.The image has three zones with very different sheltering ability: the buildings, the trees and the yard. To evaluate the performance of linear regression we have to make a comparison between the obtained value and the desired one. To do that we have assigned by inspection the sheltering factor following the classification that we have established in Table 2.3. The resulting **M** matrix can be plotted in a colour scale for an easier and clearer comprehension and it is represented in Figure 4.3.

$\mathbf{C}_{tot}\mathbf{p}$ is 160 elements vector, that can be re-arranged in a 10×16 grid to obtain the

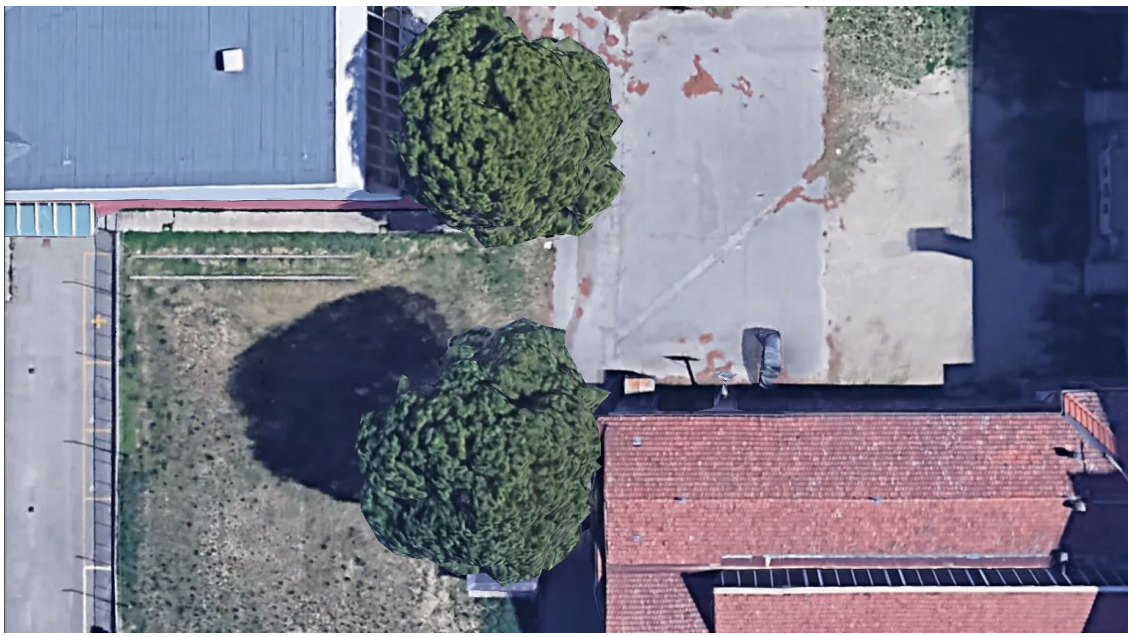


Figure 4.2: Aerial Picture 1

matrix, dimensionally equivalent to **M**. The matrix obtained in this way can be plotted in the same colour scale as the previous to have an immediate comparison of the result (Figure 4.4)

The histogram representing the distribution of the errors between the desired values of sheltering factor and the values obtained with linear regression is shown in Figure 4.5. To evaluate the performance of linear regression in this example, the mean error, the variance and the mean absoolute error have been computed and the results are reported in Table 4.2. The mean absolute error is higher than 2 both in the example and in the general case considering all 200 images (32000 sub-parts in total). It is twice the quantization level and such a bad result can not lead to a reliable map generation tool. Therefore we have decided to abandon the linear regression approach and to use, instead, machine learning

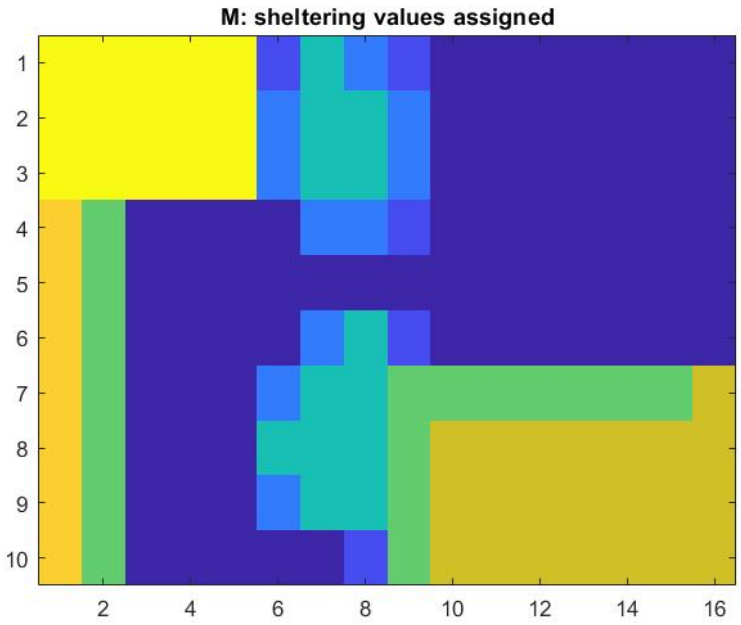


Figure 4.3: Sheltering map assigned by inspection of Figure 4.2

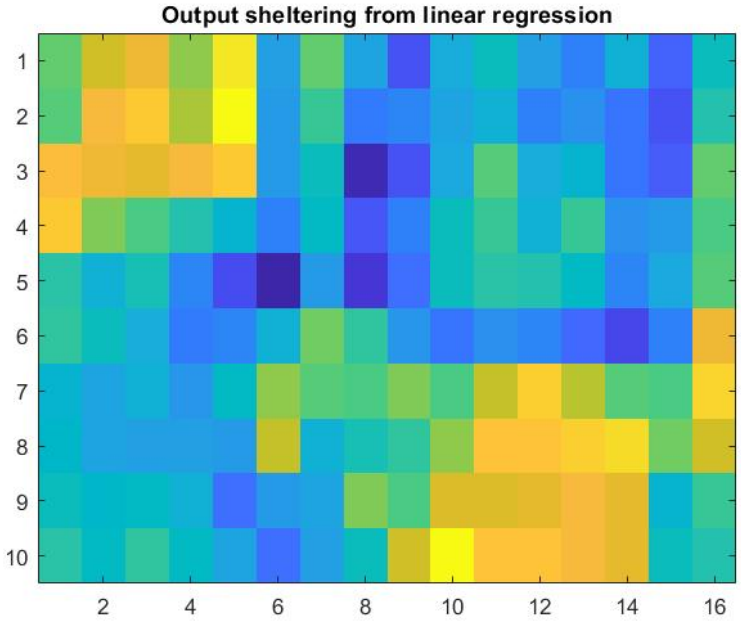


Figure 4.4: Sheltering map obtained with linear regression applied to Figure 4.2

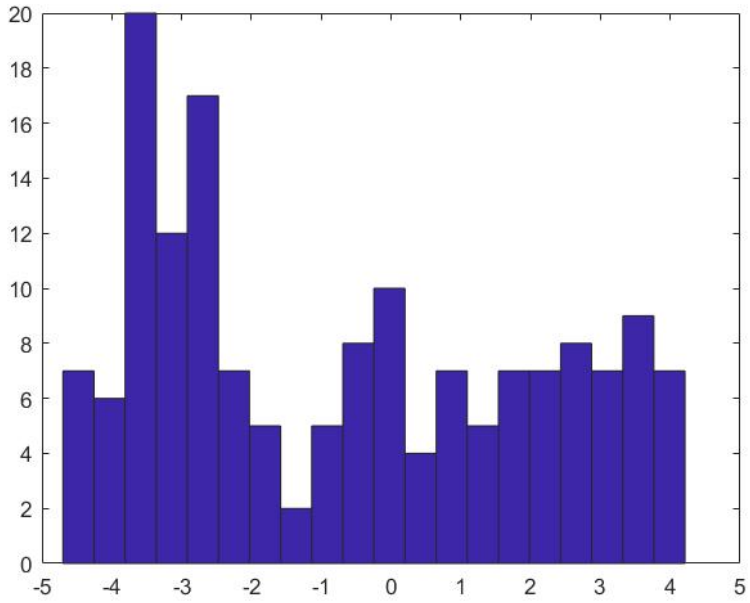


Figure 4.5: Distribution of the errors

Sheltering Factor Mean Error	-0.6344
Variance	7.2219
Mean Absolute Error	2.4473

Table 4.2: Results of Linear Regression

to model the relationship between image analysis data and the desired sheltering factor. Hence the next chapter will introduce the activity we have made on the neural networks.

Chapter 5

Neural network

Artificial Neural Networks (ANNs) are computing systems biologically inspired to human brain and intended to replicate its learning ability.
They consist of input and output layers and in most cases at least one hidden layer formed by interconnected artificial neurons. These are able to transform the inputs coming from the first layer into an output that can be used for different kind of tasks. In fact in the last decades algorithms have improved so much that currently ANNs provide the best solutions in problems like image and speech recognition, classification, predictive analysis and they have even been used with good results in medical diagnosis. In general they are excellent tools to find patterns in problems that are too complex or big for a traditional solution.
The ANN we are using is created and trained in MATLAB neural network toolbox, called *nnstart*. It allows to build a rather simple network that yet exploits the computational power of the software and is, in principle, able to automate the definition of the sheltering factor starting from a picture.

5.1 The neural network toolbox

The neural network toolbox is started by the command *nnstart* and it allows to create and train an artificial neural network to work on different kind of problems, including pattern recognition and classification, clustering of data and prediction based on a set of past values.
The application that suits in the best way the analysis that has to be carried out is the input-output and curve fitting one, *nftool*. In this case the neural network is trained through supervised learning of a set of input-output data to identify and approximate a relation between them. Once this relation has been modeled to the necessary accuracy, the ANN can be used for a variety of tasks, such as series prediction, function approximation, and function optimization. Due to their modular and non-linear nature, even small and simple neural networks are normally able to solve function fitting problems in an effective

way and this is the main reason that led to the decision of using one.
The toolbox can be used both through the graphical user interface and the command-line functions but for a matter of clarity it is generally better to start with the GUI and then use it to automatically generate the script.
The neural network is a two-layer feed-forward network with sigmoid hidden neurons and a linear output neuron. As said it is rather simple but yet normally very effective since it can fit even multi-dimensional mapping problems, given proper settings (data and high enough number of neurons in its hidden layer).

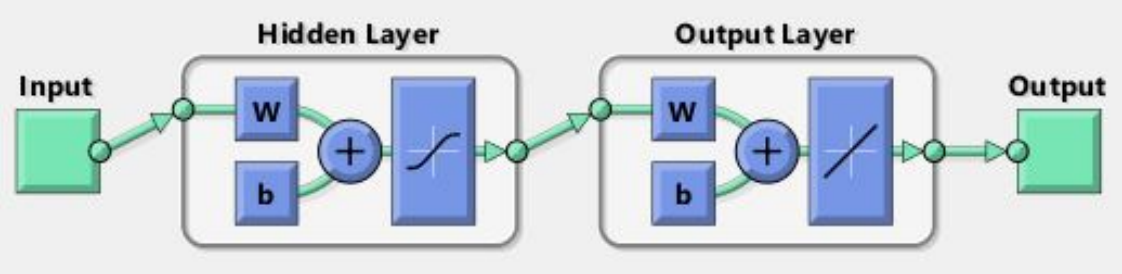


Figure 5.1: The neural network

To get started using the toolbox the first step is to define the problem by selecting a data set, that is to say a set of input vectors and one of target vectors, representing the correct output of each input. These data are loaded from the workspace and since each sample of the latter corresponds to one of the former they must have at least one dimension in common.
The neural network divides the samples randomly in three groups, each used for one specific task: training, validation and testing.

Training These samples are the ones actually used to train the network, that is adjusted according to its error.

Validation These samples are used to measure network generalization, and to stop training when generalization stops improving.

Testing These samples are not used in the training phase so they have no effect on that. Instead they provide a measure of network performance both during and after training.

It is possible to assign a certain percentage of samples to each of these tasks and the decision may affect network performance heavily. In general it is better to have a high number of samples to train the network on, usually at least 70% , but finding a trade off among meaningful data, percentages and good results is delicate and may require a very high number of simulations.
It is important to underline once again that final results depend also on the randomness used by the neural network to pick and assign samples to each of the three tasks.

Once decided the percentages, the number of neurons of the hidden layer of the ANN has to be defined. This is another parameter that greatly affects the network performance: by varying the number of hidden neurons it is possible to control the accuracy level with respect to the noise level in the data. In particular, given a good training subset, increasing the number of neurons leads to an increase of the neural network mapping accuracy [22]. On the other hand a network that is too complex may fit not only the signal but also the noise; this effect is called *overfitting* and it may produce excessive variance, resulting in prediction far beyond the range of training data. In this case the network may behave as a memory bank that recalls perfectly the training set but it is not able to perform well on samples outside of it.
Therefore it is very difficult to define in a easy way the exact number of neurons in the hidden layer of a neural network. Many studies have been carried out ([26],[11]) but no univocal solution has been found. What is generally used are series of rules of thumb giving some constraints:

> The size of this [hidden] layer is somewhere between the input layer size and the output layer size. [4]
>
> The number of hidden neurons should be the size of the input layer plus the size of the output layer, multiplied by 2/3.
>
> The number of hidden neurons should be less than twice the size of the input layer. [3]

However these three rules are just some of the many known and used and they can be considered a starting point to define the boundaries of the hidden layer. The exact number of neurons in practice is found with a trial-and-error process.
The last step before training the network is to define the algorithm to be used. The fitting application has three of them and they all perform differently depending on problem complexity and dimension.

Levenberg-Marquardt It is one of the most used algorithm for solving generic curve fitting problems. It updates weight and bias values according to Levenberg-Marquardt optimization and it is the fastest backpropagation algorithm in the toolbox, although it does require more memory than others. The method uses the Jacobian for computation, so the trained networks use either the mean or sum of squared errors performance function.
Validation is needed to stop training early if the network performance on the dedicated vectors fails to improve or remains the same for max_fail epochs in a row (by default max_fail=6).
It is also known as damped least-squares (DLS).

Bayesian Regularization It takes place within the Levenberg-Marquardt algorithm. Bayesian regularization minimizes a linear combination of squared errors and weights

and at the end of the training it modifies it to improve the network generalization qualities. The method uses the Jacobian for computation, so the trained networks use either the mean or sum of squared errors performance function. With respect to ANNs trained with other algorithm they are more difficult to overtrain and overfit and, despite being generally slower, they require less memory than pure Levenberg-Marquardt trained networks. ANNs trained with Bayesian Regularization are more robust than standard back-propagation nets and can reduce or eliminate the need for validation, that, in fact, is disabled by default: training continues until an optimal combination of errors and weights is found.

Scaled Conjugate Gradient ANNs trained with this method update weight and bias values according to the scaled conjugate gradient method. The algorithm can train any network as long as its weight, net input, and transfer functions have derivative functions. Backpropagation is used to compute derivatives of performance with respect to the weight and bias variables. The scaled conjugate gradient algorithm is based on conjugate directions but this algorithm does not perform a line search at each iteration.
Validation is needed and it stops when a maximum number of repetitions is reached, the maximum time is exceeded, the performance gradient falls below a certain threshold or when the performance is optimized to the goal.

Once again, there is not a single best method for nonlinear optimization but it must be selected according to the characteristics of the problem to be solved. In general Levenberg-Marquardt algorithm is very used and it is a good base to start with since it usually has a good trade off between performance, memory request and time.
When the algorithm is picked, the network is ready to be trained. The status can be observed in real time through the neural network training tool in Figure 5.2. This window is the real core of the toolbox and it is particularly useful since it allows to monitor many parameters of the training process. Moreover it is possible to plot some data like the training state, error histogram and regression (like the ones in figures 5.3 and 5.4) to evaluate the performance of the neural network. In case the performance of the ANN is poor, it can be retrained: repeating the training multiple times will generate other results due to different initial conditions (weights and bias of the neurons) and sampling. The process can be repeated multiple times, both to find the best trade off among algorithms, number of neurons and sampling distribution and to find the best outcome for a given configuration.
When the results are satisfying the toolbox allows to automatically generate a command-line script that reproduces the network and allows to make changes and tune all its parameters. It is also possible to extract the ANN fitting function that approximates the input-output characteristic and can then be used for any other task. In our case, the cloud-based UAV traffic manager will use this resulting function to compute, from a satellite image, the sheltering map of the area in which the UAV mission has to be performed, it will check if the overall risk of the mission is compliant with the standards

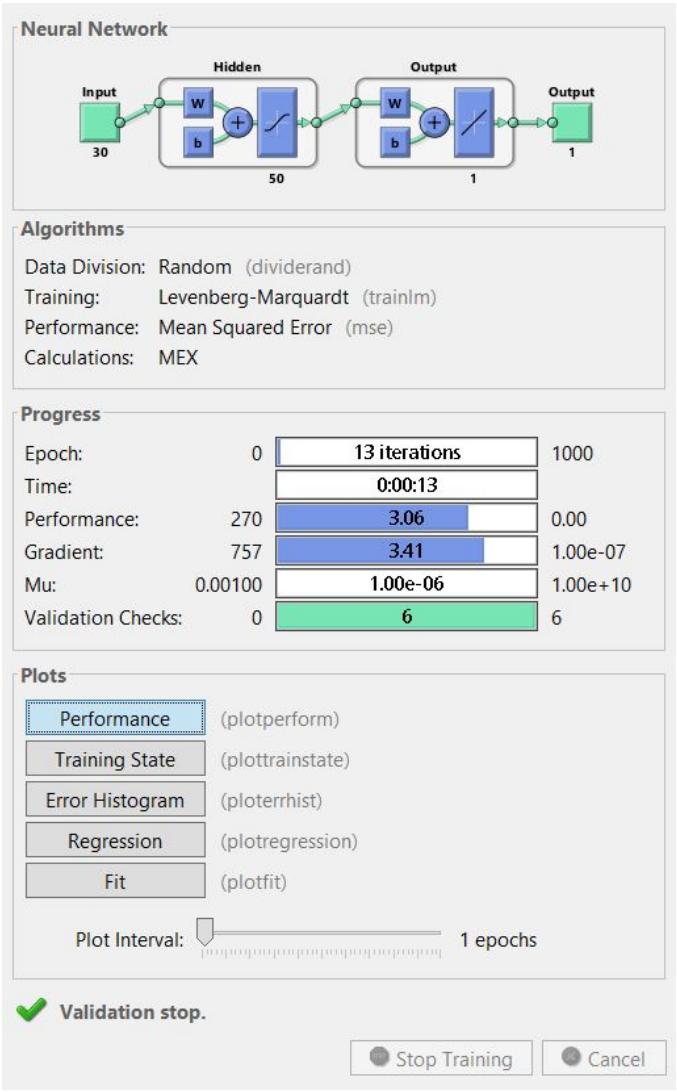


Figure 5.2: Neural Network Training Tool

and, if it does, the path planner will identify the path, among all the possible ones, that minimizes the risk for humans.

5.2 Simulation

The purpose of this part of the work is to establish a relation between the values extracted with image analysis (RGB mean values and edge detection[1]) and the sheltering factor values assigned by inspection. This means in particular that $\mathbf{C}_{tot}$ is the matrix of input data to present to the network, while $\mathbf{m}$ is the vector of target data defining the desired network output.
As previously said, the $\mathbf{C}$ matrix associated to every picture has dimension 160×30, so $\mathbf{C}_{tot}$ containing all the $\mathbf{C}$'s of every image is a 32000×30 matrix. In the same way $\mathbf{m}$ is a vector 32000 elements vector. Therefore the inputs of the neural network are composed by 32000 samples of 30 elements and the targets by 32000 samples of 1 element.
The distribution of the samples, the number of hidden neurons and the best algorithm to be used can not be defined univocally, as explained previously, so, to find the best compromise a trial-and-error approach has been used.
For what concerns the division of the samples the configurations have been chosen in order to have a training set big enough, at least 70% of the total number of samples. At the same time, in order to have a meaningful outcome we have thought to use at least 30 images (15% of the whole data set) for the testing phase, since using less could have led to misleading results. Three configurations per algorithm have been tested and they are listed in the next paragraphs 5.2.1 and 5.2.2.
The number of neurons in the hidden layer have been selected with reference the rules of thumb previously listed, especially the ones expressed by [4] and [3]. Given the size of the input layer equal to 30 and the one of the output layer equal to 1, six different numbers of hidden neurons have been tested: 10, 20, 25, 30, 35, 40, 45, 50 and 60.
Both Levenberg-Marquardt and Bayesian Regularization algorithm have been used in order to find the best fit, while the Scaled Conjugate Gradient method has given such a poor outcome from the beginning that it has been abandoned.

5.2.1 Levenberg-Marquardt algorithm

The different data distributions to apply the algorithm on have been chosen in order to have at least 70% of the data set dedicated to training and 15% to testing.
All the six possible configurations have been tried to find the best one:

- Training 70% - Validation 15% - Testing 15%
- Training 70% - Validation 10% - Testing 20%
- Training 70% - Validation 5% - Testing 25%

[1] See section 3.1.2

- Training 75% - Validation 10% - Testing 15%

- Training 75% - Validation 5% - Testing 20%

- Training 80% - Validation 5% - Testing 15%

The following tables show the best results in terms of regression, obtained using the Levenberg-Marquardt algorithm on different distribution of samples and with different number of neurons. Since regression shows how strict is the connection between the outputs and the inputs of the network, we have used this parameter to have a first evaluation of the network performance. A regression equal to 1 means a close relation, while if it equal to 0 it indicates a random one.

Neurons	**Training**	**Validation**	**Testing**
10	0.671	0.668	0.657
20	0.687	0.658	0.681
25	0.70	0.671	0.681
30	0.721	0.691	0.689
35	0.723	0.683	0.679
40	0.735	0.69	0.678
45	0.74	0.688	0.68
50	0.749	0.689	0.685
60	0.755	0.68	0.671

Table 5.1: Levenberg-Marquardt algorithm - 70/15/15 division

Neurons	**Training**	**Validation**	**Testing**
10	0.658	0.651	0.659
20	0.692	0.67	0.66
25	0.703	0.673	0.68
30	0.733	0.665	0.691
35	0.725	0.672	0.681
40	0.735	0.678	0.67
45	0.741	0.676	0.67
50	0.75	0.687	0.69
60	0.753	0.686	0.681

Table 5.2: Levenberg-Marquardt algorithm - 70/10/20 division

Neurons	Training	Validation	Testing
10	0.675	0.653	0.654
20	0.695	0.664	0.668
25	0.728	0.677	0.674
30	0.707	0.661	0.678
35	0.736	0.678	0.69
40	0.748	0.674	0.683
45	0.734	0.664	0.676
50	0.741	0.665	0.691
60	0.732	0.681	0.678

Table 5.3: Levenberg-Marquardt algorithm - 70/5/25 division

Neurons	Training	Validation	Testing
10	0.671	0.649	0.667
20	0.698	0.665	0.684
25	0.713	0.681	0.688
30	0.725	0.658	0.688
35	0.734	0.677	0.685
40	0.726	0.664	0.68
45	0.736	0.686	0.676
50	0.755	0.711	0.693
60	0.754	0.697	0.686

Table 5.4: Levenberg-Marquardt algorithm - 75/10/15 division

Neurons	Training	Validation	Testing
10	0.675	0.653	0.654
20	0.695	0.664	0.668
25	0.728	0.677	0.674
30	0.707	0.661	0.678
35	0.736	0.678	0.69
40	0.734	0.675	0.686
45	0.747	0.678	0.683
50	0.741	0.665	0.691
60	0.753	0.686	0.681

Table 5.5: Levenberg-Marquardt algorithm - 75/5/20 division

Neurons	Training	Validation	Testing
10	0.657	0.653	0.651
20	0.705	0.681	0.686
25	0.709	0.705	0.687
30	0.713	0.655	0.693
35	0.72	0.697	0.686
40	0.722	0.703	0.684
45	0.715	0.703	0.685
50	0.725	0.685	0.698
60	0.758	0.694	0.675

Table 5.6: Levenberg-Marquardt algorithm - 80/5/15 division

5.2.2 Bayesian Regularization algorithm

The following tables show the best results in terms of regression, obtained using the Bayesian Regularization algorithm on different distribution of samples and with different number of neurons. Once again, since regression shows how strict is the connection between the outputs and the inputs of the network, we have used this parameter to have a first evaluation of the network performance. A regression equal to 1 means a close relation, while if it equal to 0 it indicates a random one.
It is important to notice that with this method the training can continue until an optimal combination of errors and weights is found, so the validation stops are disabled by default. However it is not possible to assign the 0% of the samples to the validation phase and this implies a loss of samples, that are practically not used at all.
The different data distributions to apply the algorithm on have been chosen in order to have at least 70% of the data set dedicated to training and 15% to testing.
All the three possible configurations have been tried to find the best one:

- Training 70% - Validation 5% - Testing 25%

- Training 75% - Validation 5% - Testing 20%

- Training 80% - Validation 5% - Testing 15%

Neurons	Training	Testing
10	0.667	0.669
20	0.706	0.683
25	0.721	0.698
30	0.728	0.69
35	0.733	0.691
40	0.737	0.686
45	0.738	0.682
50	0.752	0.70
60	0.759	0.68

Table 5.7: Bayesian regularization algorithm - 70/5/25 division

Neurons	Training	Testing
10	0.676	0.664
20	0.708	0.681
25	0.711	0.696
30	0.724	0.695
35	0.736	0.698
40	0.733	0.697
45	0.734	0.693
50	0.754	0.702
60	0.756	0.687

Table 5.8: Bayesian regularization algorithm - 75/5/20 division

Neurons	Training	Testing
10	0.674	0.678
20	0.705	0.684
25	0.718	0.69
30	0.724	0.691
35	0.731	0.704
40	0.736	0.697
45	0.745	0.697
50	0.753	0.701
60	0.755	0.685

Table 5.9: Bayesian regularization algorithm - 80/5/15 division

5.2.3 Discussion of the results and choice of the neural network

The percentages shown in the previous tables refer to the regression value, that is a measure of the correlation between outputs and inputs of the neural network. A high percentage indicates a close relationship, while a low percentage a random one. This parameter is the one we have used to have a first evaluation of the network performance. Of course, the real, overall, behaviour of the network will be evaluated once the corresponding function generated by the toolbox is integrated in the map generation algorithm in Chapter 6. In fact the real performance will be evaluated with reference to the real errors between the map generated by the algorithm and the desired one. Again, the regression found in the previous simulations is only used to have a first idea of which one of the neural networks is the most promising for the next steps of the work.
The results we have obtained are generally good, but not excellent as we would have hoped. This is could be due to the difficulty of the neural network to predict the value for those portions of image that show a mixed situation, that are all the ones in which, when assigning the sheltering factor by inspection, we had to make a decision between two distinctively different values. Or, since the image analysis is based mainly on colours, the cause could be the similarity in colour of situations with very different sheltering values like, for example the grey of a street compared with the grey of the rooftop of a building. This lack of difference could have lead to wrong prediction of the sheltering value.
As can be observed, the difference between the results obtained with the two algorithm are generally so small that they can be considered practically equivalent. However the Bayesian regularization algorithm performs slightly better, so we have decided to adopt the ANN that has given a 0.704 regression, that is the one obtained with 35 hidden neurons and a 80/5/15 distribution of samples between training, validation and testing. The regression plots in figure 5.4 display the network outputs with respect to targets for training and test sets. For a perfect fit, the data should fall along the line identified by the equation that links the outputs to the targets. In this case, since offsets and gains are applied to the output to equal the target and yet the regression is not 1, the line is not a 45 degrees line. The closer to 1 is the regression, the closer to 45 degrees is the angle of the line. Figure 5.3 shows the error histogram that can be used to obtain additional verification of network performance: errors are the difference between the targets, that is to say the desired values, and the output of the neural network. Blue bars represent training data and the red bars represent testing data. The distribution is centred in zero and most errors fall in the interval [-5,6].

Since the results have been obtained with a neural network trained on 25600 samples and tested on 4800, we believe that it is reliable and consistent. This will be proved in the next chapter, when the corresponding neural network function generated from it, will be integrated in the map generation algorithm and evaluated based on the final results.
With reference to the results obtained by the different neural networks, it is important to notice that sometimes when the number of neurons exceeds the one of input layers, as in the case of 50 neurons, the difference in percentage between the training and the testing

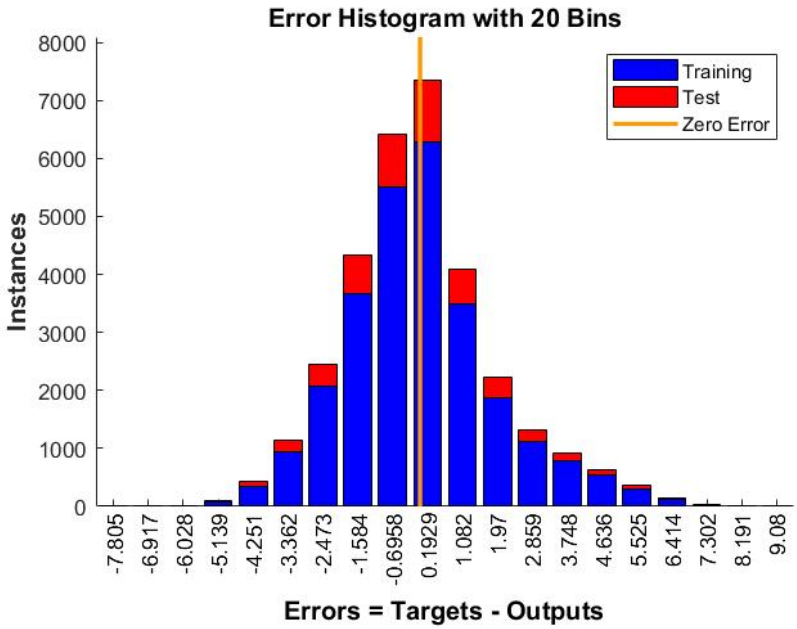


Figure 5.3: Error histogram of the chosen neural network

set tend to grow, in this case spanning from 4 to 8%. This may suggest that the ANN is too complex for the problem and it is gone in overfitting: the network recalls the training samples properly but it is not able to generalize and perform in the same way on new samples outside of it. For this reason we have preferred to avoid choosing one of these networks, picking, instead, a more consistent one.

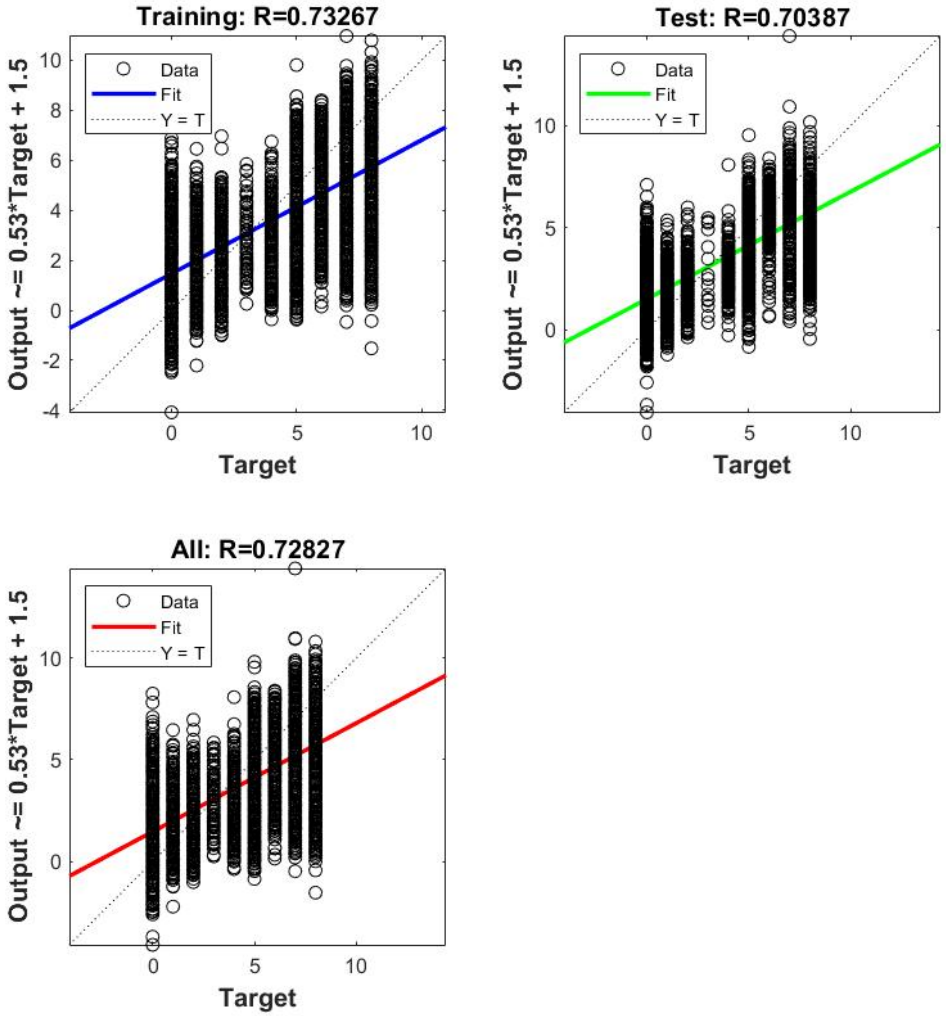


Figure 5.4: Image processing scheme

Chapter 6

Map generation

6.1 Sheltering factor extaction

One of the big advantages of MATLAB neural network toolbox is that it allows to automatically generate the ANN script and function. The script recreates the network at command-line level: every parameter can be controlled and tuned, from the percentages of the data set distribution, to the algorithm, to the number of hidden neurons. Running the script calls the training tool (Figure 5.2), starts the training process and allows to check the performance and results, exactly like what can be done through the GUI.
The neural network function is even more interesting: when called, it takes as input a matrix, in this case the **C** matrix coming from image analysis, of any picture we could possibly want to submit and it analyses it according to the function fitting it has been trained on with the original data. At first it applies to the input a set of offsets and gains, then the result is passed through the inner layers where it is given weights and bias by the neurons. Finally the output is computed and it should have an high correlation with the inputs given. In some way at this point the ANN can be seen as a sort of black box implementing this fitting function between inputs and outputs. It is yet important, once again, to underline that a neural network function developed in this way elaborates data with reference to the model and the rule it was built on: at this point it is used for evaluation of data, but its generalization can not be improved any further. In our case however this is not a problem, since the ANN has been trained on a large set of images, all representing an urban scenario, so its generalization ability, as will be shown in the next pages, is satisfying.
Once extracted from the neural network training tool, the ANN function can be integrated and used in a MATLAB script like any other function.
In particular an ANN has been used for the computation of the sheltering factor of an area starting from an aerial image, so we are interested in testing its performance. In order to do that we can write a program that, given a picture of an urban scenario, elaborates and analyses it and exploits the function derived from the neural network to assign a

sheltering factor to any of its zones.
The first step to make is to choose a neural network that has given good results in terms of regression and that, therefore, would establish the stricter correlation possible between the input, the aerial image, and the output, the sheltering map. The ANN that has given the best results in these terms is the one developed with 35 hidden neurons, the Bayesian Regularization algorithm and a 80/5/15 percent division of the data set among training, validation and testing. This network proved a regression of 70.4%, as shown in Table 5.9. Of course an higher result could have been obtained lowering the number of samples to perform the testing phase on. As a matter of fact, at every retraining of the network, the distribution of samples changes randomly, as well as the weights and bias of the neurons, so testing the ANN on a low number of pictures could probably, sooner or later, lead to very high regression. However this result is deceiving, since it is not general and could show poor outcome once new pictures, outside the data set are presented. Instead we have decided to test the neural network on at least 30 images (4800 sub-parts) to ensure an higher generalization ability, rather than an high regression on a small basis, yet meaningless.
The MATLAB script we have elaborated to test the ANN and obtain the sheltering map loads an aerial image A and performs the same image processing described in Chapter 3.1.2: the picture is divided in 160 sub-parts, each one covering an area around $25m^2$, for any of these RGB mean values and edge values are computed with a pixel-level analysis. To have an even evaluation of every sub-part, the external layer of the picture is doubled to expand the image itself. Finally the $\mathbf{C}$ matrix is created, containing all the RGB mean values for each sub-part and for its neighbourhood and the values coming from the edge analysis for each colour, in a specific order. Given a generic sub-part (x,y) of the original image, the corresponding row of the $\mathbf{C}$ matrix has the configuration seen in Equation 3.3. This matrix, once again, is formed by 160 samples of 30 elements each, to be given as input to the ANN to find as output 160 elements that are the sheltering values. Therefore the neural network function is applied to $\mathbf{C}$, it applies its offsets, weights and bias and it finds a resulting vector $\mathbf{s}$. This basically represents all the sheltering factor values associated to the sub-parts of the original picture A and extracted from image analysis. At this point the $\mathbf{s}$ vector is can finally be compared to the $\mathbf{m}$ vector containing all the target values of sheltering factor previously assigned by inspection. The difference between the two is the error vector and it is particularly important to have a quantitative representation of the performances of the neural network and of the method in general. The last step is reshaping vector $\mathbf{s}$ in a 10×16 matrix, called $\mathbf{S}$: by doing this, the algorithm basically reconstructs a map that overlaps the original image A perfectly. In fact they are both matrices, one of values, the other of sub-parts of A, so that given a certain portion of the picture, the corresponding cell of $\mathbf{S}$ contains the extracted sheltering factor value. Therefore the $\mathbf{S}$ matrix represents exactly the sheltering map that was the objective of the whole work. The scheme in Figure 6.1 resumes graphically the entire process.

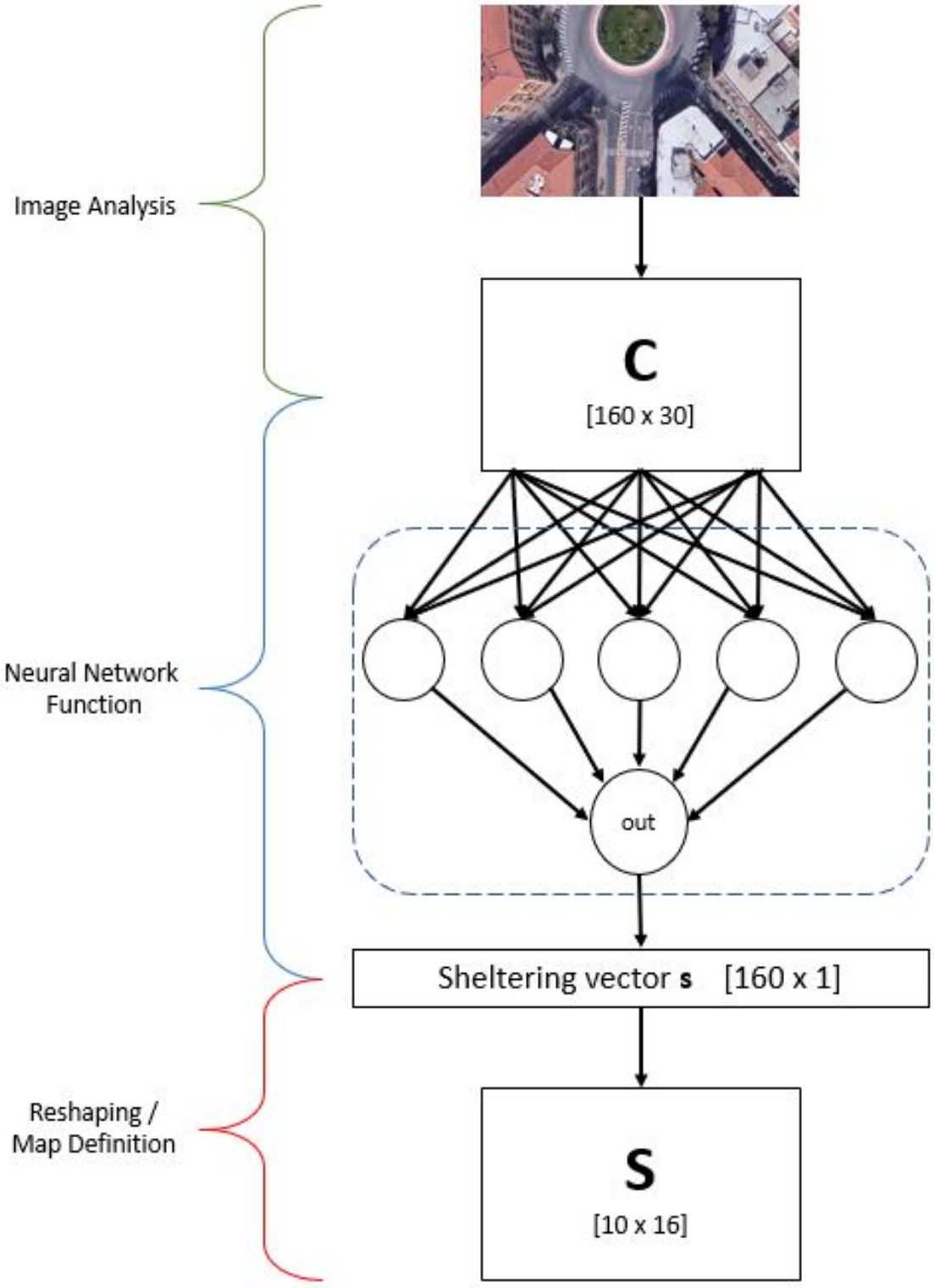


Figure 6.1: Graphic representation of the sheltering extraction and map generation

6.2 A second approach: adding a filter

A neural network like the one we have exploited basically explores the sub-parts of the original image A, observing their mean colours and the possible presence of edges and, as a consequence of this, it extracts the corresponding estimated sheltering factor. This is possible because the network was subject to *supervised training*, that is one of the most interesting techniques of machine learning. In fact, it consists in defining a function that maps an input to an output based on input-output pairs that are called indeed *training examples*. These are composed by an input object, normally a vector, and a desired output value, also known as *supervisory signal*, that are used to train the network to deduce labels from the example and produce a function that can correctly determine those class labels for completely new outputs. This means that the network is able to recognize common features and make a connection between similar characteristics and a certain desired output. Obviously in order to do that, the ANN has to be given an high enough number of examples to be able to identify a pattern.
Supervised training mimics human concept learning, so the definition that was given to the latter, by the american psychologist Jerome Bruner in 1967, describes it perfectly as well:

> The search for and listing of attributes that can be used to distinguish exemplars from non exemplars of various categories

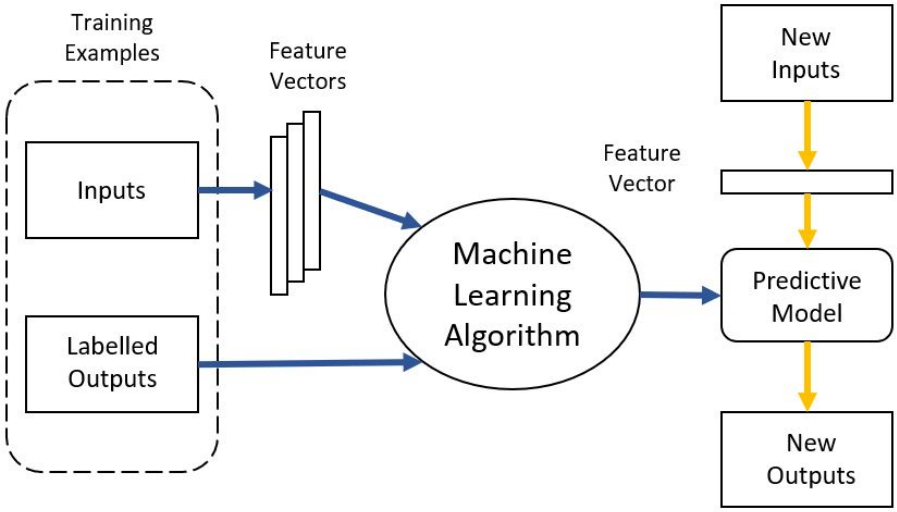


Figure 6.2: Supervised Training working principle

In the case of the training examples we have used, this pattern recognition ability allows

the network to establish a relation between the colour of a certain sub-part of the original image and a desired output among the ones listed in the sheltering factor table 2.3. For example the ANN has learned the association between red roofs and the category of residential buildings, or between green jagged areas and the one of trees.
However the situation is seldom uniquely clear and usually sub-parts present mixed case, that, by inspection we have decided to assign to one or the other level of the sheltering factor scale, performing what can be seen as a kind of discretization. The ANN instead analyses these mixed situations with a continuous scale, interpolating between different levels. In some cases this could result in a more precise evaluation, but on the other hand a continuous representation is easier to be deceived by casual or unpredictable change of colour, like the one of a car parked in a completely free area, for example. Moreover it could not consider possible conservative choices that have been made classifying the image by inspection, leading to a dangerous lower definition of the sheltering factor.
We have decided to perform, in parallel with the original map generation, a filtered version, where the output of the neural network function is discretized to the nearest value of the sheltering factor scale. This makes sense also because the phase following map generation, that is path planning, must receive as input values that can be recognised and fitted in a define framework. The filter is a simple double for loop that explores all the cells of the sheltering matrix $\mathbf{S}$, turning the values into the nearest one of Table 2.3.
In the next section some simulations are performed and both the normal and the filtered sheltering maps are shown and compared.

6.3 Testing the algorithm

To observe the behaviour of the algorithm described so far and to perform map generation, a set of satellite pictures has been chosen and put under test. All of them are taken in an urban environment, since it is the scenario in which possibly the majority of UAV missions would and will take place, nowadays or in future smart cities, as explained in Chapter 1. Moreover cities have an high population density and, as a consequence, they present an high level of risk that needs to be mitigated for the UAVs to be compliant with the standards and be given the permission to fly. Hence defining the proper sheltering map in this context is crucial.
The following images have been chosen to represent several situations with different sheltering factors to have a complete observation of the behaviour of the algorithm and to make an exhaustive evaluation of its performance.
Once again, the neural network function we have integrated in the algorithm is the one produced by the ANN with 35 hidden neurons, the Bayesian Regularization algorithm and a 80/5/15 percent division of the data set among training, validation and testing (see Table 5.9). This is the network that proved the highest regression (0.704), that is to say the strictest correlation between outputs and target.
In order to evaluate the obtained results, for each image the sheltering map associated

by inspection is presented at first, followed by the ones obtained through the application of the algorithm described so far, both without and with filter. The errors will be computed for both the approaches as a comparison between each of them and the desired sheltering map. We are showing an histogram representing the error distribution and we are computing the average (or mean) error present in the specific case. However error is, by definition, varying with positive and negative increments around a central value and therefore the mean error tends inevitably to that value, that, in the best case, is equal to zero. For example it is possible to imagine a distribution of data with no error but a positive one +50 and a negative one -50: the average error computed on the whole distribution is equal to zero, that should imply that no error is present, that is clearly false. Therefore positive and negative cancellation make the mean error not helpful to evaluate the overall performance of the algorithm. To do that, we have to consider instead the error without its sign, since, in this phase, we are not interested in understanding if it is made due to an underestimation or to an overestimation, but only in how the algorithm behaves on a given picture. Therefore another value we are taking into consideration in the analysis of the results is the Mean Absolute Error (MAE), defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x| \tag{6.1}$$

where:
x_i, in this case, is the sheltering factor associated to the i-th cell, computed by the algorithm
x, in this case, is the value of sheltering factor assigned by inspection to the cell.

To have an immediate representation of the behaviour and performance of the algorithm, the Mean Absolute Error could also be expressed as a percentage. Given the 10 levels of sheltering factor (Table 2.3),

$$MAE_{\%} = \frac{MAE}{10} \times 100 \tag{6.2}$$

where $MAE_{\%}$ indicates the percentage of the error in the whole map with respect to the desired value assigned by inspection.
The following simulations are divided in best and worst cases to highlight the strengths and the limits of the algorithm. A discussion of the overall results is presented at the end in section 6.3.3.

6.3.1 Best cases

Test One

The first aerial picture to be put under test represents a typical situation in an urban scenario, that are some narrow streets in between the buildings, with a particularly recognizable shape; hence it has two zones with very different sheltering ability. To evaluate the map generation algorithm we have to make a comparison between the output of the latter and the desired one. To do that we have assigned by inspection the sheltering factor following the classification that we have established in Table 2.3. The resulting **M** matrix can be plotted in a colour scale for an easier and clearer comprehension and it is represented in Figure 6.4.

The output of the neural network function applied on the original image 6.3 is, as said,

Figure 6.3: Aerial Picture 1

a vector **s**, that can be re-arranged in a 10×16 grid to obtain the **S** matrix, dimensionally equivalent to **M**. This last one can be plotted in the same colour scale as the previous to have an immediate comparison of the result. The map obtained with the map generation algorithm is shown in figure 6.5, while figure 6.6 represents the sheltering map that outcomes from the the application of the filter on the previous. At a first sight, the result is quite good, since the algorithm is able to identify the two different zone, the profile of the buildings and the shape of the narrow streets. However it assigns an higher sheltering factor to the sub-parts corresponding to the street, but in this case, since the value for the buildings is much higher, it would probably not be a problem for navigation. It is yet important to notice that, in general, overestimating the sheltering ability of an area can

Figure 6.4: Sheltering map assigned by inspection of Figure 6.3

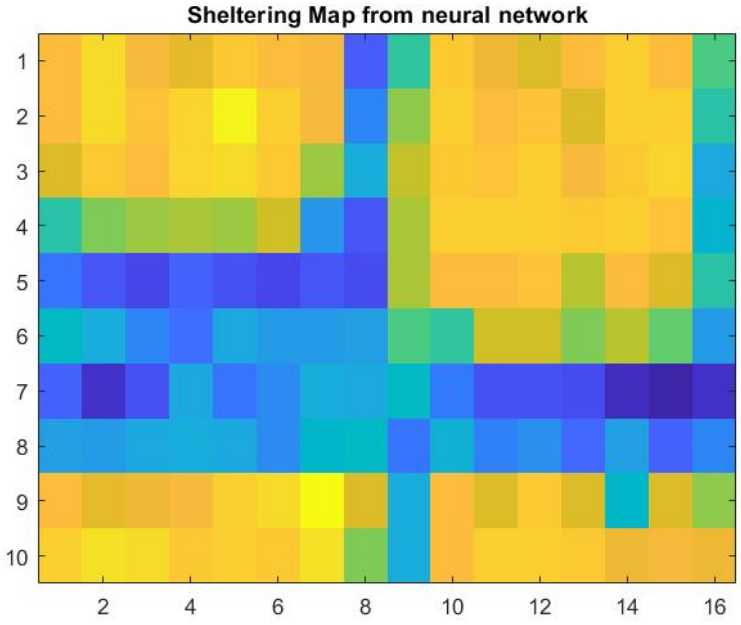


Figure 6.5: Sheltering map obtained with neural network function applied to Figure 6.3

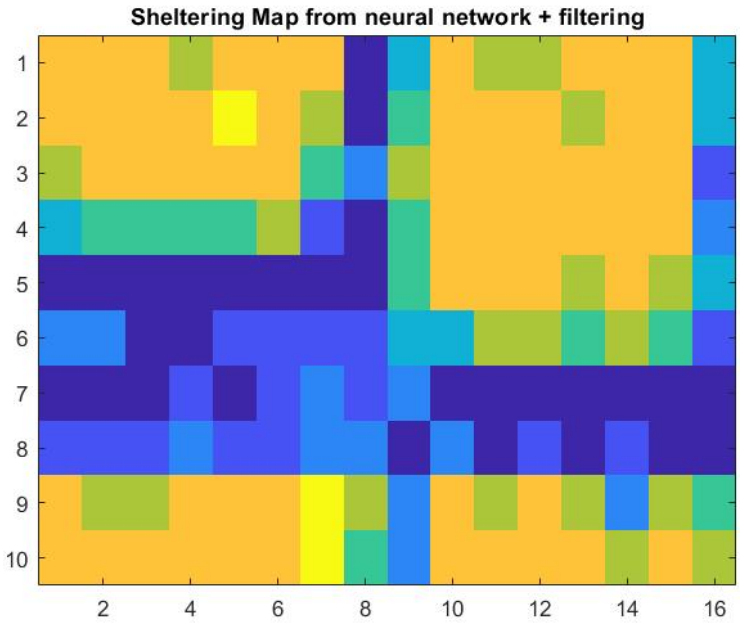


Figure 6.6: Sheltering map obtained with neural network function and filter applied to Figure 6.3

lead to a non optimal trajectory in terms of safety.
Moreover the filtered map reduces a lot this effect, lowering the sheltering factor of the aforementioned sub-parts and it makes the values, and hence the colours, more homogeneous, as reasonable, since the Path Planner expects discrete values of sheltering factor.
Leaving aside the graphical inspection, we must focus on the more precise error analysis. Once again, the error is the difference between the values of maps 6.5 and 6.6 and the one of the desired sheltering map 6.3. Histograms in figures 6.7 and6.8 show the distribution of errors in the 160 sub-parts that constitute the map.

As we expect the histogram of the filtered map have discrete values, corresponding to the levels of sheltering factor given in 2.3, while 6.7 shows a continuous distribution. We have computed the mean error and the variance and they are reported in Table 6.1, together with the mean absolute error. As explained in the previous section, the MAE is able to quantify not only the overall number of errors, but also their entity and therefore we use it to evaluate the correctness of the map generation algorithm.
As clear from table6.1, MAE is very low, under 0.801, and the addition of the filter has a positive effect, reducing it further by 0.18.
Finally, figure 6.9 shows the plots of error vector both filtered and unfiltered: the filter applied to the error vector is able to reduce some minor noise and to discretize the sheltering factor values on the expected levels.

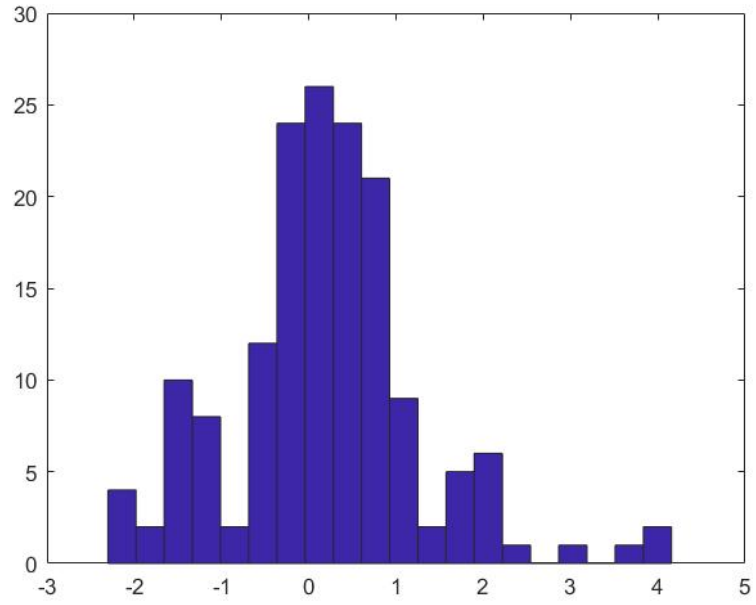


Figure 6.7: Distribution of the errors between sheltering maps 6.5 and 6.4

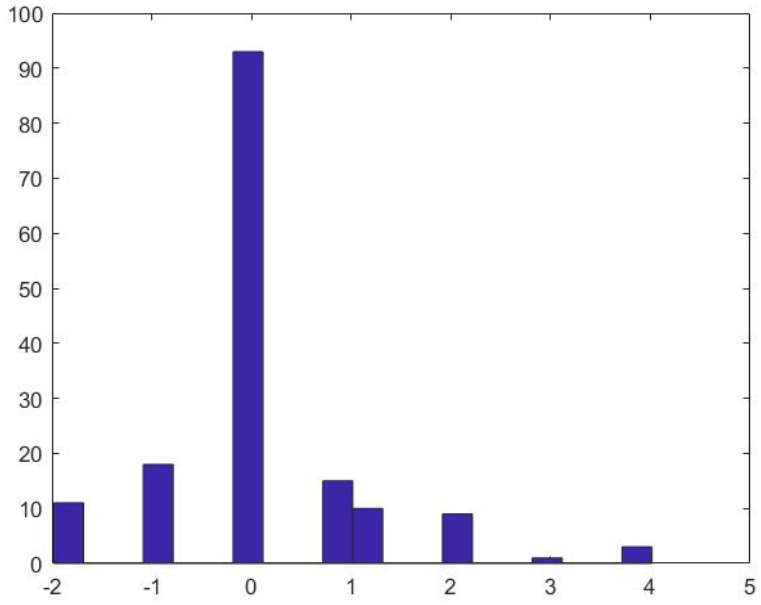


Figure 6.8: Distribution of the errors between sheltering maps 6.6 and 6.4

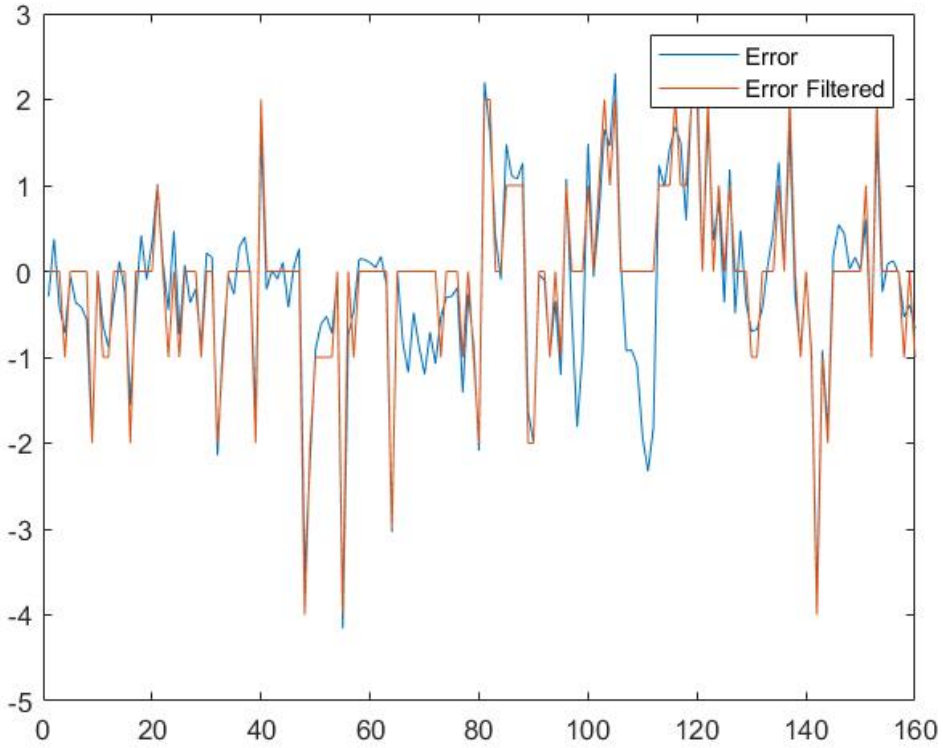


Figure 6.9: Effect of the filter in reducing the noise on the original error signal

	Neural Network	Neural Network + Filter
Sheltering Factor Mean Error	-0.1987	-0.1050
Variance	1.2075	1.1221
Mean Absolute Error	0.801	0.621

Table 6.1: Results of Test One

Test Two

The first picture to be put under test represents a typical situation in an urban scenario, that is a square surrounded by buildings, so it has two zones with very different sheltering ability. To evaluate the map generation algorithm we have to make a comparison between the output of the latter and the desired one. To do that we have assigned by inspection the sheltering factor following the classification that we have established in Table 2.3. The resulting **M** matrix can be plotted in a colour scale for an easier and clearer comprehension and it is represented in Figure 6.11.

The output of the neural network function applied on the original image 6.10 is, as said,

Figure 6.10: Aerial Picture 2

a vector **s**, that can be re-arranged in a 10×16 grid to obtain the **S** matrix, dimensionally equivalent to **M**. This last one can be plotted in the same colour scale as the previous to have an immediate comparison of the result. The map obtained with the map generation algorithm is shown in figure 6.12, while figure 6.13 represents the sheltering map that outcomes from the the application of the filter on the previous. At a first sight, the result is quite good, since the algorithm is able to identify the two different zone, the profile of the buildings and it also detects the presence of the two narrow streets in the lowest part. However, here as well, it assigns an higher sheltering factor to the sub-parts corresponding to the square. In this case, since the value for the buildings is much higher, it would probably not be a problem for navigation.

The filtered map apparently reduces this effect, lowering the sheltering factor of the aforementioned sub-parts. Moreover it makes the values, and hence the colours, more homogeneous, that is reasonable, since the Path Planner expects discrete values of sheltering

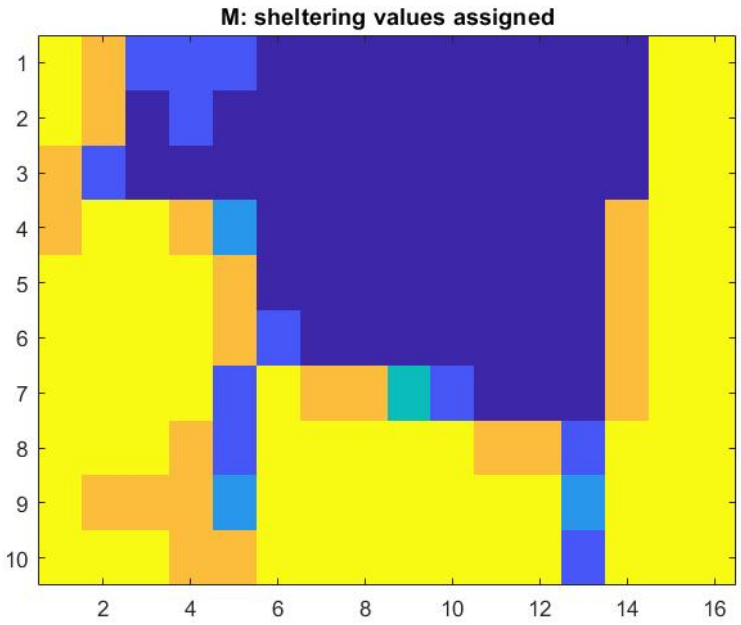


Figure 6.11: Sheltering map assigned by inspection of Figure 6.10

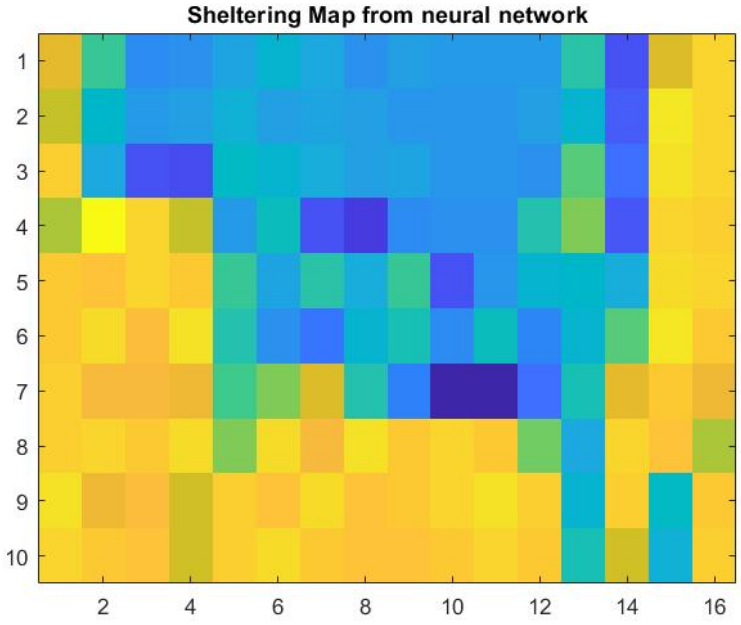


Figure 6.12: Sheltering map obtained with neural network function applied to Figure 6.10

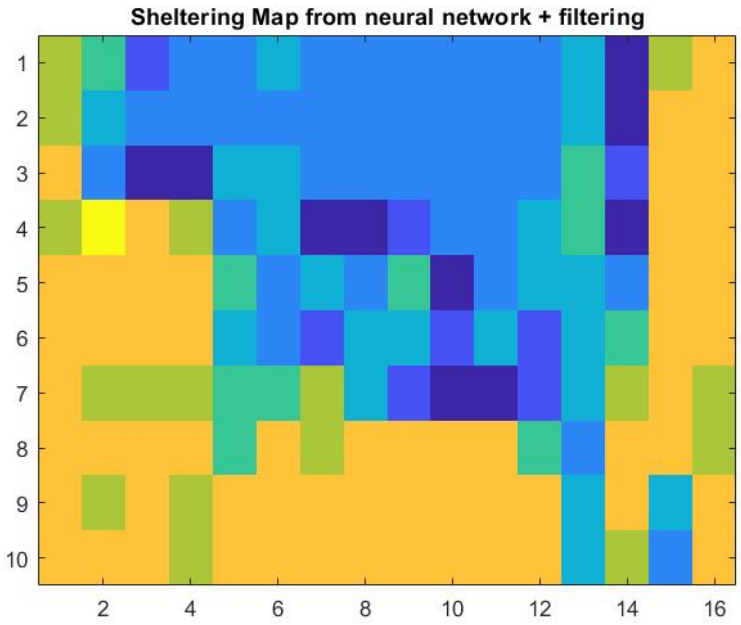


Figure 6.13: Sheltering map obtained with neural network function and filter applied to Figure 6.10

factor.
Leaving aside the graphical inspection, we must focus on the more precise error analysis. Once again, the error is the difference between the values of maps 6.12 and 6.13 and the one of the desired sheltering map 6.10. Histograms in figures 6.14 and6.15 show the distribution of errors in the 160 sub-parts that constitute the map.

As we expect the histogram of the filtered map have discrete values, corresponding to the levels of sheltering factor given in 2.3, while 6.14 shows a continuous distribution. We have computed the mean error and the variance and they are reported in Table 6.2, together with the mean absolute error. As explained in the previous section, the MAE is able to quantify not only the overall number of errors, but also their entity and therefore we use it to evaluate the correctness of the map generation algorithm.
As clear from table6.2, MAE is very low and the addition of the filter has a positive effect, reducing it by a further 0.076.
Finally, figure 6.16 shows the plots of error vector both filtered and unfiltered: the filter applied to the error vector is able to reduce some minor noise and to discretize the sheltering factor values on the expected levels.

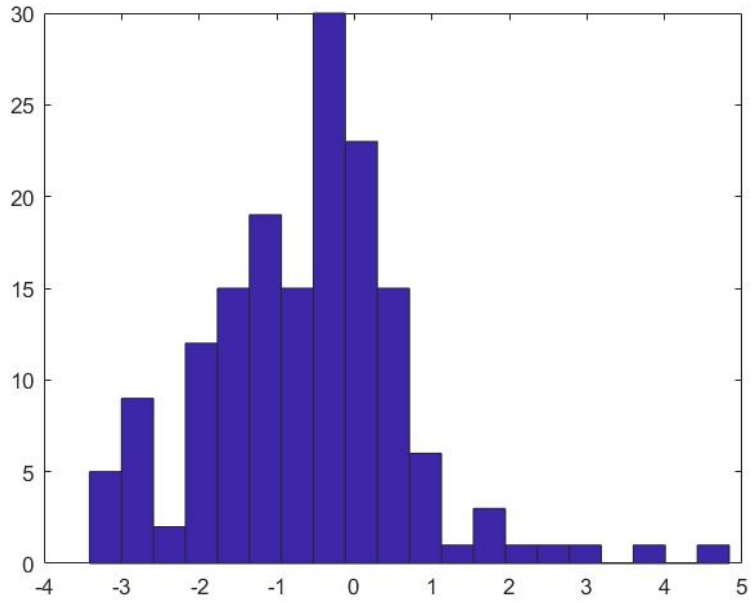


Figure 6.14: Distribution of the errors between sheltering maps 6.12 and 6.11

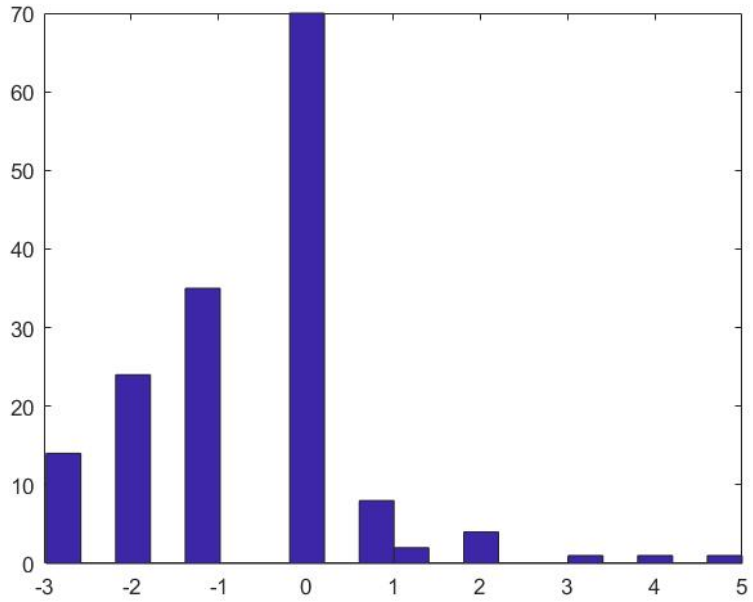


Figure 6.15: Distribution of the errors between sheltering maps 6.13 and 6.11

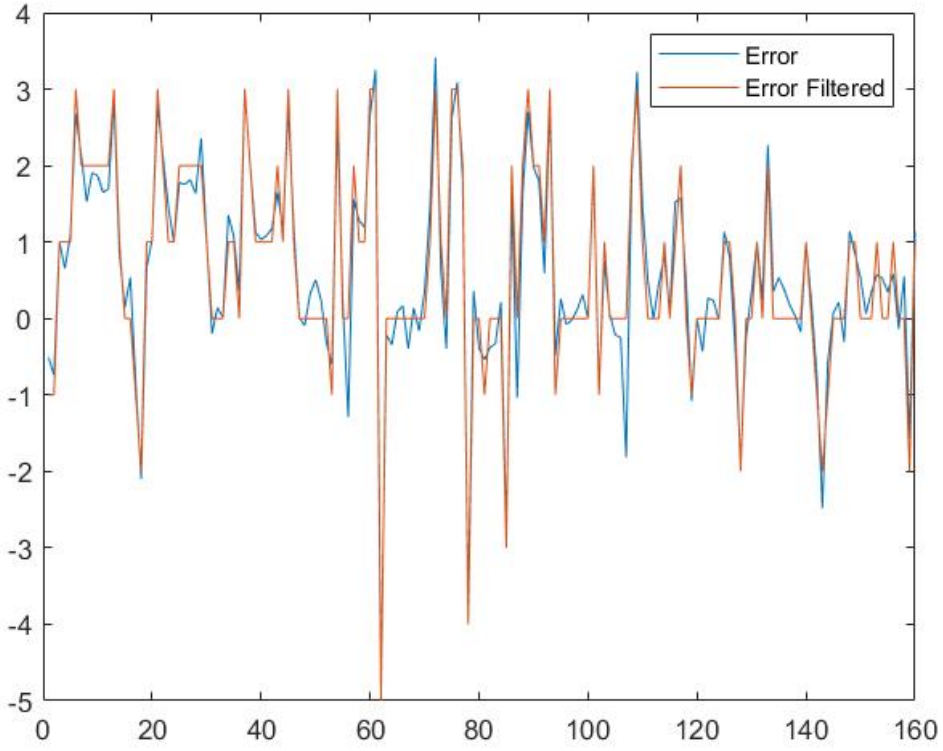


Figure 6.16: Effect of the filter in reducing the noise on the original error signal

	Neural Network	Neural Network + Filter
Sheltering Factor Mean Error	0.5973	0.5980
Variance	1.6475	1.7142
Mean Absolute Error	1.05	0.974

Table 6.2: Results of Test Two

Test Three

The following aerial picture represents another typical situation in an urban scenario: some buildings, probably residential, with a garden in between and some trees. Despite the presence of the trees, they are quite sparse and therefore thy could not be identified as a defined zone. Yet it is interesting to see if the algorithm perceives the trees, but we could say that the area has two zones with very different sheltering ability: the buildings and the garden. To evaluate the map generation algorithm we have to make a comparison between the output of the latter and the desired one. To do that we have assigned by inspection the sheltering factor following the classification that we have established in Table 2.3. The resulting **M** matrix can be plotted in a colour scale for an easier and clearer comprehension and it is represented in Figure 6.18.

The output of the neural network function applied on the original image 6.17 is, as said,

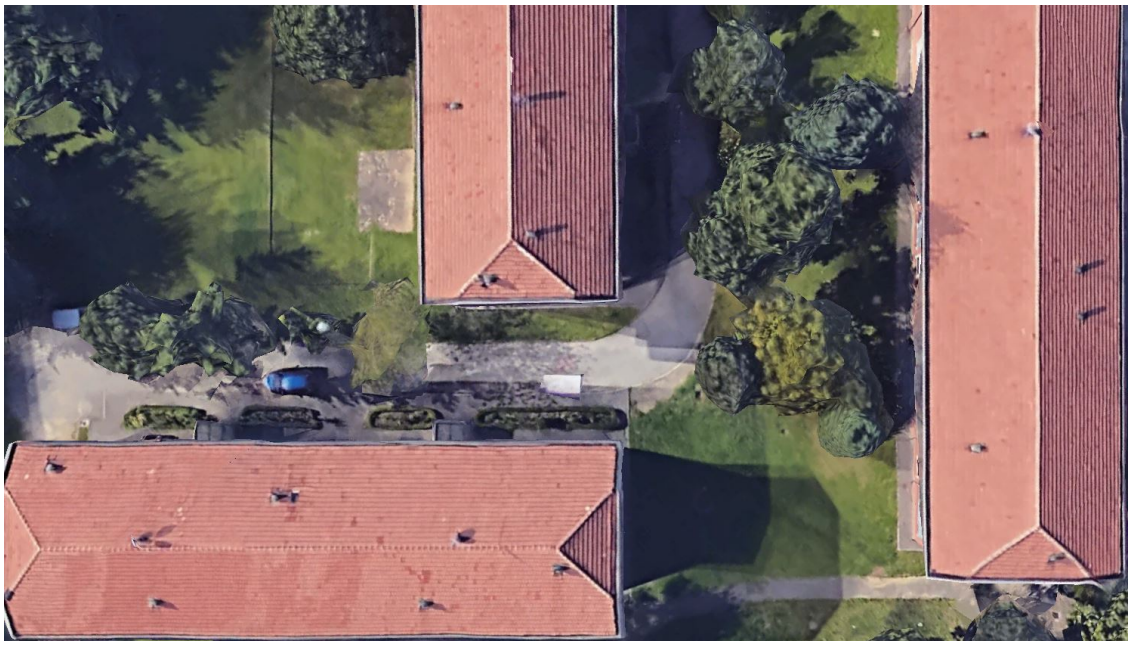


Figure 6.17: Aerial Picture 3

a vector **s**, that can be re-arranged in a 10×16 grid to obtain the **S** matrix, dimensionally equivalent to **M**. This last one can be plotted in the same colour scale as the previous to have an immediate comparison of the result. The map obtained with the map generation algorithm is shown in figure 6.19, while figure 6.20 represents the sheltering map that outcomes from the the application of the filter on the previous. At a first sight, the result is very good, since the algorithm is able to identify the two different zone, the profile of the buildings and even the presence of the trees. Nevertheless, in this case as well, it assigns an higher sheltering factor to the sub-parts corresponding to the garden, but the error seems so slight with respect to the other values in the map, that it probably would not impact in any way the path planning phase. Moreover the filtered map reduces this effect,

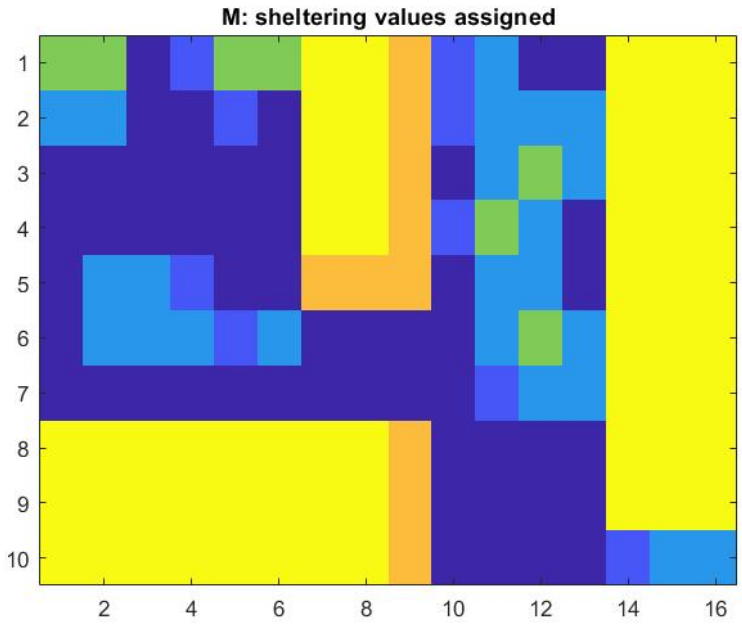


Figure 6.18: Sheltering map assigned by inspection of Figure 6.17

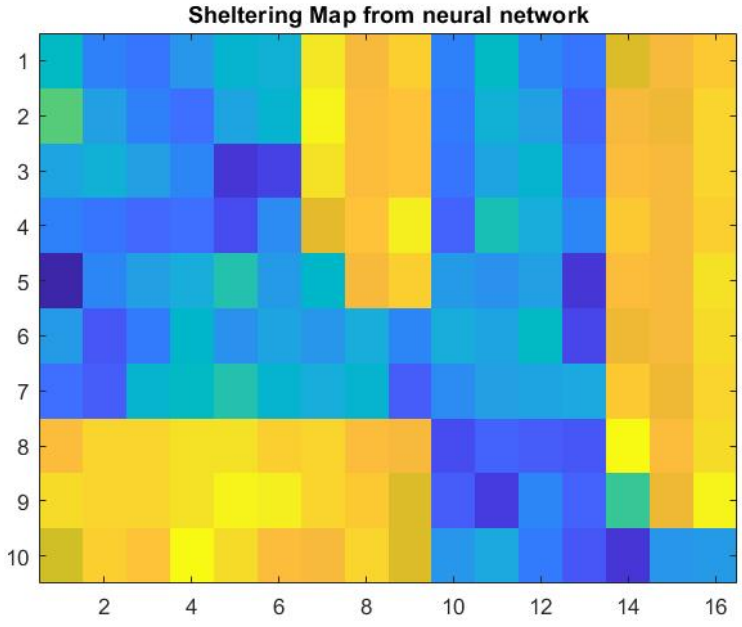


Figure 6.19: Sheltering map obtained with neural network function applied to Figure 6.17

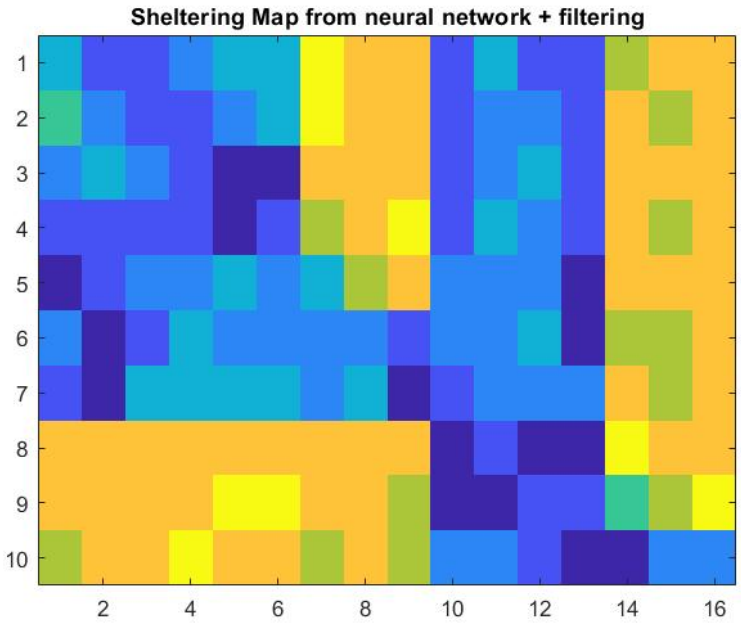


Figure 6.20: Sheltering map obtained with neural network function and filter applied to Figure 6.17

lowering the sheltering factor of some of the aforementioned sub-parts and it makes the values, and hence the colours, more homogeneous, as reasonable, since the Path Planner expects discrete values of sheltering factor.
Leaving aside the graphical inspection, we must focus on the more precise error analysis. Once again, the error is the difference between the values of maps 6.19 and 6.20 and the one of the desired sheltering map 6.17. Histograms in figures 6.21 and6.22 show the distribution of errors in the 160 sub-parts that constitute the map.

As we expect the histogram of the filtered map have discrete values, corresponding to the levels of sheltering factor given in 2.3, while 6.21 shows a continuous distribution. We have computed the mean error and the variance and they are reported in Table 6.3, together with the mean absolute error. As explained in the previous section, the MAE is able to quantify not only the overall number of errors, but also their entity and therefore we use it to evaluate the correctness of the map generation algorithm.
As clear from table6.3, MAE is low, under 0.9, and the addition of the filter has a positive effect, reducing it further by 0.095.
Finally, figure 6.23 shows the plots of error vector both filtered and unfiltered: the filter applied to the error vector is able to reduce some minor noise and to discretize the sheltering factor values on the expected levels.

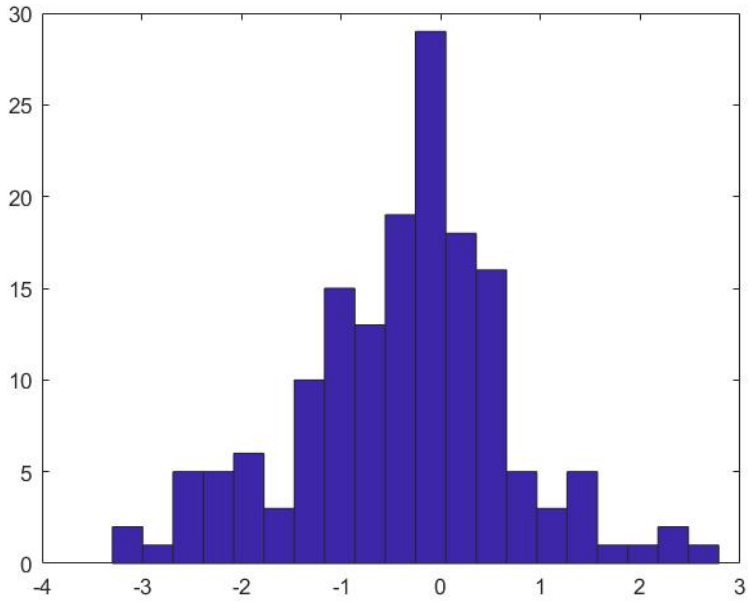


Figure 6.21: Distribution of the errors between sheltering maps 6.19 and 6.18

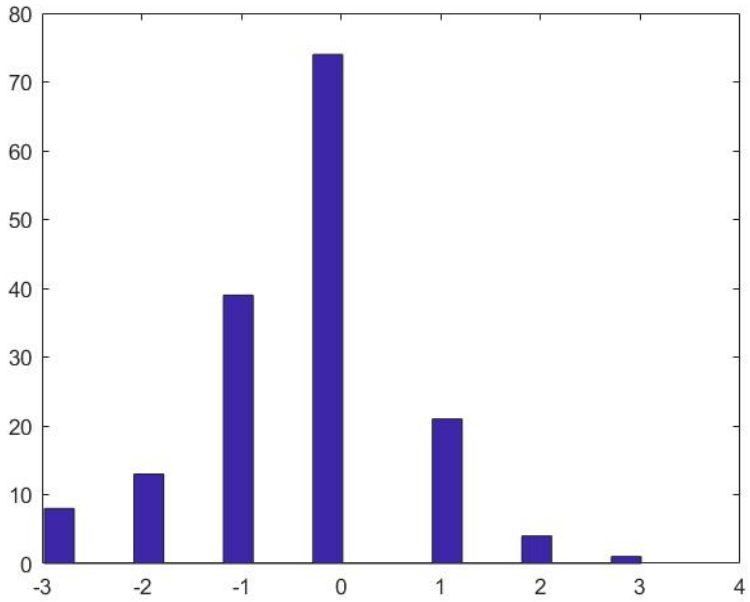


Figure 6.22: Distribution of the errors between sheltering maps 6.20 and 6.18

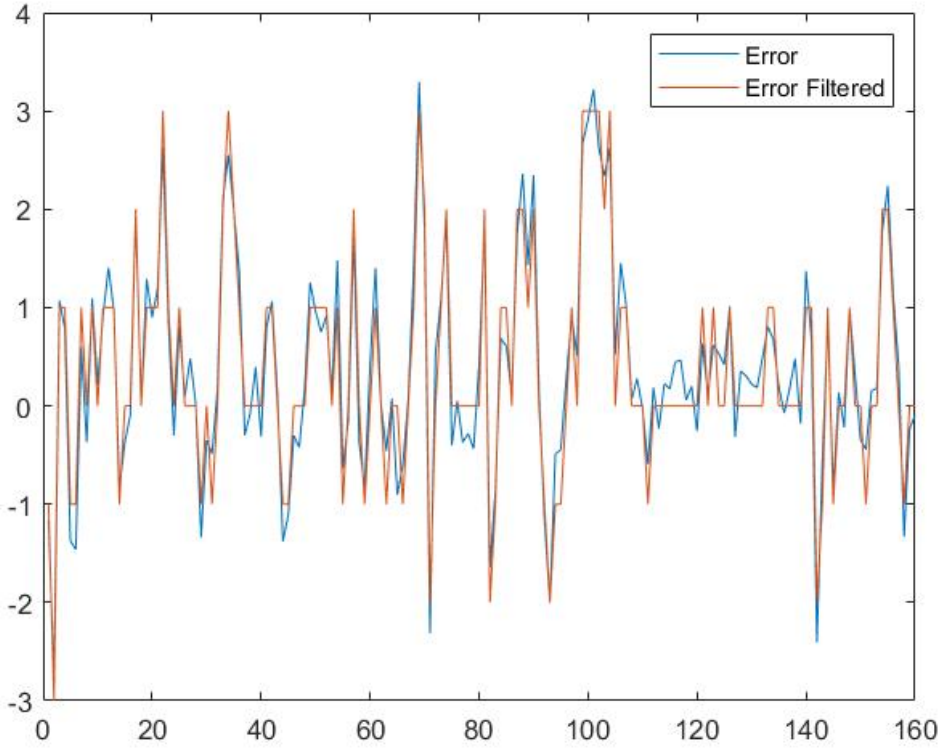


Figure 6.23: Effect of the filter in reducing the noise on the original error signal

	Neural Network	Neural Network + Filter
Sheltering Factor Mean Error	0.3886	0.3592
Variance	1.1753	1.1724
Mean Absolute Error	0.856	0.761

Table 6.3: Results of Test Three

Test Four

The following aerial picture represents another typical situation in an urban scenario: an angular building with a garden and some trees, so the area has three zones with very different sheltering ability. To evaluate the map generation algorithm we have to make a comparison between the output of the latter and the desired one. To do that we have assigned by inspection the sheltering factor following the classification that we have established in Table 2.3. The resulting **M** matrix can be plotted in a colour scale for an easier and clearer comprehension and it is represented in Figure 6.25.

The output of the neural network function applied on the original image 6.24 is, as said,

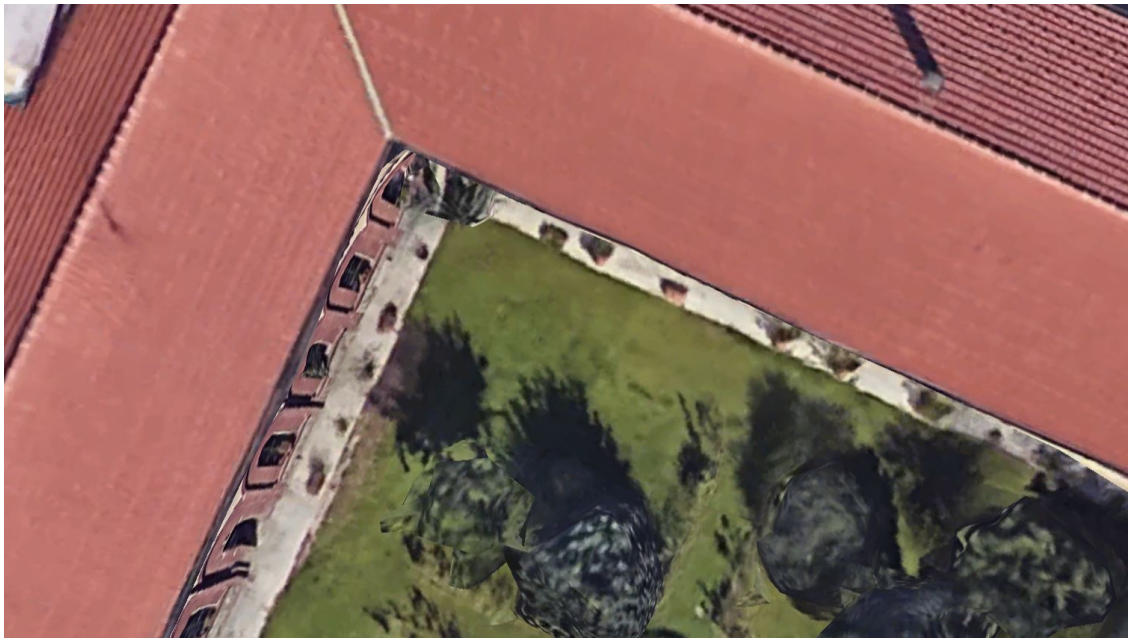


Figure 6.24: Aerial Picture 4

a vector **s**, that can be re-arranged in a 10×16 grid to obtain the **S** matrix, dimensionally equivalent to **M**. This last one can be plotted in the same colour scale as the previous to have an immediate comparison of the result. The map obtained with the map generation algorithm is shown in figure 6.26, while figure 6.27 represents the sheltering map that outcomes from the the application of the filter on the previous.

At a first sight, the result is very good, since the algorithm is able to identify the profile of the building, the garden and even, slightly, the presence of the trees. Nevertheless, in this case as well, it assigns an higher sheltering factor to the sub-parts corresponding to the garden, but the error seems so small with respect to the other values in the map, that it probably would not impact in any way the path planning phase. Moreover the filtered map reduces this effect, lowering the sheltering factor of some of the aforementioned sub-parts and it makes the values, and hence the colours, more homogeneous, as reasonable, since the Path Planner expects discrete values of sheltering factor.

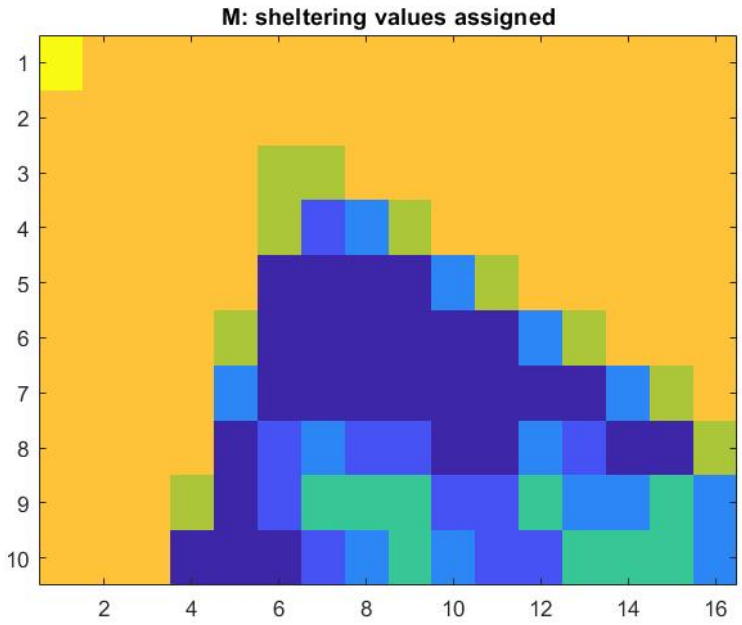


Figure 6.25: Sheltering map assigned by inspection of Figure 6.24

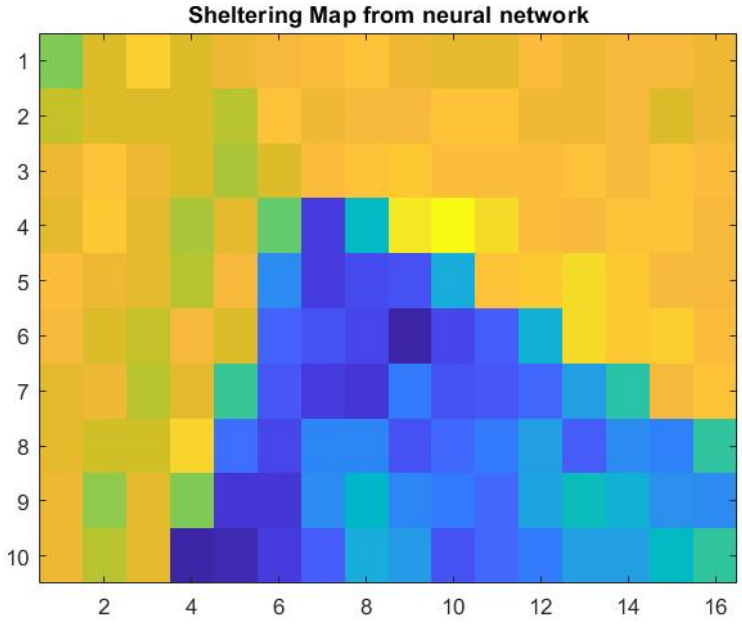


Figure 6.26: Sheltering map obtained with neural network function applied to Figure 6.24

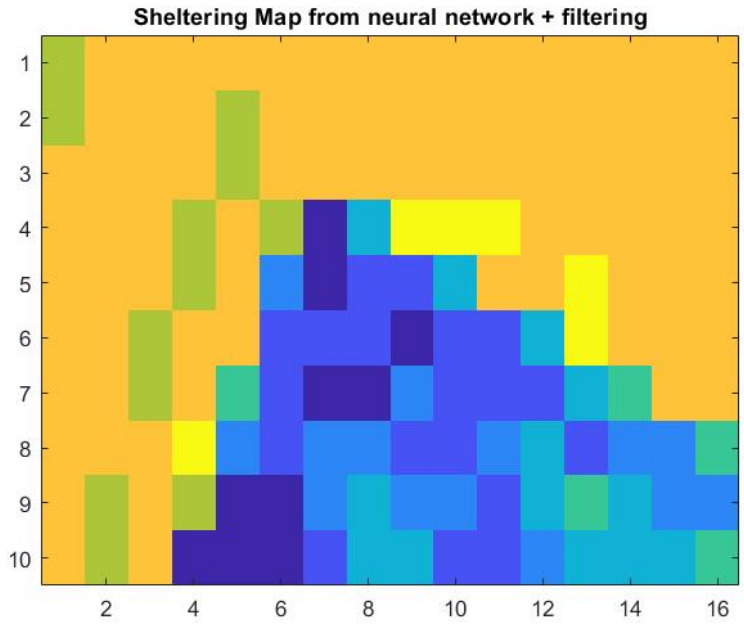


Figure 6.27: Sheltering map obtained with neural network function and filter applied to Figure 6.24

Leaving aside the graphical inspection, we must focus on the more precise error analysis. Once again, the error is the difference between the values of maps 6.26 and 6.27 and the one of the desired sheltering map 6.24. Histograms in figures 6.28 and6.29 show the distribution of errors in the 160 sub-parts that constitute the map.

As we expect the histogram of the filtered map have discrete values, corresponding to the levels of sheltering factor given in 2.3, while 6.28 shows a continuous distribution. We have computed the mean error and the variance and they are reported in Table 6.4, together with the mean absolute error. As explained in the previous section, the MAE is able to quantify not only the overall number of errors, but also their entity and therefore we use it to evaluate the correctness of the map generation algorithm.

As clear from table6.4, MAE is very low, in particular it is the lowest we have found, and the addition of the filter has a positive effect, reducing it even further by 0.081.

Finally, figure 6.30 shows the plots of error vector both filtered and unfiltered: the filter applied to the error vector is able to reduce some minor noise and to discretize the sheltering factor values on the expected levels.

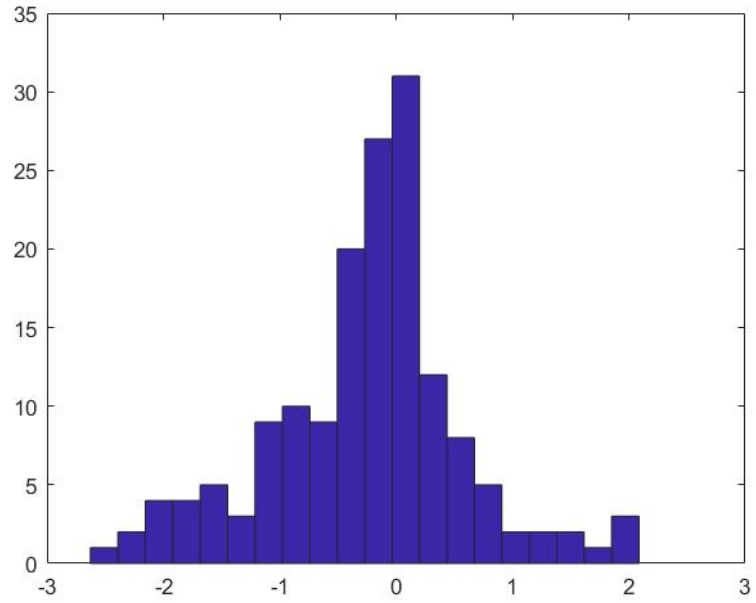


Figure 6.28: Distribution of the errors between sheltering maps 6.26 and 6.25

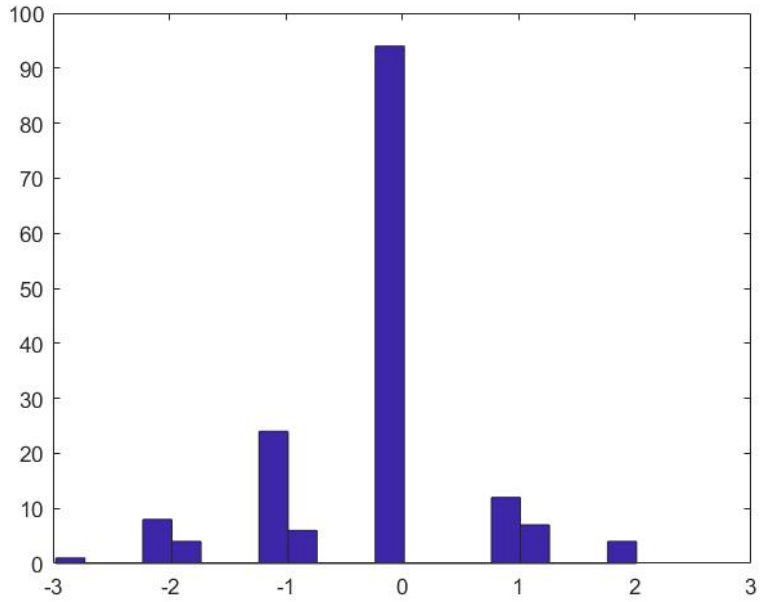


Figure 6.29: Distribution of the errors between sheltering maps 6.27 and 6.25

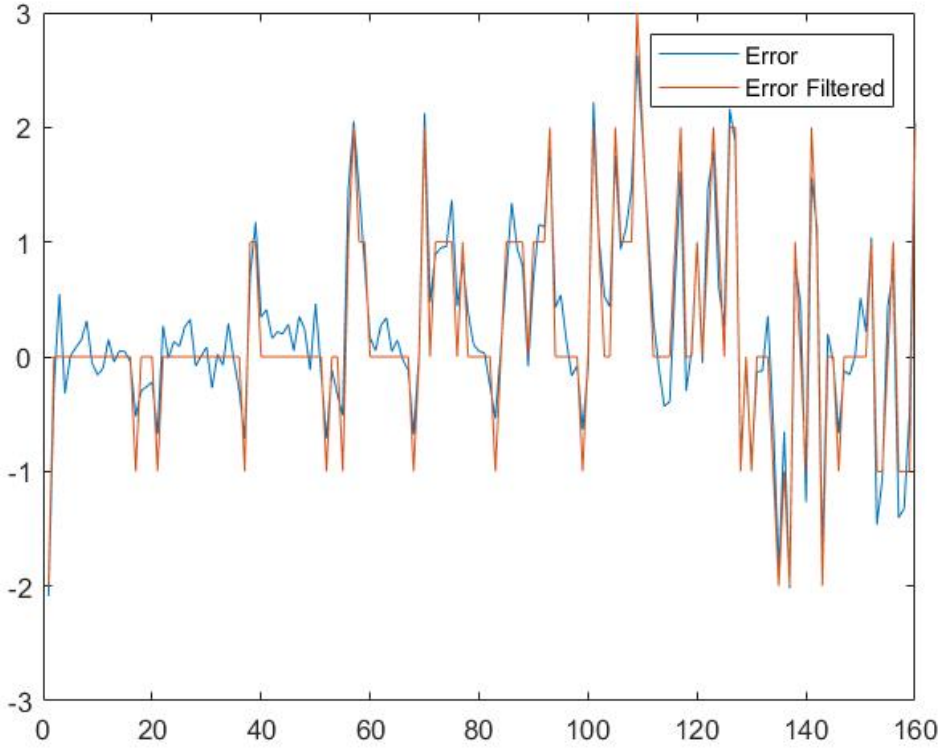


Figure 6.30: Effect of the filter in reducing the noise on the original error signal

	Neural Network	Neural Network + Filter
Sheltering Factor Mean Error	0.2477	0.1972
Variance	0.6934	0.7208
Mean Absolute Error	0.616	0.535

Table 6.4: Results of Test Four

6.3.2 Worst Cases

Test Five

The following aerial picture represents a typical situation in an urban scenario: an industrial building, in dark grey, surrounded by other types of building. In this case, the lighter one has been considered an educational one, while the building with the red roof a residential. Hence, if we consider also the small yard corner, the area has four zones with different sheltering ability, according to what we have established in Table 2.3. To evaluate the map generation algorithm we have to make a comparison between the output of the latter and the desired one. To do that we have assigned by inspection the sheltering factor following the aforementioned classification. The resulting **M** matrix can be plotted in a colour scale for an easier and clearer comprehension and it is represented in Figure 6.32.

The output of the neural network function applied on the original image 6.31 is, as said,

Figure 6.31: Aerial Picture 5

a vector **s**, that can be re-arranged in a 10×16 grid to obtain the **S** matrix, dimensionally equivalent to **M**. This last one can be plotted in the same colour scale as the previous to have an immediate comparison of the result. The map obtained with the map generation algorithm is shown in figure 6.33, while figure 6.34 represents the sheltering map that outcomes from the the application of the filter on the previous.

At a first sight, the result is very bad, since the algorithm is not able to identify the buildings, not even in their shape. The definition of the educational and residential buildings is, despite being poor, at least outlined, but the sheltering factor associated to the

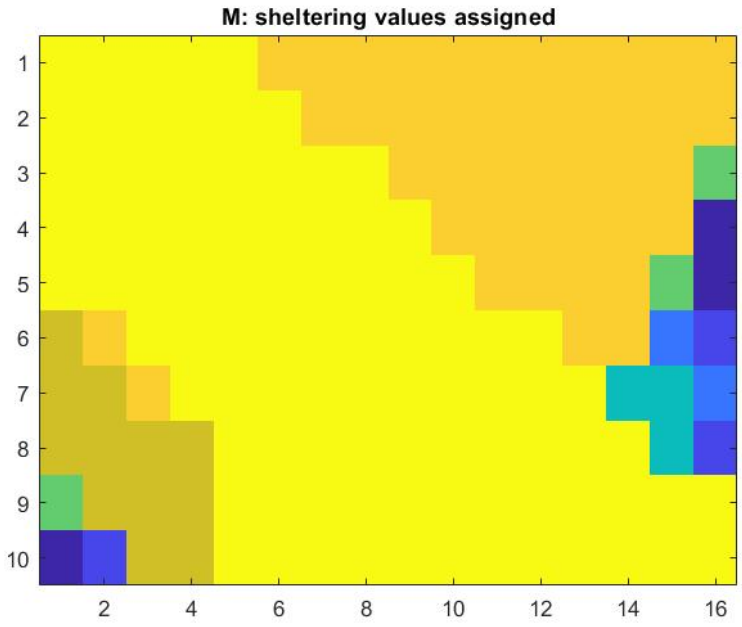


Figure 6.32: Sheltering map assigned by inspection of Figure 6.31

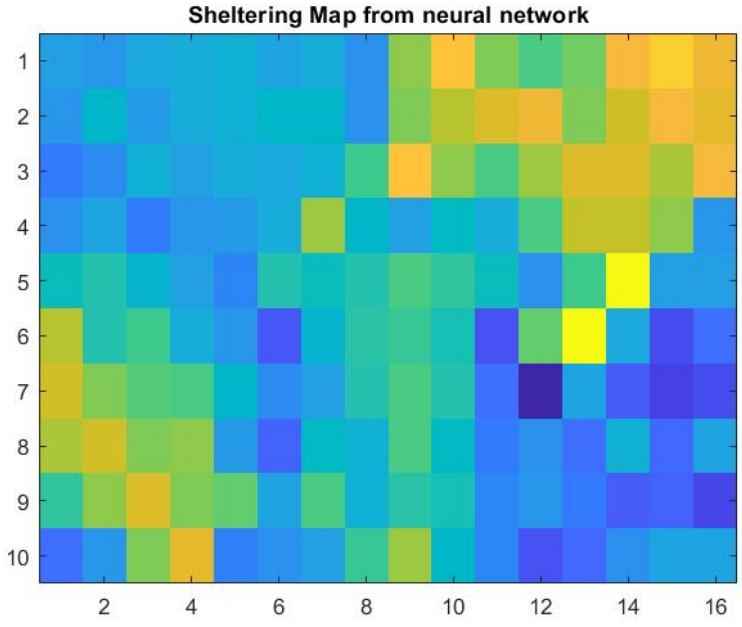


Figure 6.33: Sheltering map obtained with neural network function applied to Figure 6.31

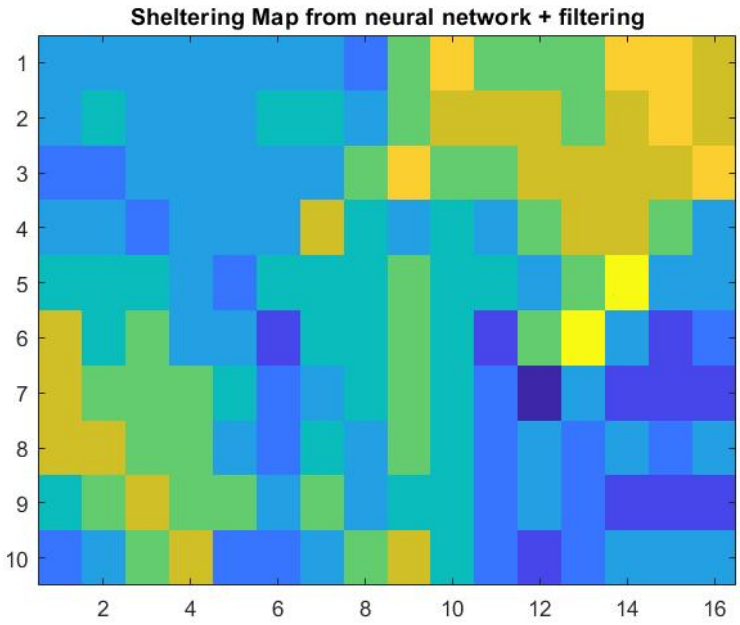


Figure 6.34: Sheltering map obtained with neural network function and filter applied to Figure 6.31

industrial building is completely wrong, lower that it should be. This map is absolutely insufficient to perform map planning, since, as a matter of fact, it does not carry useful information.
The filtered map does not show any visible improvement beside making the values, and hence the colours, more homogeneous and discrete as expected the Path Planner.
Leaving aside the graphical inspection, we must focus on the more precise error analysis. Once again, the error is the difference between the values of maps 6.33 and 6.34 and the one of the desired sheltering map 6.31. Histograms in figures 6.35 and6.36 show the distribution of errors in the 160 sub-parts that constitute the map.

As we expect the histogram of the filtered map have discrete values, corresponding to the levels of sheltering factor given in 2.3, while 6.35 shows a continuous distribution. None of them is centred in zero and to verify it, we have computed the mean error and the variance and they are reported in Table 6.5, together with the mean absolute error. As explained in the previous section, the MAE is able to quantify not only the overall number of errors, but also their entity and therefore we use it to evaluate the correctness of the map generation algorithm.
As clear from table6.5, MAE is very high, in particular it is the highest we have found, and the addition of the filter has a negative effect, increasing it by an additional 0.05. In general, having a MAE around 3.4 is a bad result, but the real problem here is the distribution of the error, that is basically spread all over the map, preventing any possible

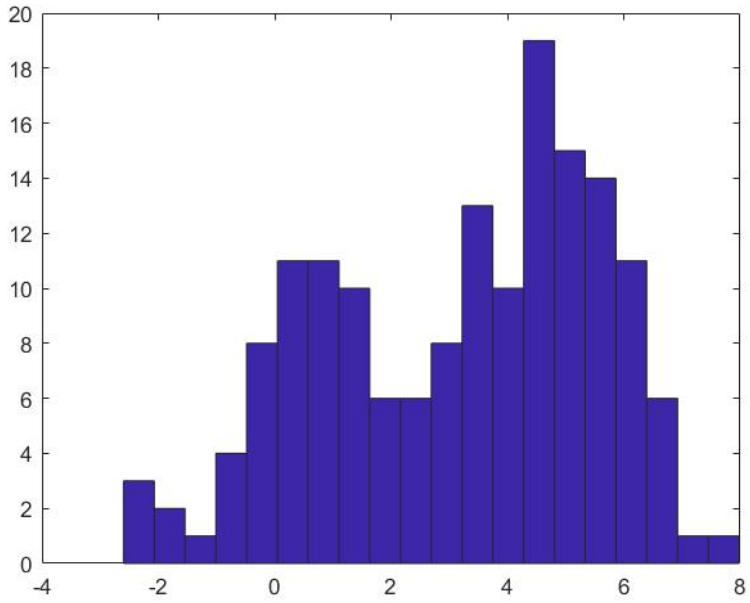


Figure 6.35: Distribution of the errors between sheltering maps 6.33 and 6.32

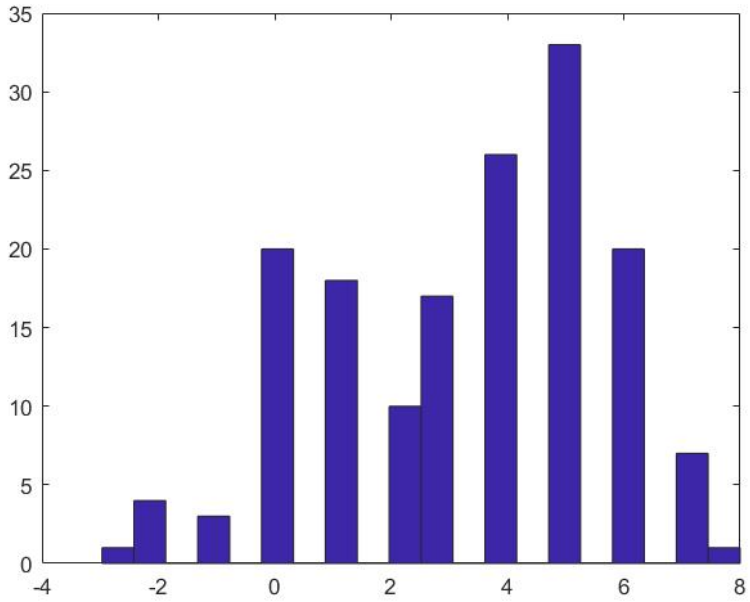


Figure 6.36: Distribution of the errors between sheltering maps 6.34 and 6.32

use of it.
Finally, figure 6.37 shows the plots of error vector both filtered and unfiltered: the filter applied to the error vector, in theory, should be able to reduce some minor noise and to discretize the sheltering factor values on the expected levels, but, as seen, its results are poor as well.
Since the whole map generation algorithm is based on colour analysis, the bad results are probably due to the colour associated to the industrial building: the dark grey is too similar to the colour of the asphalt of the streets that the neural network has been trained to associate to a zero sheltering. This may cause a confused definition of the sheltering factor that, as a matter of fact, is defined to be a value in between the free area and the industrial building. After some other simulations, this result will be discussed in section 6.3.3.

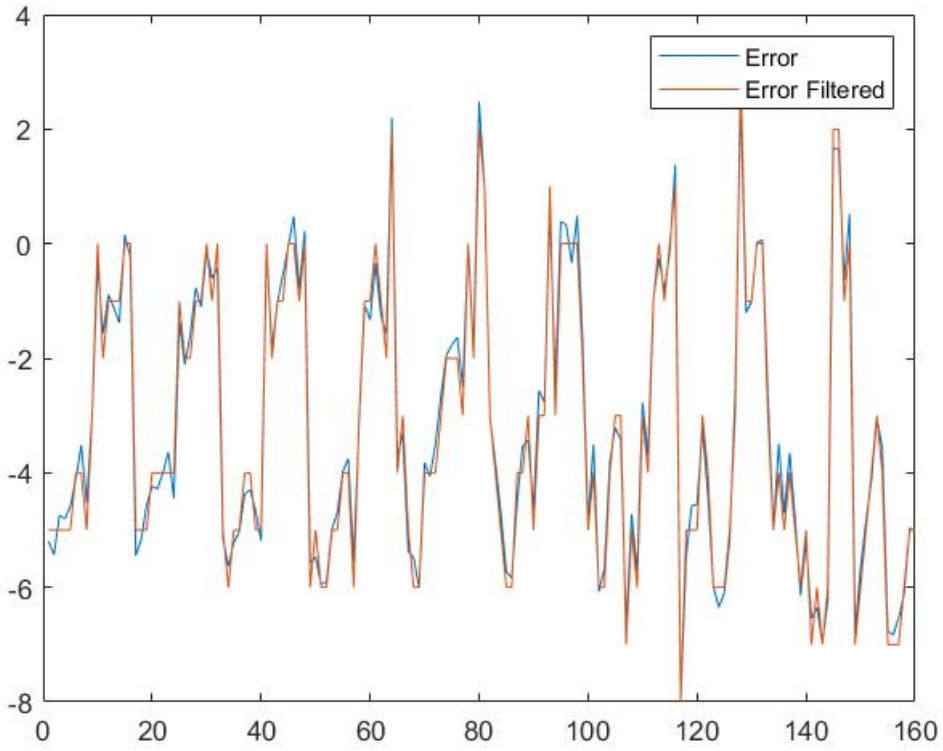


Figure 6.37: Effect of the filter in reducing the noise on the original error signal

	Neural Network	Neural Network + Filter
Sheltering Factor Mean Error	-3.1867	-3.2617
Variance	5.6263	5.7876
Mean Absolute Error	3.390	3.440

Table 6.5: Results of Test Five

Test Six

The following aerial picture represents a typical situation in an urban scenario: a place with and overlooking building and some sparse trees. The area has basically two zones with different sheltering ability, according to what we have established in Table 2.3. To evaluate the map generation algorithm we have to make a comparison between the output of the latter and the desired one. To do that we have assigned by inspection the sheltering factor following the aforementioned classification. The resulting **M** matrix can be plotted in a colour scale for an easier and clearer comprehension and it is represented in Figure 6.39. The output of the neural network function applied on the original image 6.38 is,

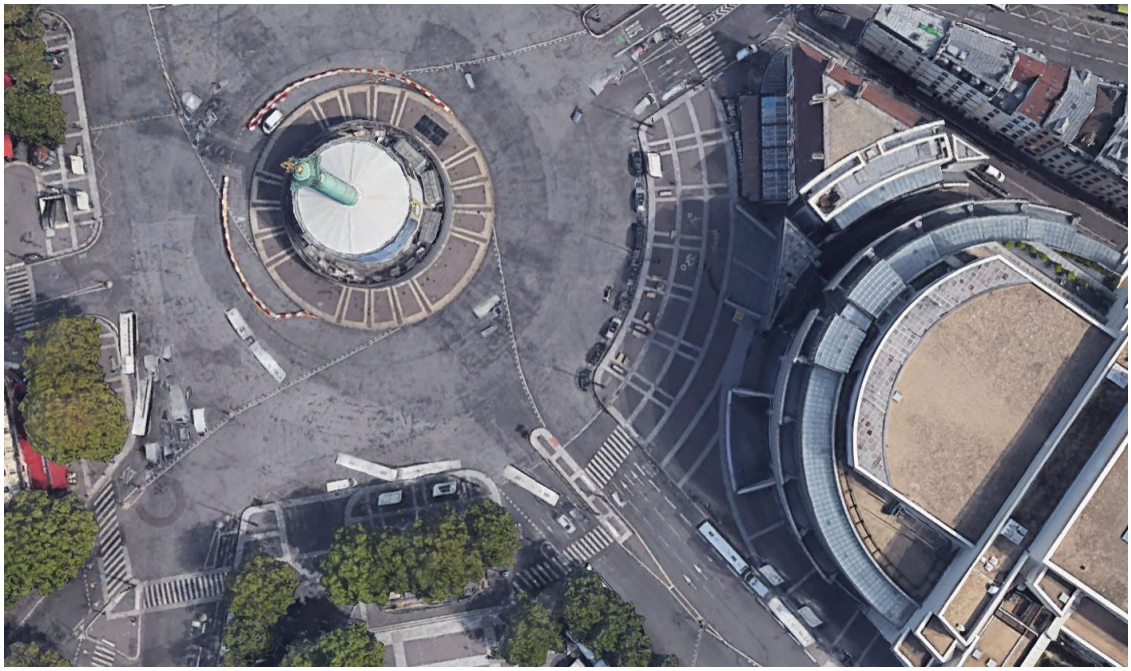


Figure 6.38: Aerial Picture 6

as said, a vector **s**, that can be re-arranged in a 10×16 grid to obtain the **S** matrix, dimensionally equivalent to **M**. This last one can be plotted in the same colour scale as the previous to have an immediate comparison of the result. The map obtained with the map generation algorithm is shown in figure 6.40, while figure 6.41 represents the sheltering map that outcomes from the the application of the filter on the previous.

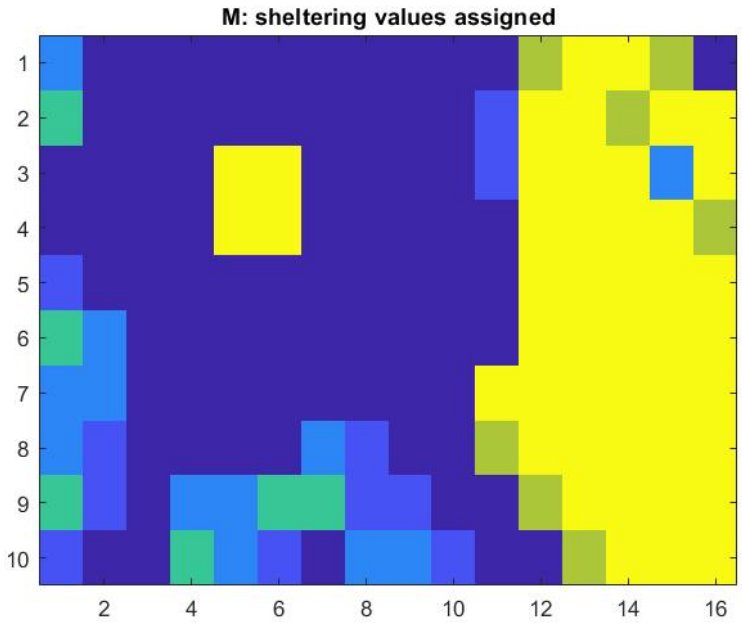


Figure 6.39: Sheltering map assigned by inspection of Figure 6.38

At a first sight, the result is very bad, since the algorithm is able to identify the mon-

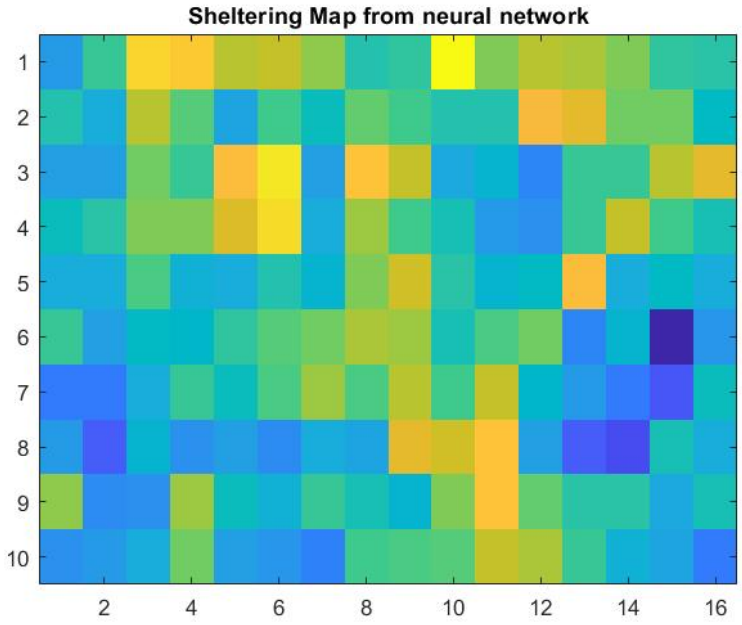


Figure 6.40: Sheltering map obtained with neural network function applied to Figure 6.38

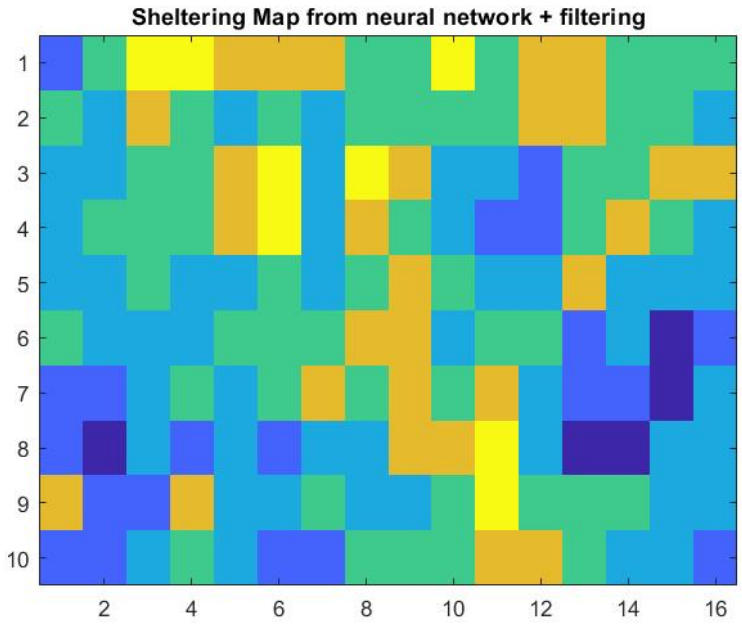


Figure 6.41: Sheltering map obtained with neural network function and filter applied to Figure 6.38

ument in the centre of the place but not the ones in the right hand side of the picture. Moreover it completely misunderstands the rest of the image, assigning high values of sheltering factor to completely free areas. Hence this map is not only insufficient to perform any kind of path planning, but it is also potentially dangerous, since it would make the UAV overly areas that do not offer any kind of sheltering to the people on the ground. The filtered map does not show any visible improvement beside making the values, and hence the colours, more homogeneous and discrete as expected the Path Planner. On the contrary, by doing that, it lowers the sheltering factor of the sub-parts actually occupied by the building, making it practically transparent.
Leaving aside the graphical inspection, we must focus on the more precise error analysis. Once again, the error is the difference between the values of maps 6.40 and 6.41 and the one of the desired sheltering map 6.38. Histograms in figures 6.42 and6.43 show the distribution of errors in the 160 sub-parts that constitute the map.

As we expect the histogram of the filtered map have discrete values, corresponding to the levels of sheltering factor given in 2.3, while 6.42 shows a continuous distribution. None of them is centred in zero and to verify it, we have computed the mean error and the variance. this is one of the cases we were referring to when expressing the need of the mean absolute error: in fact the errors are so distributed and various that positive and negative values cancel themselves, resulting in an average error very close to zero. However it is clear that the maps generated are completely wrong and that is the reason for which the only way to evaluate the performance of the algorithm is to use MAE. All

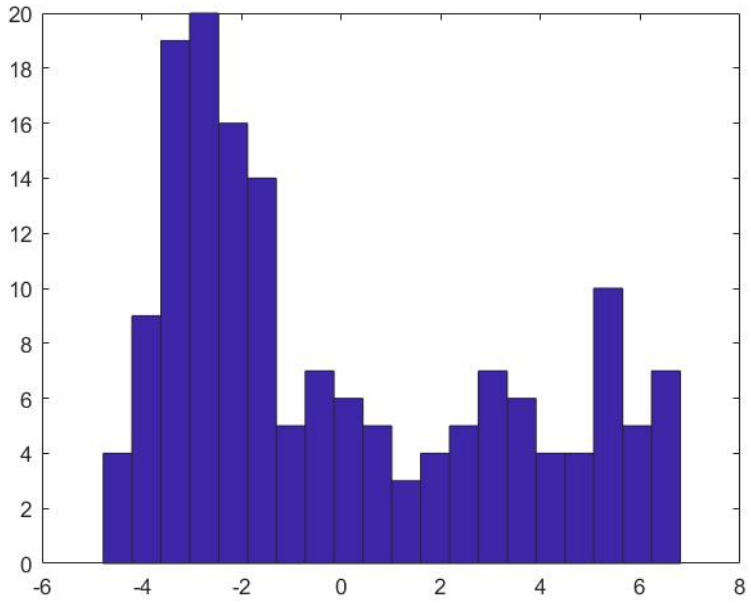


Figure 6.42: Distribution of the errors between sheltering maps 6.40 and 6.39

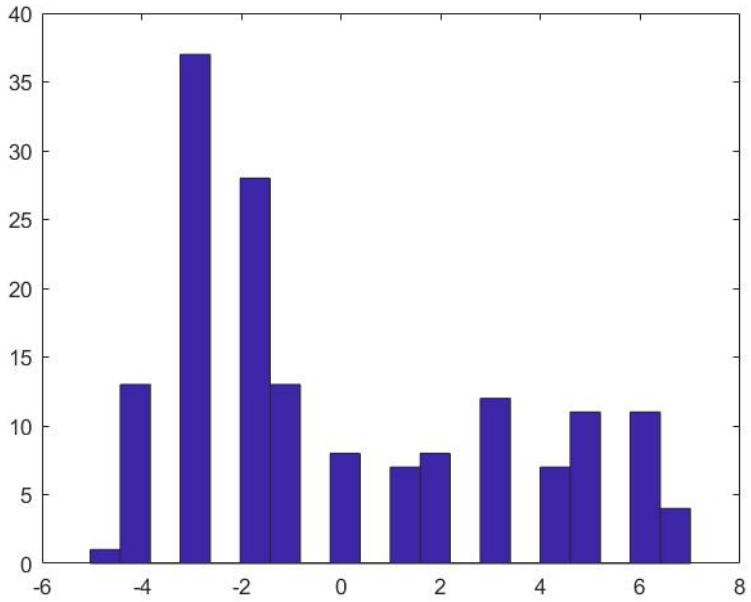


Figure 6.43: Distribution of the errors between sheltering maps 6.41 and 6.39

these three values are reported in Table 6.6, both for the filtered and unfiltered maps. As clear from table6.6, MAE is high, very close to 3, and the addition of the filter has a slight positive effect, reducing the mean absolute error by 0.003. In general, having a MAE around 3 is a bad result, but the real problem, in this case as well, is the distribution of the error, that is basically spread all over the map, preventing any possible use of it. Finally, figure 6.44 shows the plots of error vector both filtered and unfiltered: the filter applied to the error vector, in theory, should be able to reduce some minor noise and to discretize the sheltering factor values on the expected levels. The filtered signals follows almost perfectly the unfiltered one and in fact the impact of the filter is almost negligible. Again, since the whole map generation algorithm is based on colour analysis, the bad results are probably due to the colour of the buildings: it is too similar to the colour of the place itself and to other situations that the neural network has been trained to associate to a zero sheltering. This, as seen, cause a confused definition of the sheltering factor. After the next simulation, this result will be discussed in section 6.3.3.

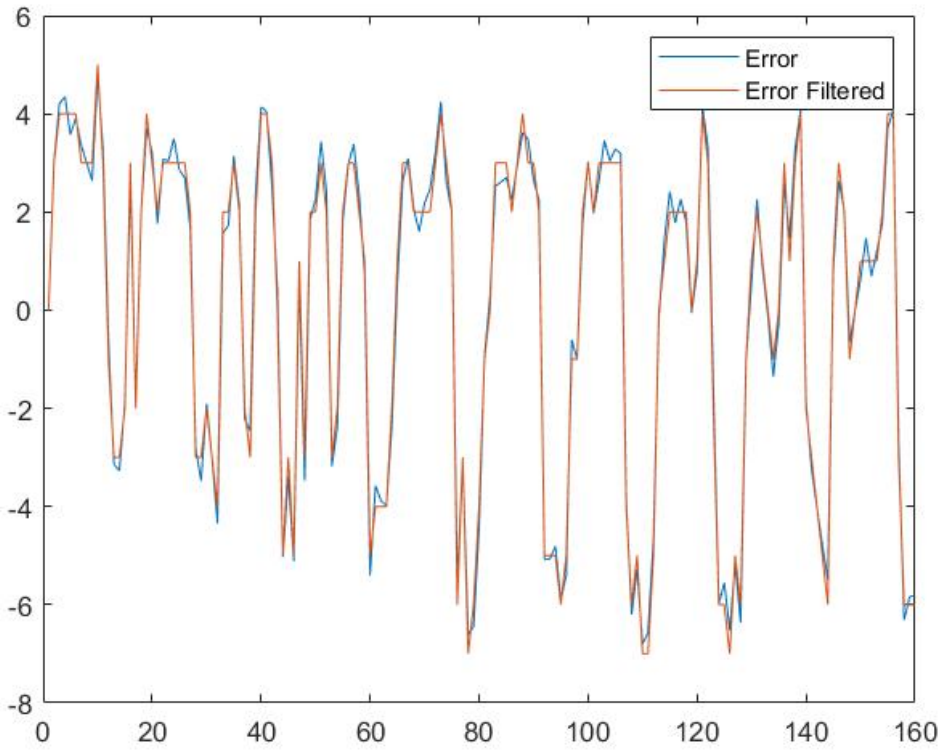


Figure 6.44: Effect of the filter in reducing the noise on the original error signal

	Neural Network	Neural Network + Filter
Sheltering Factor Mean Error	0.0040	0.0129
Variance	11.4081	11.4048
Mean Absolute Error	2.954	2.951

Table 6.6: Results of Test Six

Test Seven

The following aerial picture represents a typical situation in an urban scenario: an industrial building, in dark grey, on one side of the street, and residential building on the other. Hence, the area has three main zones with different sheltering ability, according to what we have established in Table 2.3. To evaluate the map generation algorithm we have to make a comparison between the output of the latter and the desired one. To do that we have assigned by inspection the sheltering factor following the aforementioned classification. The resulting **M** matrix can be plotted in a colour scale for an easier and clearer comprehension and it is represented in Figure 6.46.

The output of the neural network function applied on the original image 6.45 is, as said,

Figure 6.45: Aerial Picture 7

a vector **s**, that can be re-arranged in a 10×16 grid to obtain the **S** matrix, dimensionally equivalent to **M**. This last one can be plotted in the same colour scale as the previous to have an immediate comparison of the result. The map obtained with the map generation algorithm is shown in figure 6.47, while figure 6.48 represents the sheltering map that outcomes from the the application of the filter on the previous.

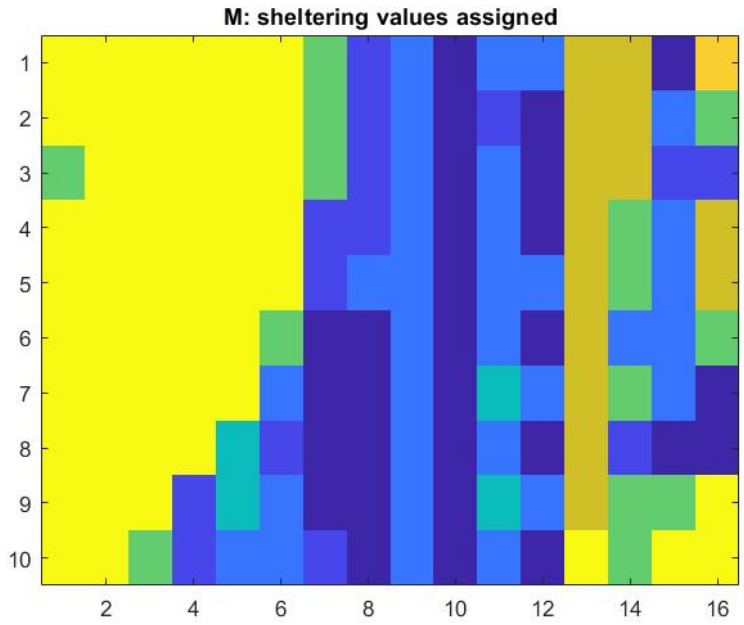


Figure 6.46: Sheltering map assigned by inspection of Figure 6.45

To an observation of the maps, the result is bad, since the algorithm is able to identify

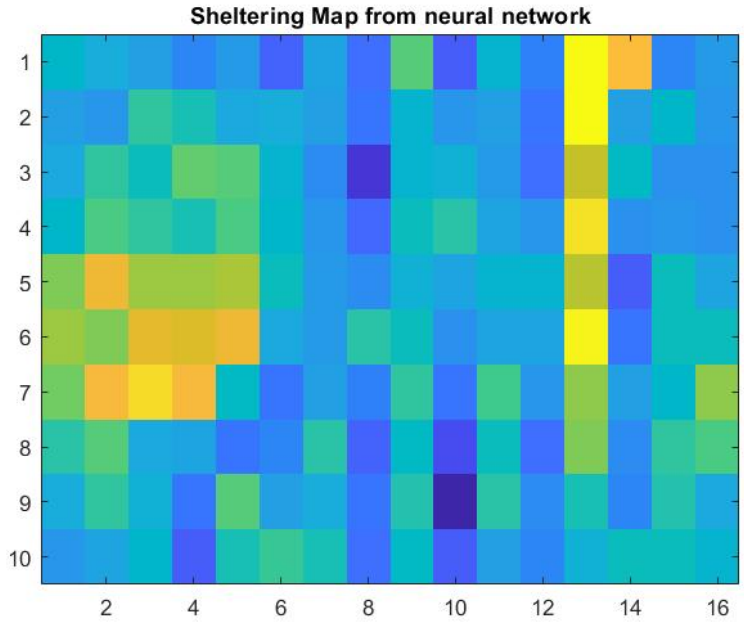


Figure 6.47: Sheltering map obtained with neural network function applied to Figure 6.45

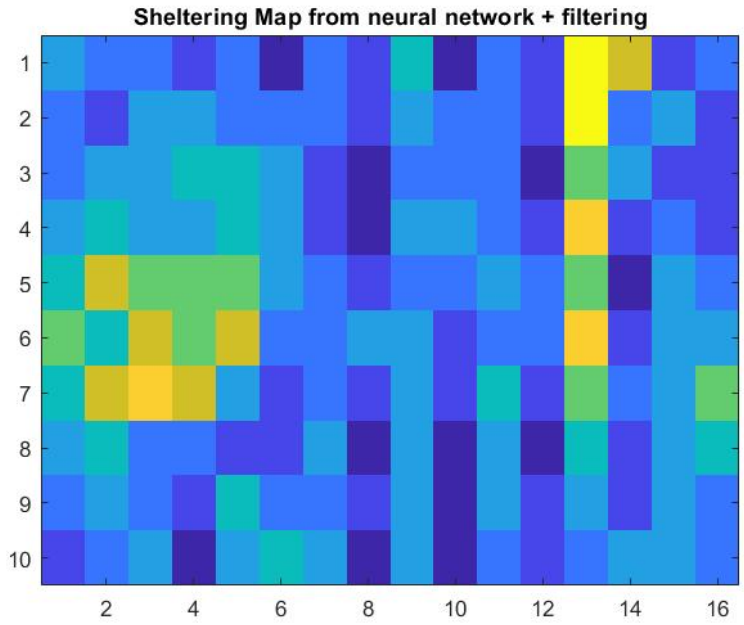


Figure 6.48: Sheltering map obtained with neural network function and filter applied to Figure 6.45

the residential buildings, even though their definition is poor and underestimated, but it does not detect the industrial one, not even in its shape. As a result, the sheltering factor associated to the sub-parts of the industrial building is completely wrong, lower that it should be, with the exception of some of them. In this case, the algorithm does underestimate the left hand building without overestimating other areas of the image: in some way this could be considered as conservative, therefore less dangerous for people than what happened in the previous tests. However this map does not represent the scenario offered by the satellite picture and, as a consequence, it is absolutely useless to perform map planning, since, as a matter of fact, it does not carry useful information.
The filtered map makes the values, and hence the colours, more homogeneous and discrete as expected the Path Planner. In particular, almost everywhere it lowers the values and this improves the definition of the street in between the buildings, worsening, ot the other hand, the analysis of the industrial building.
Leaving aside the graphical inspection, we must focus on the more precise error analysis. Once again, the error is the difference between the values of maps 6.47 and 6.48 and the one of the desired sheltering map 6.45. Histograms in figures 6.49 and6.50 show the distribution of errors in the 160 sub-parts that constitute the map.

As we expect the histogram of the filtered map have discrete values, corresponding to the levels of sheltering factor given in 2.3, while 6.49 shows a continuous distribution. We have computed the mean error and the variance and they are reported in Table 6.7, together with the mean absolute error. As explained in the previous section, the MAE is

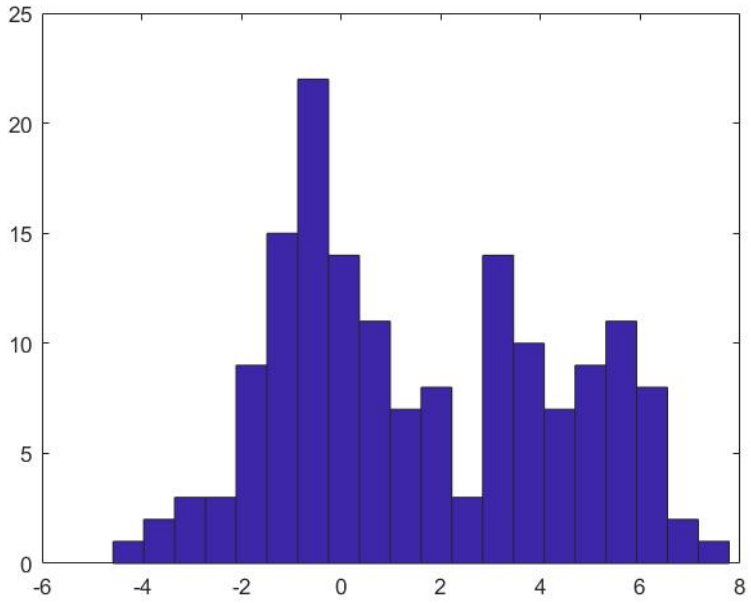


Figure 6.49: Distribution of the errors between sheltering maps 6.47 and 6.46

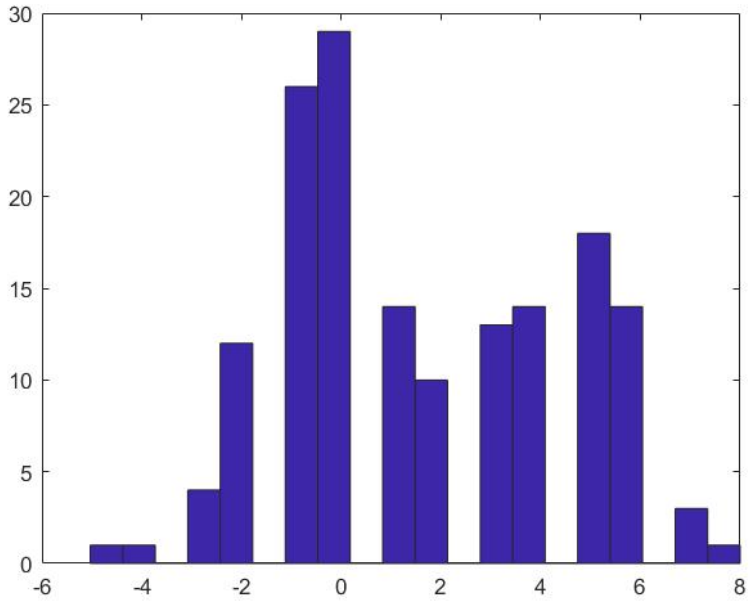


Figure 6.50: Distribution of the errors between sheltering maps 6.48 and 6.46

able to quantify not only the overall number of errors, but also their entity and therefore we use it to evaluate the correctness of the map generation algorithm.
As clear from table6.7, MAE quite high and the addition of the filter has a slight, positive effect, reducing it by 0.026. In general, having a MAE around 2.5 is a bad result, but unlike the previous tests, this case shows a situation in which the sheltering factor has been underestimated in one zone only. Hence a trajectory respecting the safety criteria is difficult to be found, but this error does not lead to harmful consequences.
Finally, figure 6.51 shows the plots of error vector both filtered and unfiltered: the filter applied to the error vector is able to reduce some minor noise and to discretize the sheltering factor values on the expected levels. It follows very precisely the unfiltered signal and in fact the impact of the filter in this case is quite low.
Once again, since the whole map generation algorithm is based on colour analysis, the bad results are probably due to the colour associated to the industrial building: the dark grey, even at a visual inspection, is too similar to the colour of the asphalt of the street that the neural network has been trained to associate to a zero sheltering. This may cause a confused definition of the sheltering factor that, as a matter of fact, is defined to be a value in between the free area and the industrial building. This result will be discussed in section 6.3.3.

	Neural Network	Neural Network + Filter
Sheltering Factor Mean Error	-1.6214	-1.6426
Variance	7.9043	8.0908
Mean Absolute Error	2.548	2.522

Table 6.7: Results of Test Seven

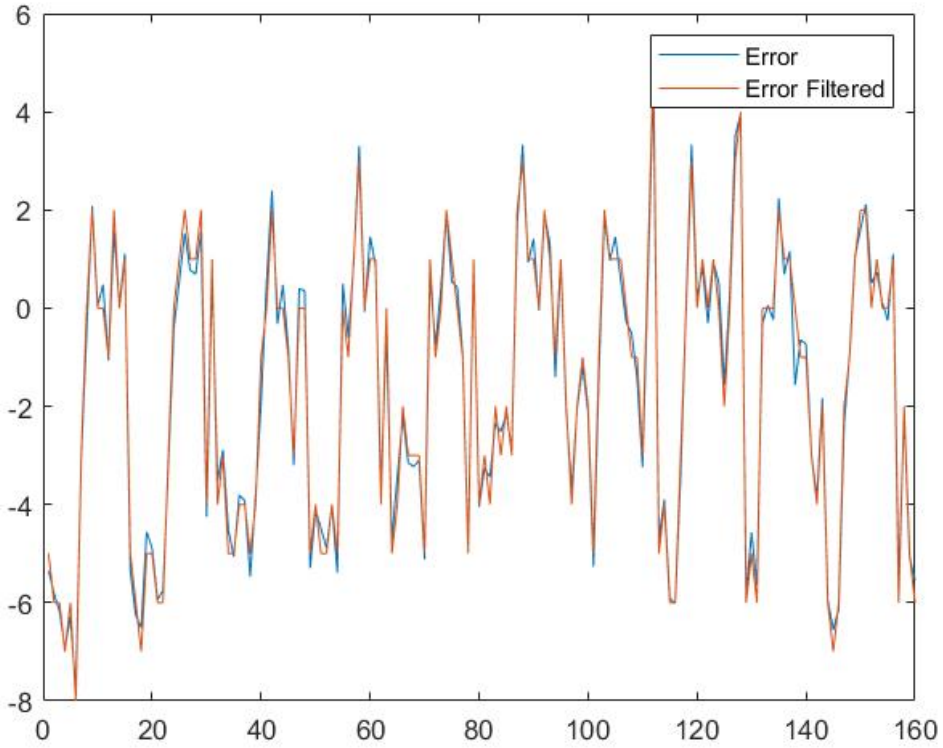


Figure 6.51: Effect of the filter in reducing the noise on the original error signal

6.3.3 Overall results

The seven examples shown in the previous sections are the best and the worst cases we have obtained in our simulations and their results varies from being excellent to completely unusable. Before investigating on the possible causes, it is important to extend the analysis to the whole original data set of 200 satellite pictures, in order to evaluate the performance of the map generation algorithm in general. To do that we have repeated the the simulation with all the images, collecting in two vectors, on one hand, all the sheltering values obtained through the algorithm, on the other, the desired sheltering factor assigned by inspection. Again, the error between the two is their difference, represented by a 32000 elements vector. The histogram that indicates the error distribution along the whole data set is reported in figure 6.52, while figure 6.53 shows the one resulting from the application of the filter after the map generation algorithm. As expected, the latter reports discrete values, corresponding to the levels of sheltering factor given in Table 2.3, while the former shows a continuous distribution.

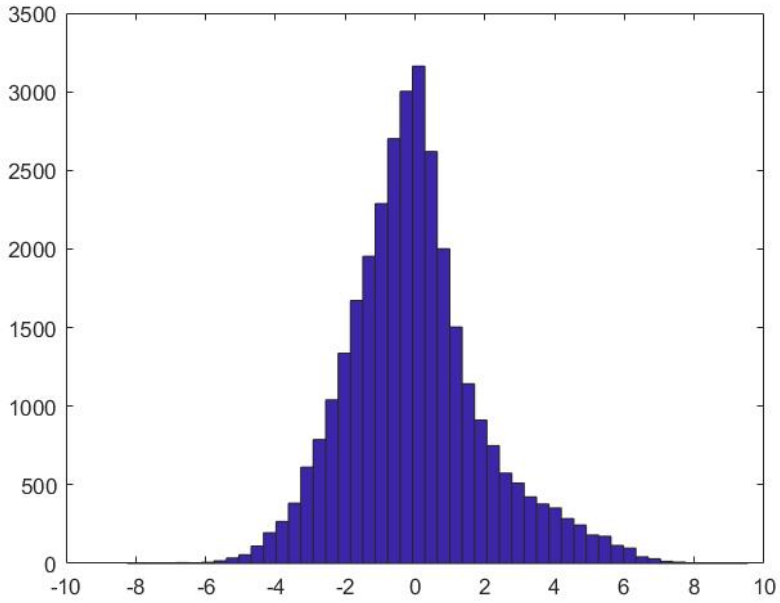


Figure 6.52: Distribution of the errors between the output of the unfiltered algorithm and the desired values of sheltering factor, computed on the whole data set of 200 pictures

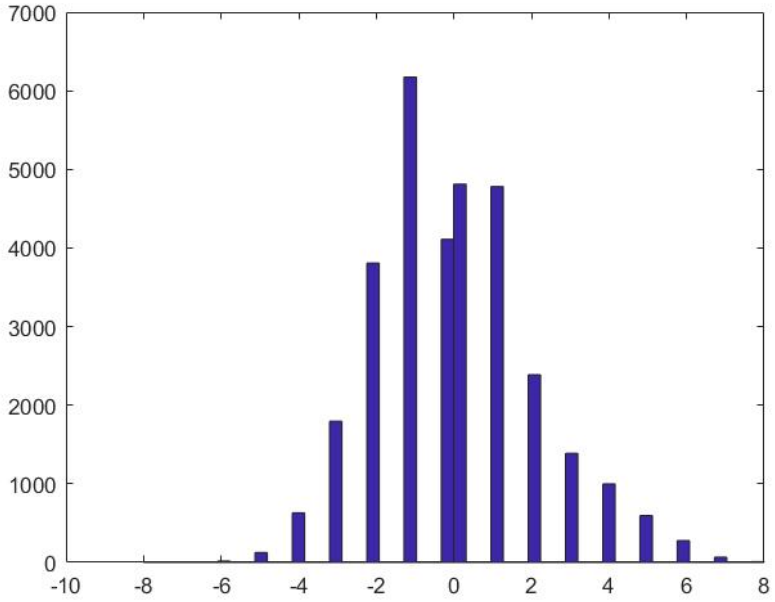


Figure 6.53: Distribution of the errors between the output of the filtered algorithm and the desired values of sheltering factor, computed on the whole data set of 200 pictures

We have computed the mean error and the variance for both the distributions and the results are reported in Table 6.8. However, as written before, positive and negative values cancel themselves in the computation of the average value and therefore we have computed the Mean Absolute Error to take into account errors in general, regardless if they are due to over or underestimation. A mean absolute error like the one we have obtained on a 10 levels sheltering factor scale, can, in some way, be seen as percentages equal to 14.62% and 14.22% respectively, representing the average part of the map that is subject to error. Following this idea we could define the average "correctness" of the algorithm estimation as 85.38% and 85.78% in the case of the filtered algorithm. This, in principle, could be a way to have an immediate evaluation of the performance of the map generation algorithm. These values are quite high and hence we could be tempted to state that the algorithm has proven to be successful. However, as seen in the previous examples, its performance vary a lot, with MAE spanning from 0.535 , in the best case, to 3.44, in the worst case and it is important to investigate the cause of this behaviour.

	Neural Network	Neural Network + Filter
Sheltering Factor Mean Error	-0.0028	0.0184
Variance	3.8064	3.8590
Mean Absolute Error	1.462	1.422

Table 6.8: Overall results obtained on the whole data set

The limits of the map generation algorithm are due to the process of image analysis: in fact it is based mainly on colour analysis of the sub-parts of the satellite picture. Once extracted the colours, they are fed as input to the neural network function. The latter has been trained through supervised learning, meaning that, to build the fitting function, it starts to associate specific colours to specific desired values defined by inspection. In particular, when assigning the latter, we have decided to assume that every building with a red roof is a residential building, while white or light grey roof indicate an educational building and a dark grey or black one is recognized as an industrial building. Moreover it is obvious that the zero sheltering level has been associated to a large variety of situations, from meadows to squares and streets, and probably this has led to a non clear, mixed, definition of the colour profile for the level. Red and light grey/white roofs are generally well interpreted by the neural network function, that makes on them very few minor errors, and associates them to the right level of sheltering. On the contrary, dark grey is probably too similar to the asphalt of the streets and the flooring of the places and therefore the neural network interpolates between the two very different values, coming to an intermediate definition that does not represent well neither the dark roof buildings, nor the low sheltering areas. This misinterpretation is systematic: all of the images of the set, whose map has a mean absolute error higher that 2, show a dark grey preponderance. Nonetheless a distinction has to be made: associating to a building a lower than expected sheltering factor, can be seen as a conservative act, since the only effect is to reduce the

probability that the trajectory chosen by the Path Planner overflies that building. This of course can be a problem in terms of path planning but it is not harmful for the humans. On the contrary, a low sheltering area that is associated an higher value, leads to a wrong risk analysis and therefore it can be seen by the Path Planner as a safe area to overfly. This may cause severe to catastrophic consequences in terms of injuries and fatalities.

Chapter 7

Map Manager and Path Planning

In this chapter we give a brief explanation on how the sheltering map produced by the algorithm we have developed is used by the Cloud-Based UAV Traffic Manager to handle the risk assessment, management and the flight of the UAV.
As seen in chapter 2 the risk management phase is crucial for the UAV to be compliant with the Target Level of Safety required by the aviation standards. In order to have a precise quantification of the overall risk, the sheltering map generation algorithm previously developed must be integrated in a tool that can be called Map Manager. The purpose of the latter is, given a satellite picture and a mission, to generate a risk map that can later be used by the Path Planner to find, among all the possible ones, the trajectory to be followed by the UAV to accomplish the mission, keeping the risk level as low as possible. Once it has been defined, a Path Validator evaluates if the UAV is able to fly guaranteeing the safety standards and, based on that, gives authorization or denial to the flight.
Before focusing on each of these phases, it is important to define precisely mission's overall risk and how it can be mapped:

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.

f_F must be evaluated using the risk assessment techniques covered in chapter 2.
The previous definition refers to a punctual resolution of the risk map, however we have decided to analyse instead the area divided in portions, or sub-parts, mainly for two reason. First of all, even having access to the high computational power given by the cloud architecture, using such a resolution is practically infeasible since no one of the information required to evaluate the risk is defined in a point, whereas it is in an area. Then, even more important, when considering the dynamic of a UAV, it is clear that a punctual trajectory and flight are just a simplification, not feasible in fact, while it is definitely more viable to work on areas that can be considered uniform for what concerns risk analysis. According to [27], an area of $25m^2$ guarantees the manoeuvrability of a UAV like the one we are considering in an urban environment. That is the reason why, since the beginning

of image processing, we have decided to divide satellite images is $25m^2$ sub-parts.

7.1 Map Manager

According to the previous definition, the risk map is bidimensional matrix based on the original picture of the area in which the mission takes place. Every cell of this matrix corresponds to a specific sub-part i and it contains its risk value $f_{F,i}$, that has to be computed knowing the following parameters:

- Population density, possibly in real-time
- Mass of the drone
- UAV flight altitude
- Sheltering factor of that area
- Ground impact frequency in the case of early flight termination

Once defined the distribution of these in the area, the risk value for each sub-part i, can be obtained, with reference to equation 2.17, as:

$$f_{F,i} = N_{i,exp} \times P(fatality|exposure)_i \times f_{EFT,i} \tag{7.1}$$

Moreover, the map manager must be aware of the possible presence and position of No-Fly Zones, that are areas in which it the previous standard equation can not be used, since they must be modelled with $f_F = \text{inf}$.

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.

These kind of zones were originally introduced by military aviation to define areas in which flight was forbidden. However in the last years, the increase of usage of UAVs for civil applications and all the risk connected and shown in chapter 2, have led to the need of defining them in some situations in an urban scenario as well. They may be established to protect some specific areas, permanently or in case of events, like concerts or sport matches, like it was done in London for the 2012 Olympic Games. Hence two kinds of No-Fly Zones are possible:

Law-Imposed No-Fly Zone that include all the areas that can not be overflown by an aircraft due to some legislative restriction. These zones are prohibited no matter the altitude of the flight.

Structural No-Fly Zone that include all the situations is which a natural or artificial structural obstacle impedes the flight of the UAV. It is the case of all those elements

higher than the altitude of the drone, like trees or building. In fact, even though they do not express an absolute prohibition to fly, it is clear that if the trajectory of the UAS includes one of these areas, a crash would happen, with possible harmful consequences and the abortion of the mission. Unlike the previous case these No-Fly zones depend on the altitude of the flight.

Some methods to identify and model NFZ can be found in [27][34][40]. This thesis will not focus on this aspect any further, but it is important to notice that a complete map manager must be aware of No-Fly Zones.
Since the purpose of this work is the generation of a sheltering map, we can make some assumptions. First of all, we consider a planar flight, that is to say that the altitude of the UAV is supposed to be constant. By doing that we are basically assuming that the unmanned system does not modify its coordinates on the z-axis and this reduces the computational power required to compute the risk value. Moreover we suppose the population density to be constant and this is reasonable on a fairly small area like the one we are considering. Finally, since the mass of the UAV and the frequency of ground impacts due to early flight termination are constant, we can reduce the analysis from the risk map to the sheltering map. For this reasons we have worked with these kind of maps and we are referring to them in the next sections. Therefore the cases shown in chapter 6 are examples of the outcome of the Map Manager.

7.2 Path Planner

The path planning is a well known problem in literature and many different approaches can be used to solve it, according to the characteristics and the needs of the robot that has to follow the trajectory. According to a quite common definition:

Definition 7. 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[1].

This means that, once known the robot dynamic and the description of the environment and given an ordered set of goal states, the Path Planner has to find a sequence of waypoints on the map that is able to bring the aircraft from the starting state to the following and on, until the ending one is reached.
Both probabilistic and deterministic algorithms can be used to solve the problem. Each one of these categories present advantages and disadvantages, especially depending on the size of the map and the computational power required. In a case like the one we are treating, we are dealing with:

[1]The definition usually refers to robots that have planar motion, however we have assumed the UAV altitude to be constant, so the flight can be considered like a motion on the $x - y$ plane only.

1. Large Scale: cities have an area of many km^2 typically, and a problem like path planning needs of course an high resolution of the map. Therefore, in order to have a solution in finite time, the algorithm must be light and supported bu an high computational power.

2. Changing environment: obstacles and in general the risk parameters may changes in time, even during the flight. The Path Planner have to be dynamical, able to change path in real-time.

3. Critical Situation: Path Planner response has to be fast, with no latency, in order to prevent collisions and consequent possible injuries and fatalities.

4. Structure of the UAV: Path Planner has to take into account that the UAV is not a point mass, but instead it has its own dynamic and structure. Therefore, as said before, trajectories have of course to be optimal, but, even more importantly, they have to be feasible for the unmanned vehicle.

According to the algorithm chosen, the path planner performs a series of computations to find all the feasible trajectories and, among them, it finds the one that presents the lowest risk for humans to be injured or killed. We are only considering this aspect since this work is focused on safety criteria and risk mitigation; however the path planner in general are required to build maps, known as *cost maps*, taking into account more variables and not just risk. Sometimes, in fact, it is useful, or even necessary, for the mission to find the best trade off between different parameter. It is the case, for example, of flight time: small UAVs like the one we are considering to use have a low battery life (usually below four hours), so the optimization of the flight time may be useful to maximize the operativeness and avoid wasting battery. In the same way, since the UAV is connected to the cloud via internet, it is important that during the path planning of a mission, the intensity of the signal is take into account so that the UAV is not blind. The same happens for those situations in which a real-time video streaming from the drone is crucial for the accomplishment of a mission.
In general, to build a cost map, the overall cost for each of the cells is:

$$C_i = \alpha \times R_i + \beta \times (B_i) + \gamma \times G_i ... \tag{7.2}$$

where:
C_i is the overall cost of the i-th cell.
R_i is the risk of the i-th cell, normalized
B_i and G_i are the other factors of the i-th cell that may influence the choice of the best path.
α, β and $\gamma$are weighting factors in the range [0,1], so that $\alpha + \beta + \gamma = 1$
Of course if the number of factors to be taken into account grows, the previous equation may be expanded by adding new weights so that the overall weights sum is equal to 1 [27].
The most important and delicate task that the path planner, and in general the CBUTM,

has to perform is the choice of the best weighting values to have the best compromise. As a matter of fact, if α is very high with respect to the others, safety is the main criteria to choose the path, but this situation can lead to practically not considering the other parameters. If, on the contrary, α is too low, safety can be considered less important than other request, with a possible outcome of a path that does not respect the safety criteria and is therefore rejected by the Path Validator.

7.3 Path validation and risk acceptance

Path validation is probably the most important phase preceding the take off of a UAV. In fact it constitutes the real interface between the analysis performed so far, with risk assessment, map generation and path planning, and the regulations imposed by national flight agencies to UAVs.

Definition 8. The Path Validator takes as input the trajectory chosen by the Path Planner and provides as output the authorization or denial for the flight of the UAV.

It is easy to understand the fundamental role played by the Path Validator that, as a matter of fact, decides whether the UAV in its mission is able to guarantee the Target Level of Safety (TLS) or not. In order to do that, it has to compare the risk of the overall mission to the maximum acceptable one for UAVs and, based on that, approve or deny the authorization to the flight.

$f_F \leq f_{F,Max}$ The trajectory chosen by the Path Planner guarantees the TLS imposed by regulations. As a consequence, mission is approved.

$f_F > f_{F,Max}$ The trajectory chosen by the Path Planner does not guarantee the TLS imposed by regulations. As a consequence, mission is not approved.
In this case the Path Validator has to identify which requirements have not been fulfilled, and how to change them in order to make the Path Planner define a better trajectory.

where $f_{F,Max}$, defined in Chapter 2 is:

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

The previous disequations verify that the risk value of every cell of the map is lower than the maximum imposed by the laws. However, to evaluate the overall risk of the whole trajectory with respect to the maximum one accepted, a stricter comparison has to be made, based on the predicted number of victims. This can be obtained multiplying the risk value of a certain area i by the time τ_i that the UAV spends on it. Given l_i the

maximum length of the i-th area and v the cruise speed of the UAV, τ_i can be computed as:

$$\tau_i = \frac{l_i}{v} \tag{7.3}$$

At this point it is immediate to define the predicted number of victims on a certain trajectory that overflies a total number n of sub-parts of the sheltering map of the mission:

$$N_{Miss} = \sum_{i=1}^{n} f_{F,i} \times \tau_i \tag{7.4}$$

In the same way it is possible to compute the maximum acceptable number of people victims as:

$$N_{Max} = f_{F,Max} \times \tau_{Miss} \tag{7.5}$$

where:
$\tau_{Miss} = \sum \tau_i$ is the estimation of the time length of the mission.

Once defined the values to be compared, the criteria to evaluate if the mission satisfies the standard is:

$$N_{Miss} \leq N_{Max} \tag{7.6}$$

Based on this disequation, the Path Validator defines if the given mission along the chosen trajectory is considered acceptable according the regulations imposed by the national flight authorities. If so, it gives the authorization to fly, otherwise it gives a denial and starts the re-evaluation of the weak points and of the parameters, in order to make the Path Planner choose a better, suitable trajectory. Figure 7.1 resumes with a schematic representation the whole process of map generation, path planning and validation that finally leads to mission acceptance or denial.

The previous approach to compute the number of victims on a certain path is just one of the many proposed in literature, probably the simplest. Other models take into consideration parameters like the Mean Time to Failure of the UAV or a more accurate definition of l_i and τ_i, based on the exact trajectory chosen by the path planner. However, since the path validation has not a central role in this work and it has been introduced only to provide a context to the map generation phase, the simplification level introduced by the aforementioned model are more than acceptable. However a more complete analysis on the path validation phase can be found in [27][34][40].

We are presenting now an example of map generation and path planning based on the resulting sheltering map. With reference to the satellite picture 6.10, a mission can be established from a starting point A to an ending point B, as showed in figure 7.2.

The first step is performed by the Map Manager, that takes the picture and applies the algorithm exposed in chapter 6: the image is divided in 160 sub-parts with a 10×16 grid and each of them undergoes image analysis to find the mean RGB values and edges. The obtained data are fed as input to the neural network function that has been trained with

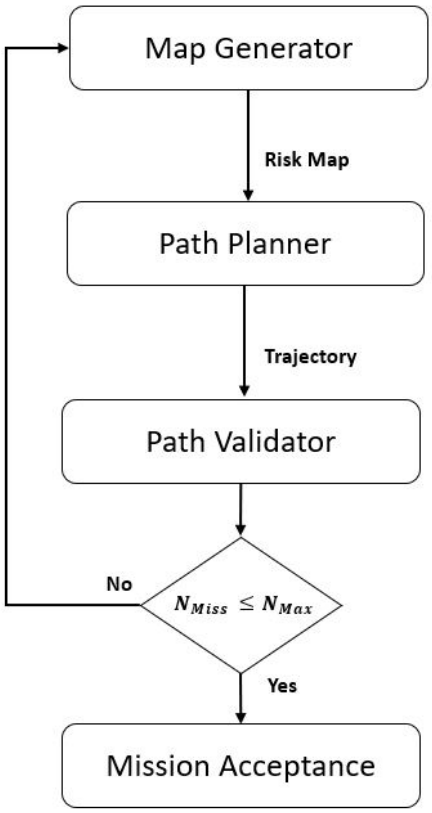


Figure 7.1: Mission risk management and acceptance

the initial data set of 200 images to automate the definition of the sheltering factor, and therefore the output is a 160 values vector that, once re-arranged in a 10×16 matrix form, constitute the sheltering map associated to the original satellite picture. Based on that, the Path Planner finds the best way in terms of safety to complete the mission: it computes all the possible trajectories between the starting and the ending point and it identifies the one among them that crosses the areas with the highest sheltering factor values possible (Figure 7.3) and that, as a consequence, presents the lowest risk for humans to suffer from injuries or fatalities due to a possible UAV or debris fall.

If the risk of every sub-part of the image is lower than the maximum one defined by the standards and if the number of possible victims along the chosen trajectory is lower than the maximum acceptable, then the mission is approved and the CBUTM communicates to the UAVs' navigation system to control the actuators in order to follow the established optimal path (7.4).

Figure 7.2: Example of mission from a point A to B

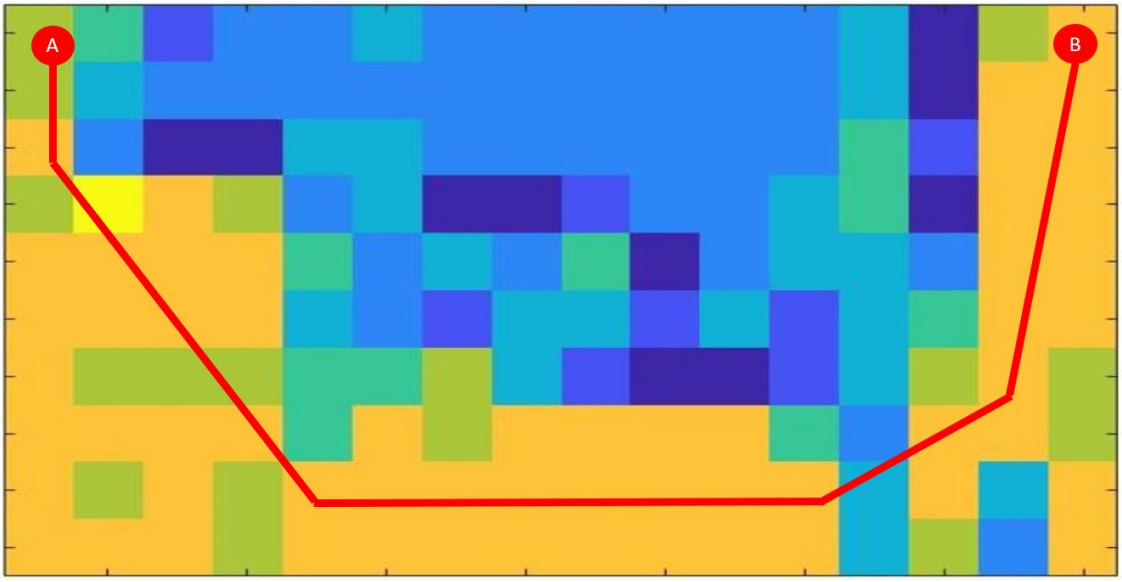


Figure 7.3: Path planning based on the sheltering map

Figure 7.4: Path followed by the UAV to minimize the risk

7.4 A cloud based architecture

The three elements presented in the previous sections are intended to be blocks of a cloud-based traffic manager. The purpose of such an architecture is to model a structured low-altitude airspace which allows UAVs operations in an urban scenario while preserving the target level of safety for humans [27][34][40]. Cloud robotics is a paradigm in which robots and automatic systems can exchange information and run programs using a common online network. As stated by [17] [25], the main advantages of cloud computing are:

Big Data The cloud gives access to a huge database of images, maps and data that every robot connected to it can access.

Cloud Computing The cloud can offer an extremely high computational power, that allows to run very heavy and complex algorithms. This possibility is crucial when addressing problems like path planning, collision avoidance or multi-robot collaboration.

Collective Learning Robots connected to the cloud can exchange instantaneously information, trajectories, and any kind of data in general.

In the case we are treating, the map generation, path planning and validation are performed in the cloud-based traffic manager environment, that, then, based on the waypoints of the chosen trajectory, uses the cloud to communicate with the UAV's control

system and, as a consequence, with the actuators. Cloud communication allows the traffic manager to receive back data from the sensors on board of the UAV to monitor flight parameters and to receive other kinds of information (Figure 7.5). This aspect is particularly interesting for future developments, since in the smart city model, UAVs can became basically fleets of sensors in the sky, able to perform a large variety of tasks, from traffic management to pollution and environmental hazard monitoring, to security services.
In our specific case cloud can be very useful for a possible implementation that exploits the on board camera, eliminating, for example, the need of satellite images and yet having real time pictures. More in general the cloud allows the connection to the internet that can be used to upgrade a static risk map to a dynamic one. As a matter of fact, through the web it is possible, in principle, for the traffic manager to become aware of events that can gather large groups of people in a certain area, raising the risk level there. It could access to the exact number of people, if available, to make a precise evaluation of the risk, or, even, collect data on the number of phone signals connected to a specific antenna to estimate the number of people in that area.
In general the cloud architecture can lead to a huge number of opportunities of implementation but the most important feature is certainly the possibility, in the future, of completely excluding humans from the control loop, leaving the cloud-based traffic manager to handle every aspect of UAV flights, from planning to landing, guaranteeing, at the same time, an high level of safety for people.

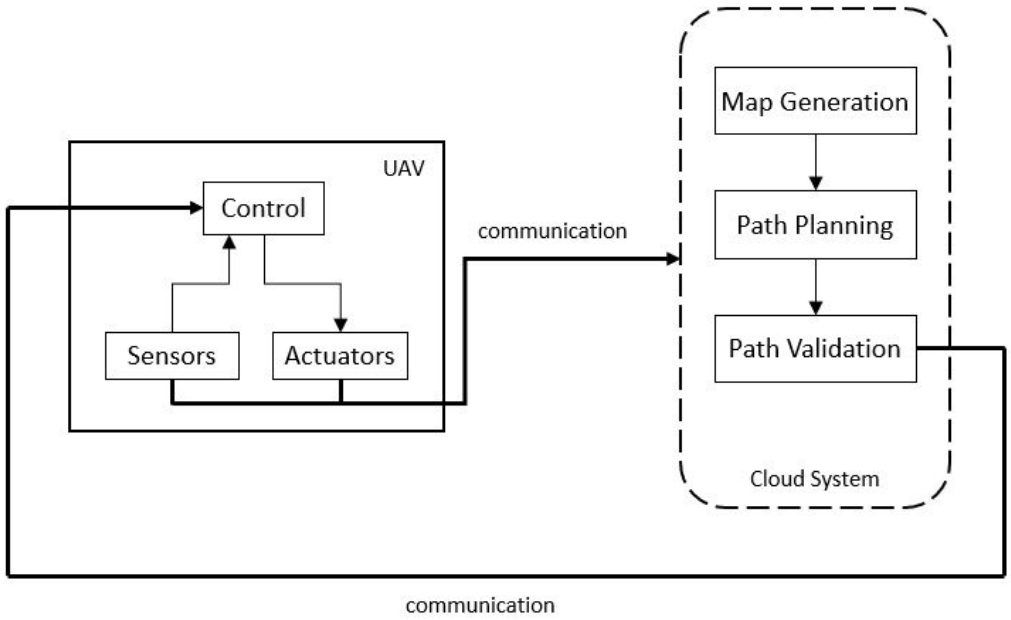


Figure 7.5: Scheme of the communication between UAVs and CBUTM

Chapter 8

Conclusions and future developments

The importance of guaranteeing safety in flights of UAVs has been widely discussed in this thesis, as well as the need of unified and coherent risk assessment techniques and regulations. In this sense, this work provides a complete picture of the state of the art of the risk analysis for unmanned missions.
The method we proposed for risk assessment is not focused on the specific UAV itself. Such an approach would require a study of the aircraft in deep, starting from its smallest components and we have assumed that it is carried out by the manufacturers. Instead we have based risk assessment on the geographical area, evaluating the risk for people to be direct or indirectly affected by the consequences of a possible accident.
We have focused our attention on the techniques to mitigate risk and in particular on the definition of sheltering factor, whose state of art has been presented before introducing an new scale to evaluate it.
Some remarkable results have been obtained by using for the first time machine learning to automate the definition on the sheltering factor starting from a satellite picture. In fact we have been able to create an algorithm that performs image analysis, extracting features like colours and edges and elaborates them using a two-layer feed-forward neural network, previously trained by supervised learning, to obtain the corresponding sheltering factors. This has allowed to define a sheltering map, a particular risk map that evaluates the shielding ability of the objects and structures present in the original satellite picture. We have explained how such a map can be used to perform path planning in order to find, among all the possible ones, the flight trajectory that present the lowest risk for humans and we have cleared how the risk for the overall mission can be computed and compared to the regulations given by national flight authorities to validate or deny the mission.
The importance of the automation of the whole process is crucial to be able to assure safety in a future in which mission will have to be accomplished in urban environments by fully autonomous UAVs.
However some limits have showed up in using the colour analysis approach, since, in some cases, the algorithm has failed in recognizing areas with similar colour but different sheltering ability. This opens the way for further future developments and we really hope

that this project can be continued in the future. The first idea to improve the results is to explore satellite images to extract some other features in addition to colours and edges, in order to produce a more precise neural network fitting function. An alternative could be to create an algorithm that cross checks and embodies these results with the ones obtained on tridimensional maps, like to ones of the algorithm developed by [27]: the union of the two would probably lead to a very accurate Map Manager. Finally, another idea is to compare the results obtained in this work with deep learning algorithm specifically trained on satellite pictures, to recognise buildings and objects. It would be definitely an hard task to develop, but merging object recognition with colour and edge analysis, would probably produce very precise results.
As a conclusion, we hope that, in the future, technological developments and new regulations will allow UAVs to guarantee safety of people in any situation and to exploit the possibilities they offer on the way to a more sustainable and efficient smart city model.

Bibliography

[1] Al-Kaff A., García F., Martín D., De La Escalera A., Armingol J.M. (2017), *Obstacle Detection and Avoidance System Based on Monocular Camera and Size Expansion Algorithm for UAVs*

[2] AUVSI (2013) *Economic Report*

[3] Berry M.J.A., Linoff G. (1997) *Data Mining Techniques*

[4] Blum A. (1992) *Neural Networks in C++*

[5] Canny J. (1986) *A Computational Approach to Edge Detection*

[6] Centre for Telecommunications and Information Engineering, Monash University (2016) *Remote Piloted Aerial Vehicles: An Anthology*

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

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

[9] Dalamagkidis K. ,Valavanis K. ,Piegl L.A. (2008),*Evaluating the Risk of Unmanned Aircraft Ground Impacts*

[10] Dotti G. (2017) *Smart city, come saranno le nostre città intelligenti nel 2032?* Wired Magazine

[11] Elisseeff A. , Paugam-Moisy H. (1997) *Size of multilayer networks for exact learning: analytic approach*

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

[13] European Aviation Safety Agency (2009), *Airworthiness certification of Unmanned Aircraft Systems (UAS). Policy statement, E.Y01301*

[14] European Aviation Safety Agency (2007), *Certification specification 25 (CS25). Amendment 3*

[15] FAA (2000), *Expected Casualty Calculations for Commercial Space Launch and Reentry Missions*, FAA-AC431.35.

[16] FAA (2016) *Model Aircraft Operating Standards. Advisory Circular No. 91-57A*

[17] Giammusso S., *Cloud Robotics in real time application*

[18] Gilli M. (2017) *Torino diventi la prima "Smart City"*, La Stampa

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

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

[21] Haber J.M., Linn A.M. (2005) *Practical models of human vulnerability to impacting debris*

[22] Haykin S. (1999) *Neural Networks: A Comprehensive Foundation. (II edition)*

[23] Joshi D. (2017) *Exploring the latest drone technology for commercial, industrial and military drone uses*, Business Insider

[24] Joshi D. (2017) *Commercial Unmanned Aerial Vehicle (UAV) Market Analysis – Industry trends, companies and what you should know*, Business Insider

[25] Kehoe B., Patil S., Abbeel P., Goldburg K. (2014), *A Survey of Research on Cloud Robotics and Automation*, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.

[26] Lawrence S., Giles C.L. , Tsoi A.C. (1996) *What size neural network gives optimal generalization? Convergence properties of backpropagation. Technical Report UMIACS-TR-96-22 and CS-TR-3617*

[27] Lorenzini A. (2018) *Cloud Based UASs Traffic Management: a Risk-Aware Map Manager*

[28] Mohammed F., Idries A., Mohamed N., Al-Jaroodi J., Jawhar I. (2014) *UAVs for smart cities: Opportunities and challenges*

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

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

[31] NCAR / EOL Workshop (2017) *Unmanned Aircraft Systems for Atmospheric Research*

[32] Neades D.N., Rudolph R.R. (1984) *Head injury prediction capability of the hic, hip, simon and ulp criteria. Accident Analysis and Prevention*

[33] Otto A., Agatz N., Campbell J., Golden B., Pesch E. (2018) *Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey*

[34] Polia F. (2018) *Cloud-Based UASs Traffic Management: Trajectory Tracking and Collision Avoidance*

[35] Ross S.M. (2015), *Probabilità e Statistica per l'ingegneria e le scienze*

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

[37] Range Safety Group, Range Commanders Council (2001), *Range safety criteria for UAV: Rational and Methodology Supplement. Supplement to document 323-99*

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

[39] Schoellig A. (2014) *The Role of Unmanned Aerial Vehicles in Future Urban Environments*

[40] Stabile E. (2018) *Cloud-Based UASs Traffic Management:General Architecture*

[41] Trippi R.R. , Turbam E. (1992) *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance*

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

[43] Valavanis K.P. , Vachtsevanos G.J. (2015) *Handbook of Unmanned Aerial Vehicles*

[44] Walker S.H., Duncan D.B. (1967) *Estimation of the probability of an event as a function of several independent variables*

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

[46] Weibel, Hansman (2004), *Safety considerations for operation of small UAV in civil airspace*, Master Thesis, Massachusetts Institute of Technology

[47] Winkler C. (2016) *Sensors insights*