December 21, 2021

Cost-based path planning software for planetary rover on uncertain terrains

---

Master of Science in Space Engineering

POLITECNICO
DI MILANO
DIPARTIMENTO DI SCIENZE E
TECNOLOGIE AEROSPAZIALI (DAER)

Telespazio Belgium

Politecnico Di Milano

Department of Aerospace Engineering (DAER)

Author: Tisi Nicola

Relator: Professor Mauro Massari

Editors: Ing. Nicola De Quattro, Ing. Filippo Iodice

Dedidcato a Sara
La mia Stella, il mio Riferimento

ABSTRACT
The exploration of the unknown excites the human race since ages, and in this last 70 years this has been possible due to the improvements of the technologies in the space and aerospace sectors. The humans have been able to reach personally the Low Earth Orbit and the Moon, with a huge effort made through the decades.
The technologies and environmental limitations have prevented the men to step on other planets, by now, but not to reach it with rovers and unmanned devices.
Any planet in which humans have sent rovers or probes is far enough to be impossible to be reached by a signal able to control actively a device and its motion, and for this reason some softwares have been created in the last decades to control the autonomous movement of the unmanned devices moving through an hostile environment.
For this reason, the aim of this Master thesis is to write a code able to guide a rover in a unknown environment, just using the DEM data obtained by a satellite and eventually a couple of stereocameras for a precise optimization in a local range, even considering the low level of computational power typical of the space-related processors.
The software is realized with a cost-based optimization, and its specifically designed to require a low level of computational effort.

Websites:
https://www.telespazio.com/
https://www.polimi.it/

ABSTRACT
L'esplorazione dell'ignoto entusiasma gli esseri umani da secoli, e negli ultimi 70 anni questo e' stato reso possibile da un grande miglioramento tecnologico in ambito spaziale e aerospaziale. Gli esseri umani sono stati in grado di raggiungere personalmente l'orbita bassa terrestre e la superfice della Luna, con un grandissimo sforzo fatto nel corso dei decenni.
La tecnologia e le condizioni avverse sono state limitazioni che hanno impedito all'uomo di sbarcare su altri pianeti, per ora, ma siamo comunque stati in grado di raggiungerli con rover e missioni senza equipaggio.
Ogni pianeta su cui gli umani hanno mandato dei rover e' sufficientemente lontano da rendere impossibile ad un segnale radio di raggiungerlo in tempi brevi per controllare attivamente il rover stesso e fornire indicazioni riguardo al suo moto, e per questo motivo negli ultimi decenni sono stati creati dei software in grado di controllare autonomamente i movimenti dei rover e dei dispositivi robotici che tutt'ora si muovono in ambienti ostili sulla superficie di altri pianeti.
Per questo motivo, l'obiettivo di questa tesi magistrale e' di scrivere un software in grado di guidare un rover in un ambiente sconosciuto, solamente usando i dati di un DEM ottenuto da satellite ed eventualmente una coppia di stereocamere per un'ottimizzazione precisa del percorso su una scala locale, il tutto tenendo conto delle limitazioni dal punto di vista computazionale a cui sono sottoposti i processori ad uso spaziale.
Il software e' stato realizzato con un sistema di ottimizzazione basato sulle funzioni di costo, ed e' specificatamente pensato per funzionare in sistemi dotati di una ridotta capacita' di calcolo computazionale.

Websites:
https://www.telespazio.com/
https://www.polimi.it/

Nomenclature

CCD	Charged-Couple Device
DEM	Digital Elevation Model
DSM	Digital Surface Model
DTM	Digital Terrain Model
FPA	Flower Pollination Algorithm
GA	Genetic Algorithm
GNC	Guidance Navigation and Control
KERS	Kinetic Energy Recovery System
LQR	Linear Quadratic Regulator
LVLH	Local Vertical, Local Horizontal
MDP	Markov Decision Process
NASA	National Aeronautic and Space Administration
OBDH	On Board Data Handling
OV	Horizontal/Vertical
RAM	Random Access Memory
TIFF	Tagged Image File Format

List of Figures

Contents

Part I
State of the art

1 Introduction

With the increasing number of exploring space missions on other planets of the Solar System, it has been necessary the develop of automatized rover to explore remote zones hostile for humans, like the surface of a planet with a low level of livability for human being.

The recent Mars exploration missions reminds us again about how the in-situ analysis of the surface of a planet with instruments and sensors must be carried out relying on devices remotely controlled, able to perform difficult activities like collect data or physical samples, movements, active thermal control, power control and distribution and so on.

All these operations could be eventually done remotely by humans, but this could be a huge disadvantage due to delay of synchronization between rover and the control engineers. In fact, this method could have a minimum of reliability only for Moon operations, since the distance Moon-Earth can be covered by light in about 1.3 sec, but for all the other missions the delay can be really relevant: e.g. the distance Earth-Mars can vary between 0.5 and 1.5 AU[1], and it can be covered by light in a time that goes from 4 to 12 mins. This means that every command given by the ground control would have such a big time delay that would make it useless for the rover.

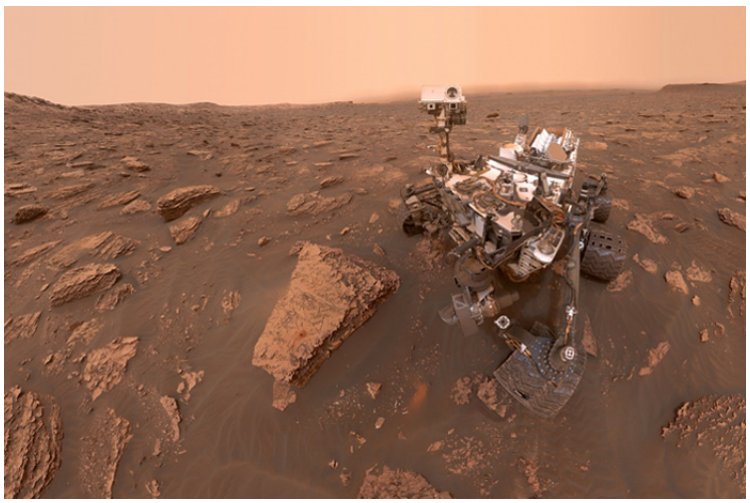

Figure 1: View of the rover Perseverance on Martian soil

[1] AU = Astronomical Units, mean distance Earth-Sun, approx. $1.5 \cdot 10^{11}$ m

The only reasonable alternative is a rover that has a system capable of performing autonomous operations in almost all the important field in its life, in particular the motion between different space points on the surface of the planet.

This last task has a big relevance in the lifespan of a planetary mission, since it takes a lot of time and power to be executed.

Moreover, the motion of a rover is enormously subjected to the unknown elements that can be present in an hostile environment like Moon, Mars or Venus' surface (et similia). The different kind of surfaces, the slippery of the materials, the ratio of penetration of the wheels in the soil, the presence of obstacles of various kind, are all important parameters that can discriminate not only the cost of the mission (in term of time or power needed), but also the eventual success of the mission.

The necessity of this part of the thesis is bounded to have a clear vision of is the actual state of the art of the guidance softwares and methods used to face this kind of problems. All this informations has been used to optimize the software and to make it more robust with respect to any variation or unexpected event in a real utilization, but at the same time to invent something new based on what already exist.

This initial part of the thesis has been done to analyze the state of the art about the rover autonomous guidance systems, using literature from multiple different sources, with the aim to analyze the method and algorithms that can solve this problem in the most efficient way.

Hereby, have been evaluated the most important aspects for a guidance system, starting from the environment in which it should work, some hints about the hardware that runs it, and the discriminations that a commercial software usually does to discretize the problem.

2 Environment

Between all the reports analyzed, some of them just tried to optimize the path (in terms of spatial distance), just considering the overview of the zone of motion, giving the obstacle a boolean value of cost (i.e. in case of a small rock, it can be seen as a path with a cost of zero or infinite, depending on the size of the external body).

This kind of approach is useful to reduce the complexity of the kinematics of the problem, but it minimize the real effort of a route planning software, since it does not consider the terrain features and the real power needed by the rover.

For this reason, it is important to have a clear idea of the characteristics of the terrain on which the rover is moving. Of course, the terrain trafficability analysis must be carried out individually for each soil (on each planet), but the main goal is to consider a generic a Mars-like or lunar-like environment: both kind of terrains have many features in common, like the presence of sand, pointy rocks distributed unevenly on the surface, craters with different dimensions, and a low level of moisture in the ground. While designing a path planning software it is important to consider the effect of the different kind of materials on the wheels, to evaluate the most convenient route.

2.1 Martian/lunar soil features

In the following paragraph are analyzed the main terrain types encountered during the martian/lunar missions. These are the main five:

1. Sand: this kind of soil is composed by thin elements that does not provide any type of damage to the structure if overcrossed, although the sand can be an hard obstacle to pass due to its low level of mechanical grip. In fact it has been considered the least dangerous type of terrain for the rover Curiosity, and even testing proved that the pressure of the sand on the wheels was not great enough to cause crack growth on the surface of them. On the other hand, for MER-class rovers, sand can be the greatest danger, since even Spirit rover ended its mission because it was immobilized by sand.
2. Bedrock: solid rock underlying loose deposits, this kind of soil can guarantee good mechanical properties and grip.
3. Loose rock: when sitting on bedrock, the loose rock cannot be pushed into the ground, which can induce wheel damage, depending on the size and geometry of the rock. The hardness of the bedrock can also induce stress concentration cracking at the structure.

4. Embedded pointy rock: This kind of soil is made of hard rocks that cannot be moved since stucked into the ground. It can be dangerous to move on them since the vincular reaction of weight can be exerted on the small surface of the tip of the rock, generating an enormous pressure. This could damage the structure of the wheel, but also stuck the wheel on the ground. It is the primary terrain encountered during the period of highest damage generation on Curiositys wheels.
5. Embedded round rock: Less hazardous with respect to the previous one, since apply a lower pressure on the surface of the wheels, but the fact that they cannot move on the ground makes them still dangerous to pass over.

To design a software able to decide the path of a rover in a local-scale, the analysis of the kind of soil encountered is foundamental. For a global-scale, otherwise, the composition of the local soil can be neglected.

In the software design each kind of obstacle or terrain can be evaluated with an associated cost function.

The utilization of the path planning software on martian soil is the final goal, but in order to try its capabilities it shall be necessary to try the software on a more known terrain, so basically on Earth. Let's see the other alternatives:

2.2 Polar region soil features

As said at the beginning, the basic idea behind this paper is to analyze the characteristics and performance of a guidance software for an autonomous rover on unknown terrain, without real-time human commands. This model of work can be useful while working on another planet like Mars, but also while facing impervious remote zones right here on the Earth, just like the polar caps.

The icy zones of the earth are a good alternative to design a software of this kind, for many reasons: same uneven and difficult terrain, no presence of human help or direct control, extreme environmental condition, but mostly the presence of a huge amount of data to work on (e.g. DEMs, as seen in chapter 3.2) to test the funtionality of the software.

The polar landscape, however, has different characteristics with respect to martian or lunar terrain in terms of both mechanical properties of the soil and kind of obstacles.

Due to the low value of friction coefficient, the level of mechanical grip can decrease significantly while moving a metal wheel on the snow/ice, but this create a difference with

respect to martian case only for the dynamic equation and the power distribution, not for the kinematic approach used to derive a path planning software.

Moreover, the polar regions present another difference with respect to the rocky soil due to the very low density of small obstacles to be avoided, and the lower effect they have on the structure if overcrossed.

This kind of terrain is very useful to test the autonomous path planning software on a macro-scale.

2.3 Terrestrial deserts soil features

Another option that can be considered intermediate between the previous ones is the availment of a rocky desertic soil similar to the one of Mars, but on Earth. Many different rocky deserts have been already used to design and test everything connected to space exploration, from farming to simulation of life in space, passing through the test of rovers (both for software and hardware).

The mechanical characteristics are totally the same of Mars or the Moon, except for the lower level of radiaton absorbed by the hardware and the different intensity of gravity.

This kind of soil is perfect to test the dynamic equation of motion and the software for power distribution or the local path planning software.

Figure 2: View of a polar environment and a rocky desert soil

3 Hardware

As a premise, let's say that this study is based on the software part of the autonomous navigation system, and not the hardware.

But said this, it is important to keep in mind the hardware work environment to keep this research as realistic and concrete as possibile, and to have a clear idea of what type of instruments could be used in a system of this kind.

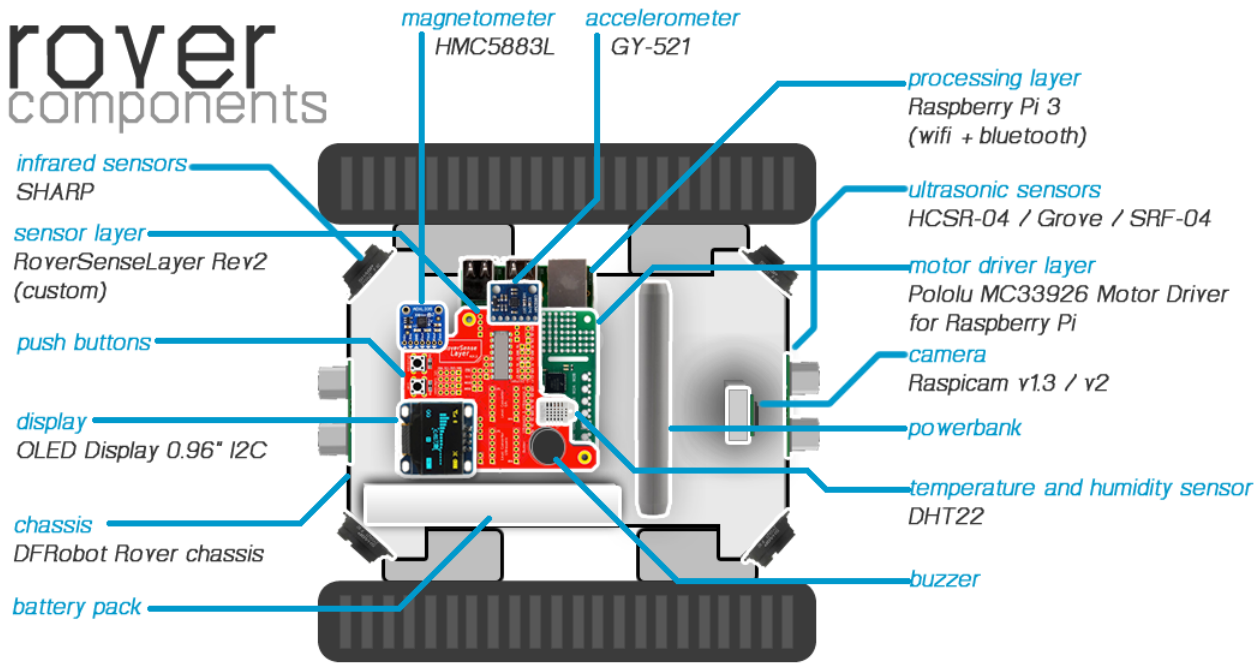

Figure 3: Example of the hardware of a small amatorial rover

3.1 On-board data acquisition

There are many different devices that can be used to obtain informations on the motion constraints, like obstacles or terrain properties.

Almost all of them are *passive* sensors, since they do not interact with the environment around, with the exception made by the group of Radar/Lidar/Sonar which otherwise measure data that are not available in the environment.

In particular, the main ones are:

- Stereo cameras: this is a type of camera with two (or more) lenses able to obtain images with a separate sensors for each lens (usually classic CCD[2] sensors). This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images. Stereo cameras may be used for making analysis on the distance of the obstacles while deciding the path to be followed. Strongly used for local path planning.

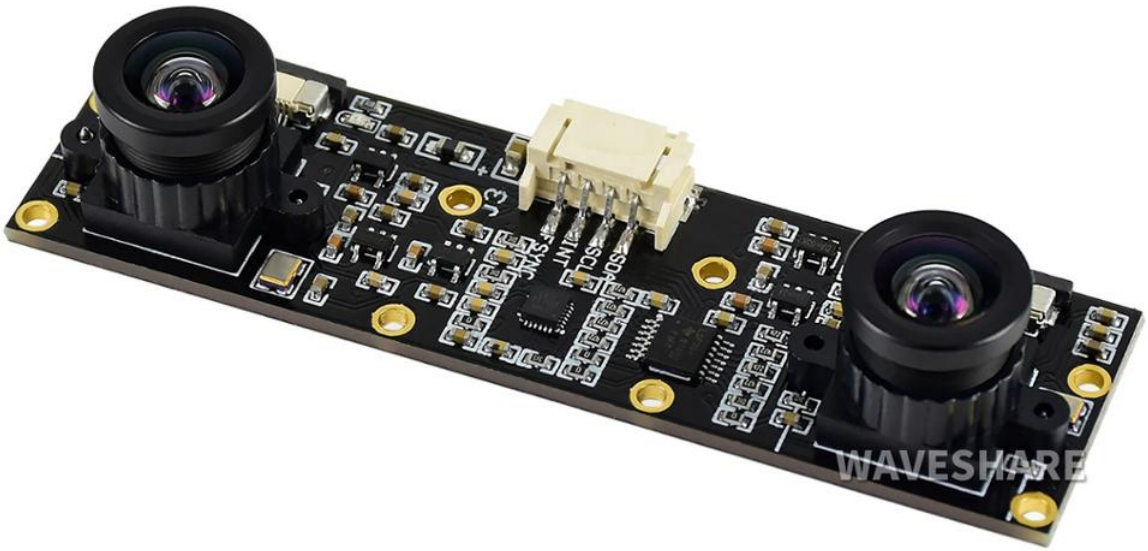

Figure 4: Example of stereo cameras

- Accelerometer: devices able to detect and quantify the value of instantaneous acceleration; are often used for inertial navigation systems for aircraft and missiles. Regarding a GNC[3] system for a rover, an accelerometer can be useful in case of evaluation of the dynamics of the system, not the kinematics.
- Infrared Sensors: this is a device that is used to sense certain characteristics of its surroundings relying basically on temperature and heat emission. It does this by both emitting or detecting infrared radiation. Since the devices are based on electromagnetic waves with a lower frequency with respect to the visible light, they are able to "see" the light emitted at low temperature. Very used in rover navigation.

[2]Charged-Couple Device, typical device used to capture the image in a camera
[3]Guidance, Navigation & Control

- Gyroscope: is an instrument, consisting of a inertial wheel mounted into two gimbals providing pivoted supports, to allow the wheel to rotate about a single axis. Eventually can be used a set of 2 or 3 axis to measure the value of angular velocity in 2 or 3 independent directions. They measure the value of angular velocity, so regarding rover motions there are a poor number of applications.
- Radar/Lidar/Sonar: obtain information about the outer world by sending a signal into the environment and then observing how information from that signal propagates back to the sensor. This makes them *active* sensors. Are really useful for the navigation software, both for a local and global scale.
- Attitude/positioning sensors: there are many kind of sensors able to achieve a complete determination of the position/attitude, for example sun sensors, star trackers, magnetometers, et similia. Their use is often connected to satellites, but they can also be useful for the rovers, expecially if the rover recieves data in real time from a satellite.

3.2 External data acquisition.

In this chapter will be presented something different than an hardware, but they are still "external instruments" that are necessary to face the design of the software.

A digital elevation model (DEM) is a 3D graphic representation of elevation data to represent terrain, commonly used for a planet. Also a DSM (Digital Surface Model) can be used to model the surrounding landscape and build a digital rapresentation of the areas intrested by motion.

A powerful technique to generate digital elevation models is interferometric synthetic aperture radar, obtained by two passes of a radar satellite[4] that collect sufficient data to generate a DEM with the size of tens of kilometers and with a resolution of around ten meters.

As described in chapter 4.2.2, the use of DEMs is made necessary to obtain a global overview of the area that could be possibly covered by the rover while moving, and to set up correctly a general path.

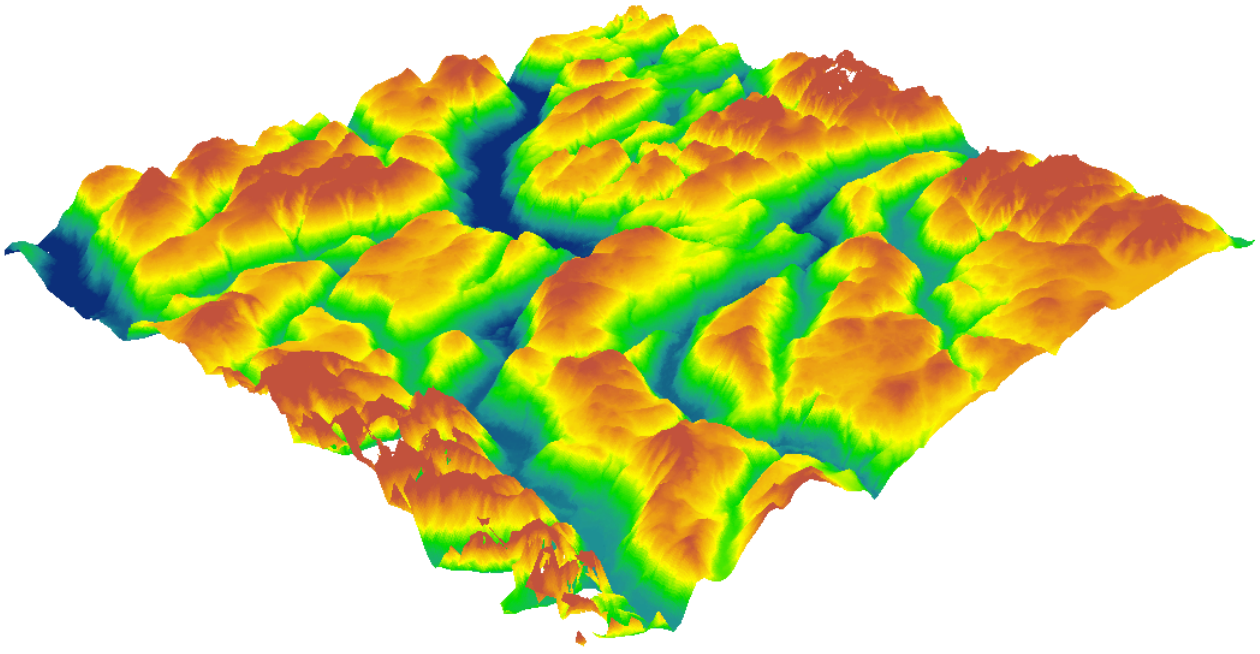

Figure 5: Example of DEM

[4]such as RADARSAT-1 or TerraSAR-X or Cosmo SkyMed

4 Software

4.1 Path discretization tecniques

One of the most important parameters to be determined while creating a software is the kind of discretization. This decision is mostly regarding space, since time is always discretized linearly with an accuracy that depends on the resolution of the track and the computational cost.

For what concern the route, most of the existing software present 4 main kind of space division, depending on the software architecture and motion accuracy

These four kind are the following:

- Square-shaped discretization (cartesian): this kind of discretization is the most simple and common. It has good features like the possibility to analyze the space as a cartesian plot, assigning each square a precise value of content (from the classic boolean one or more complex ones that can interact with the cost funcionals). It has great computational efficiency, since the total amount of routes to be analized is a finite number, and is often limited by the presence of macro-obstacles. However, this kind of discretization can limit the possibility of move in case of a complex terrain[5]. Anyway, cartesian point of view can be used to analyze the path in a compact way and with a focus on the cost of each step.

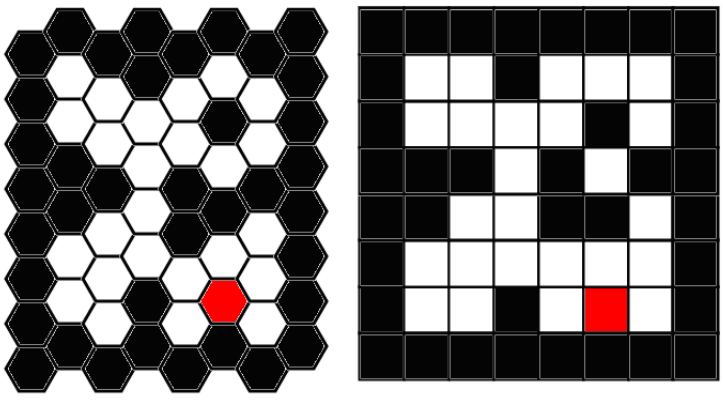

Figure 6: Path discretization with exagonal and cartesian approach

[5]E.g. cartesian discretization could be limiting in a situation where two rocks can be identifyed as near limiting obstacles, even if the wheel of the rover could pass between them without any increase of cost.

- Angular discretization (polar): this kind is the less used, it is based on the division of the space in 360 degrees. With respect to the previous case, this is more useful for a movement in a reduced range of space, but cannot be used easily in a global-scale movement.
- Exagonal discretization: quite similar to the first case, but with an exagonal pattern. This gives more motion options, but it is a little bit more complicated from the software point of view.
- No discretization (free movement): the motion of the rover is not bounded to a precise pattern created to analyze the ground, although it is based on the "raw" geometry of the ground and its obstacles. To give an example, in case of a surface with a rock (discretization step: 0.1 meters, diameters of the rock: 0.1 meters), the cartesian software would avoid a zone that has a surface up to 4 square (4 dm^2), while the absence of discretization allow to neglect a surface[6] of 0.8 dm^2. Obviously, the absence of discretization is the most advanced and difficult to develop, with the highest computational cost but also the best result in term of power consumption for the motion.

[6]Plus a safety margin

4.1.1 Boundary motion conditions

The path determination is basically a kinematic problem in which the equation of displacement must be created relying on external data (and also on dynamic equations). For this sake it is important to remember that, as all the kinematic equations, also this kind of problem necessitate to boundary conditions to have precise extreme of motions.

In the existing softwares, there are three main kind of approach to achieve this determination:

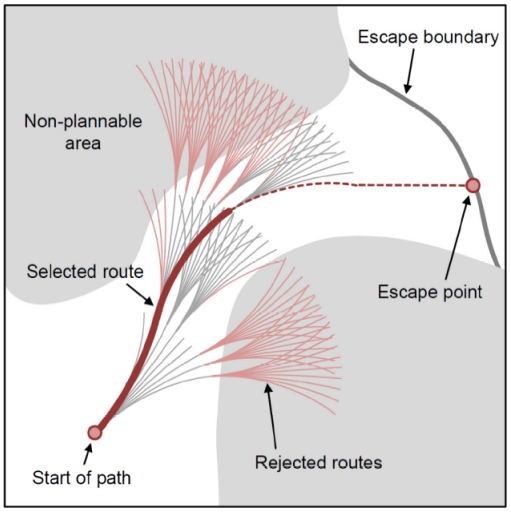

Figure 7: Concept of "escape boundary". Taken from reference [1]

- Fixed boundary (end to end): this kind of condition is absolutely the easiest one: there is the need to go from a well-determined point A towards another point B. Even if there are many ways to complete this task (actually infinite), the problem is still well determined and easy to be solved. This is due to the fact that in a system with an high value of degrees of freedom, every constraint can reduce the time of solving and the computational cost. On the other hand, this may not found the best solution for the motion.
- Fixed start, multiple arrival: similarly to the previous case, here is still considered a motion that starts and arrives in precise points, even if in this case the choices for the goal is still undetermined[7]. Straightforwardly, this implies that the time needed to find

[7] e.g. this kind of boundary constraints is exactly like the funcion implementated in the common mobile apps that are used to find train hours to connect a precise starting point with a generic "all the stations" of a city

the possible paths is increased to evaluated every single options. Is still a good choice if the last point is not a compulsory passage.

- Escape boundary: this kind of constraints is the most “open”, since is based on a “finish line”concept, like a marathon, rahter than a precise point. Of course this implies that the path is way more customizable, with the possibility of chosing many different roads rather a single optimal one. chosing an end that is not determinated a priori can guarantee an high level of robustness to the system in case of variation/alteration of the condition of motion. E.g. let’s immagine to determine a precise path, and to discover at the half of it that it has been changed by external factors or by mistake in the data relevation: in this case the flexibility of the system can be crucial. On the other hand, this system is way more complex to be designed and requires a huge computational cost. Moreover the non-uniqueness of the solution may be a problem while implementing boundary conditions.

4.2 Motion constraints

The software that must guide the rover in a unknown terrain shall take decision relying on data acquired by sensors. These data are then elaborated to allow the software to associate a cost function to all the obstacles, and then evaluated the cheapest route.

To face this aspect of the software, it is necessary to distinguish two different kind of motion: the local one (in which there is the need to avoid single small obstacles) and the global one (in which is decided the average path to be followed to go from the start to the end).

4.2.1 Local motion constraints

This kind of obstacles are divided into discrete obstacles (like a rock) or distributed one (like sandbars). Each obstacles must be associated to a cost function to be elaborated by the software.

The kind of obstcles has been already elencated in chapter 2.1, and can be summarized depending on their main features.

The software can be tuned in such a way to recognise some excluded areas, both deleting them from the available zones or also giving them an high cost (similar to infinite), to make not convenient any choice that include those points in the path.

This motion constraints must be evaluated locally to have the most precise informations about the nature of the obstacles. This work is done mostly by stereocameras and infrared sensors.

4.2.2 Global motion constraints

If the route of the rover would be long enough, a local evaluation of the path could not be efficient. A software based on the logic "local optimum implies global optimum[8]" often does not provide the best solution since does not take into account the effect that an initial decision can have on the final decisions. For this sake, it is also necessary to complete the software with a global overview of the path, to be sure that every single step taken is a part of a bigger route finalized in reaching the final point (or the escape boundary).

To achieve this level of motion determination it is necessary to have a perfect overview of all the surroundings of the motion zone, and this can be done almost exclusively from

[8]Mostly known as "Divide et Impera"

above (expecially in case of terrains with frequent slope variation, since in that case the on-board instruments would have limited visibility). A satellite recognition can be achieved "quite easily" specially on another planet. With a single orographic map of the zone it is possible to establish a general track to be followed relying on the the presence of the macro-obstacles or compulsory paths. This kind of data are called DEM[9] or DTM[10] and can easily be obtained with a resolution of few meters.

4.3 Attitude determination

A generic software could work, as said previously, with both local and global informations to manage the best solution. This implies that the data obtained must be coincident in terms of time and space. There is the need of a single reference frame to be used, or otherwhise 2 different reference frames but a numerical relation between them (e.g. a LVLH[11] referernce frame for the rover and a classic topocentric[12] one for the DEMs).

Here are the main kind of reference frames that can be used to analyze the position and the attitude of a rover:

- LVLH: local vertical, local horizontal. This kind of reference frame is centered in the object and gives the value of position with respect to it. If associated with a local measurement of time, it can also be called "hic et nunc", or "here and now" reference frame. It is very useful to describe the motion of the rover, but at the same time it is hard to be used in case of other measurements in a different reference frame.
- Topocentric: this kind of spacial frame is commonly used to determine the position of a device on the surface of a planet. it is based on the measure of two different angles: the Right Ascension (also called α), and the Declination (indicated with δ). The first one is used to indicate the longitude with a value between $0\,\mathrm{rad}$ and $2\pi\,\mathrm{rad}$. The second one is used for the latitude, with a value between $-\pi/2\,\mathrm{rad}$ and $\pi/2\,\mathrm{rad}$, centered on the equator line. If coupled with the distance from the center of the planet, it can generate a spheric polar coordinate system. With respect to LVLH, it has a global validity, but can be difficult to be used locally.

[9] DEM: Digital Elevation Model
[10] DTM: Digital Terrain Model
[11] Local Vertical, Local Horizontal
[12] Based on Right Ascension and Declination

4.4 Computational cost

In many occasion, the path planning algorithm has been found as a pure operational research problem, with the only purpose to optimize the path in terms of power, time and space.

This is done in many type of software without any limit to the computational power, since many software requires few seconds to be runned on a standard computer with 16Gb of RAM memory.

Unfortunately, in real world the rover used have often a huge difference in terms of computational resources, in particular the one that have been used for a planetary mission on Mars.

Just to give some numbers as example: the ultimate NASA's rover Perseverance, arrived on Mars on 2021, has a processor with a clock of 200 MHz, meanwhile a normal laptop can reach a value of 3/4 GHz. Regarding the memory, Perseverance has 2Gb of flash memory, and 256Mb as RAM, with respect to 1Tb and 16Gb for a commercial computer. Those values, for the previous rovers Spirit and Opportunity, were even 10 times lower.

This limitation is basically given by environmental restriction due to radiation exposure, and the quick advance of this technology that cannot keep the pace of the TRL scale[13].

To be able to use a software on a martian rover, it is necessary for it to be as light as possible or, even better, that a part of it could be runned by another external computer.

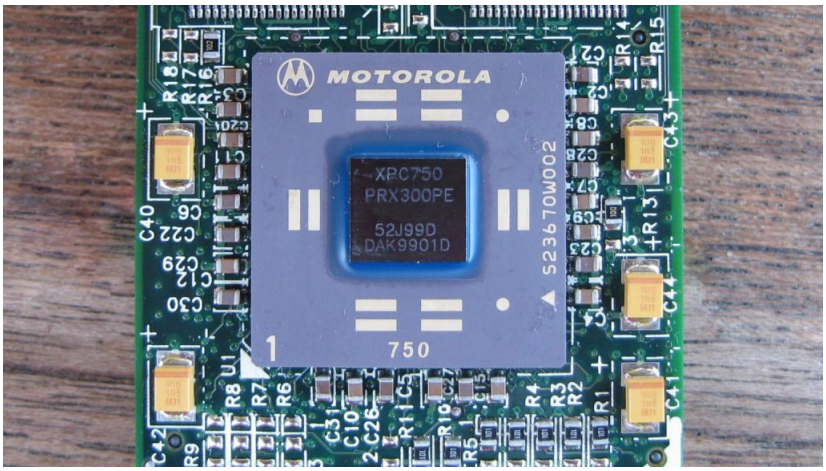

Figure 8: Processor used for Perseverance, taken from a Mac dated 1997

[13] Technology Readiness Level

4.5 Machine learning

Machine learning involve teaching a system to perform a task and/or finding the optimal solution to a complex problem.

This kind of coding does not involve simple algorithm utilization, or straight relations between input and output, otherwise it is based on a more complex elaboration of a set of data (often uncomplete), in order to obtain an output anyway.

There are many kind of machine learning, here are the main ones:

- Supervised Learning: algorithms based on this approach teach a system to discriminate between various groups or classes of input, relying on examples of each class. This can be done only after using a certain amount of examples carried out by the user, after which the system is able to manage data that have not been presented before.
- Reinforcement Learning: teach a system appropriate actions based on the concepts of reward and punishment. It rely on "Markov Decision Process" (MDP), which is a nondeterministic graph representations of the events: given a certain input, the output event may or may not happen. The software use this nondeterministic process to obtain a reward or a punishment.
- Neural Networks: this kind of machine learning is a network model inspired by the disposition of biological neurons in an human brain, so basically a set of nodes connected by edges. In practice, the function may output binary values rather than an intermediate non-boolean value. When activated, the node sends a signal along its outgoing edges, in order to activate the connected nodes.
- Adaptive Boosting: is a meta-algorithm that iteratively calls a seperate supervised learning algorithm. It works with an initial set of training values, each one associated with a weight depending on how much they influence the model. After each iteration, incorrectly classified examples are weighted relative more, while the correctly labeled examples are wighted relatively less. This causes the algorithm to focus its attention on the examples that are misclassified, and correct them.
- Q-learning: this kind of machine learning rely on the concept of delayed reward by accounting for future rewards that indirectly result from its previous actions. This allows the system to avoid locally optimal actions that are suboptimal globally, avoiding the strategy "local optimum implies global optimum". Because future rewards are uncertain, they are discounted by a coefficient γ for each time-step into the future that

they are expected to occur. The sum of discounted reward is also indicated with Q, and this gives the name at the method.

4.6 Already existing algorithms

As a basic element of all the software, it is necessary to identify an algorithm able to withstand all the processes and data, with the most accurate result in the least computational time.

All the possible algorithm analyzed are iterative, and their deploy must be associated to a study on their convergence capability and ratio.

The main kind of algorithms used for path planning are the following:

- LQR-based: this is an optimizing method based on the concept of "linear quadratic regulator", that assign a penalty to each step evaluated. These values of penalties are then summed to obtain the total cost of the path, often called J, and then the objective of the software is to minimize the value of J.
- Genetic algorithm (GA): this method rely on a population of candidate solutions (called individuals, or phenotypes) that must face an optimization problem and evolves toward better solutions. Every candidate solution is correlated to a set of properties (its chromosomes or genotype) which can be mutated and altered; usually, the solution is expressed with boolean values, even if other coding are possible.
- Flower Pollination Algorithm (FPA): based on the necessity of the plants to reproduce, this algorithm has been developed less than 10 years ago and has the aim to verify the pollination of a certain amount of flowers, both in local and global scale. It is based on the concept of self/cross pollination, and an iterative process to verify that the probability of a plant to be pollinated is above a reference valure "p".

These are the main algorithm, however different strategies can be used to achieve the same results, with different computational performance.

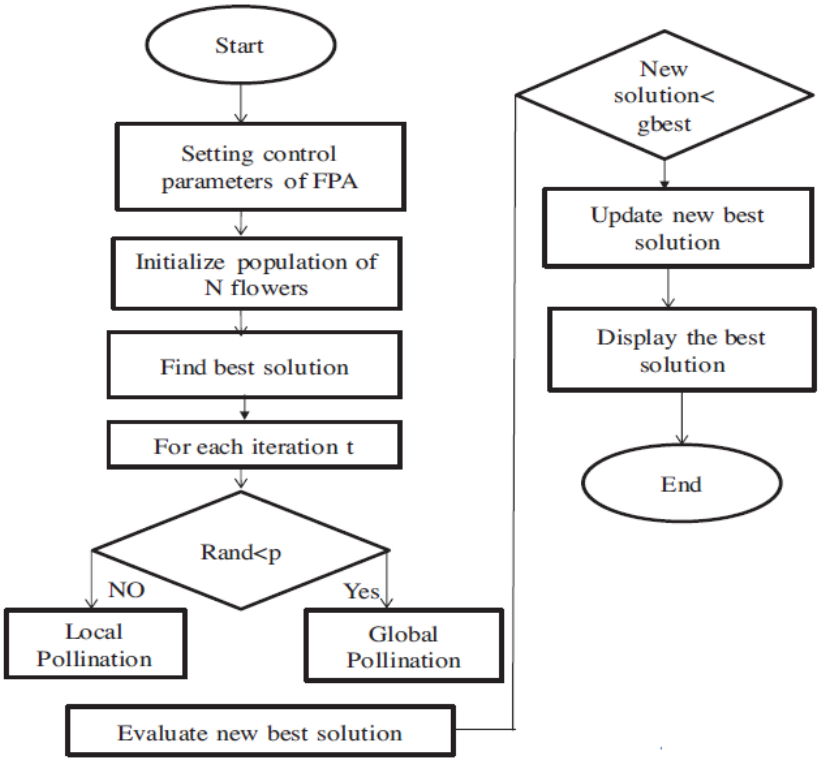

Figure 9: Flowchart of Pollination Algorithm, taken from reference [10]

4.6.1 Convergence of the algorithms

It is important to notice that all the algorithms must be convergent to be useful. The convergence is needed to obtain a final result that satisfy the boundary condition of the path, with a limited value of space crossed or power spent.

The convergence of the algorithms means that with an increase value of path elaborated, the best solution is supposed to be more convenient.

This procedure is iterative, but it does not have a monotonic convergence; this is basically the summary of the complexity of this problem, since if we would have a monotone converging method, it would be enough to keep iterating it to reach an asymptotic value.

Obviously, while developing the algorithm the software should be able to converge with the highest rate possible, to reduce the calculation cost of the process, expecially due to the low calculation capacity of a planetary rover.

5 State of the art and possible applications

This report has analyzed everything that concern the design and developement of a software for the path planning of a rover. There are already many kinds of software, for every kind of situation.

Relying on the necessity of a software with low computational cost, but an high value of convergence ratio, a good compromise can be a software that apply a strategy to concentrate the possible research of a path in a specific part of the field considered; changing the zone of interest while moving. This help to reduce the computational cost.

To be able to optimize the route to be found, the best solution is to use a penalty approach relying on the data obtained by Digital Elevation Model (DEM). These kind of data are really useful to solve this problem, because of their reduced complexity and compact informations regarding the terrain considered. Moreover DEM matrices can be implemented straight from a satellite relevation, increasing the performance of the software and the capability of autonomous work.

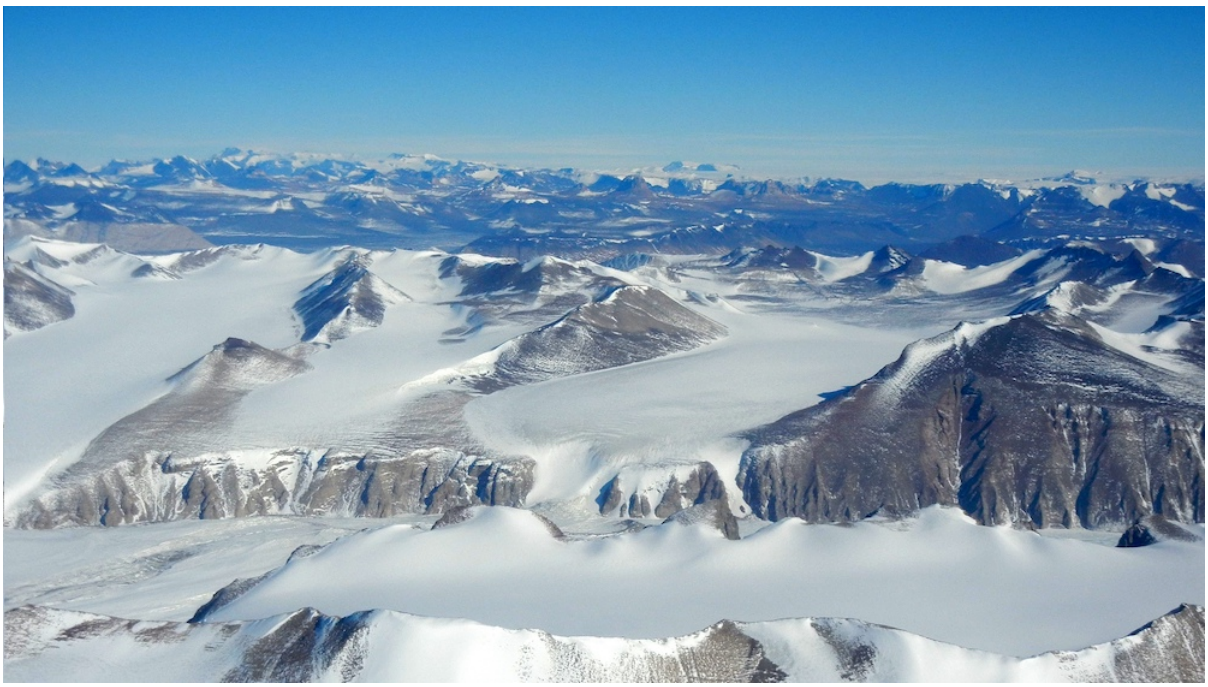

Figure 10: Example of environment useful for the application of the software

This path planning software will use a cartesian discretization of space, this method has

been chosen to reduce the complexity of the software and to match the discretization obtained from the existing DEM. It is also efficient to be used with two dimensional arrays.

Regarding the route optimization, it has been chosen a penalty-approach to find the best path, and reduce at minimum the energetic cost for the movements. This guarantee a good compromise between path accuracy and computational cost. This optimization is done both in a local and global scale, to ensure a good reliability even for long paths.

Moreover, the newborn software could be tested in an hardware that simulate the computational capability of a rover, to ensure the convergence capability even with reduced computational power. To rely on real data, a set of DEMs of terrestrial terrain can be used, instead of hypotize the orography of another planet.

In the following chapters will be analized the functioning and capabilities of this software, with a deep analysis regarding every single algorithm used and the different case studies evaluated.

Part II
Functioning of the software

6 Introduction

After having analized all the possible aspects for a guidance system, starting from the environment in which it should work, some hints about the hardware that runs it, and the discriminations that a commercial software usually does to discretize the problem; let's now consider the main software, written to obtain a path with both a global and local optimization, and a cost functioning that depends on many parameters like terrain orography, solar shadow, random obstacles and so on.

The main goal of this software is to obtain a path that link two different points in space, andand find a route that minimize the effort needed to run across it.

The expectation is to be able to run a rover in an hostile environment, like a rocky planet/moon as Mars or the Moon, in which a lots of human efforts has been concentrated in the last decades. Actually this software could be useful also to guide a rover in a known planet like our, but in a remote zone with a lover human assistance, and to prove this it has been carried out three different case studies.

The reason it has been chosen these three case studies is because all of them are different from the others, with unique features of the code and variations to make it fit better to the different necessities of each situation. For this reason it has been created a software that could work with any kind of DEM image, taken from any place of the Earth and beyond, but of course the different use of it will implies different kind of optimization rather than different boundary conditions.

All the case studies are runned on the Earth instead of another planet just for a simple couple of reasons: the disponibility of data and the possibility to verify better the output obtained.

This software has been realized using a Python3 programming code, with an Anaconda/JupiterLab distribution.

7 Software structure

This software has been realized with a precise separation between algorithms, in order to distinguish the logical function of each one.

This logical decomposition of the software is not only necessary to ensure the correct functioning of all the components, but also to allow a better correction in case of malfunctioning, and an easier way to implement any kind of improvements.

The software is articolated in different part, about a tenth of different functions, and a main code in which every single function is used.

In order to obtain a good path between a point A and a point B, it is necessary to have both a local and global optimization. These kind of optimization are complementary, often do not agree one with the other, but the road selected must take into account both of them, and the best solution could be nor the optimum in a local or global scale. This is given by the fact that the global optimization is made to optimize the whole path to find a road between two distant points, without considering any kind of small obstacle or little disadvantage, meanwhile the local optimization has the aim to follow the main path but with all the small corrections needed to avoid small rocks or insidious terrains.

To have a good evaluation of the path, it is necessary to have a cost function that keeps into account all the possible effect of the terrain on the rover, starting from the altitude variation that can be obtained from the DEM file, and adding any kind of optimization depending on the necessity (presence of spare rocks, slippery of the terrain, presence of shadow etc)

The software can be subdivided into four main parts:

1. Data and libraries acquisition
2. Cost functions
3. Global optimization
4. Local optimizationThe case study hereby presented uses a more complex version of the software, that can been considered similar to the basic one but with some “enhances”

However, in this middle version there are some features that can be changed to adjust the results, and/or to show the differences between the modalities of use.

With respect to the first case, it has been given way less importance to the energetic optimization, rather than the actual feasibility of the movement. This means that the road could

even be not optimized with respect to the energetic point of view, rather than guarantee the ability to reach the target point. It may seems trivial, but since this case study has been organized to be used near domain border (e.g. near the sea), it is way more important to be able to be sure that the rover do not touch the water, to avoid losing it. On the other side, having a path that is so near a border may impact the possibile roads to rely on, and this may affect the energetic performances of the rover (that, however, in this case study are secondary).

Here are some characteristics that has been changed in the following try:

- The ability of the software to evaluate the shadow presence on the ground, and include it in the cost function
- The possiblity of the target to move from its initial location and have a generic motion near the domain border
- Distance between the starting and final points

All these components of the software will be analyzed in the following chapters.

7.1 Libraries implementation

In order to make this software work, some libraries have been implemented and used to make it easier the work of the most trivial functions that usually are not implemented in the standard version of Python3, (e.g. the calculation of a square root, or a standard deviation).

Even if nowadays there are many pre-built libraries regarding path planning or decision-taking, none of them have been used here; this is not only given by an obvious semplification and reduction of the work itself, but also to have the most basic functioning without any weighting on the computational cost, since this software has been specifically written to have the lowest effort on the OBDH system of a rover.

These libraries are hereby listed and reported, for a better clarification:

- Math (version 3.10.0): this module of python has been used to calculate the square root of a number (with command “math.sqrt”) and to approximate a number (command “math.round”).
- Mpmath (version 1.2.0): library used for the inverse goniometric functions and cotangent function (“mpmath.asin” and “mpmath.cot”).
- Cmath (version 3.10.0): needed to convert the cartesian coordinates of a set of points into polar coordinates, with the function “cmath.polar(complex(x,y))”.
- Random (version 3.9.0): as its name suggests, this function is used to generate random number in order to obtain stochastic target motion or casual obstacles, with the function “random.randrange”.
- Matplotlib.pyplot as plt (version 3.4.3): used to plot every kind of DEM used, with the relative remarkable points on it. Used mostly with function “plt.imshow” and similar.
- Numpy as np (version 1.21.2): most used library, fundamental to work with multi-dimension array and to manage the DEM files (“np.array”, “np.hstack”, “np.shape” etc).
- Combi (version 1.1.4): necessary to use the function “combi.permspace”, that provide a matrix with all the permutations of a given input vector.
- Tifffile as tiff (version 2021.8.30): this library has been used with the function ”tiff.imread”, that behaves an intermediator between the DEM taken from a satellite as a tiff file, and the software itself that must work it out.

- Statistics (version 3.4): the functions obtained from this library are just a couple, necessary to evaluate the mean value of a vector ("statistics.mean"), and the standard deviation of a 2D matrix ("statistics.stdev").
- Time (version 3.8): basic library used to measure the machine time of execution of the software with "time.time".

The deployment of these libraries is not foregone, since the overmentioned functions could not work with a different distribution of the same library.

7.2 Data acquisition

The first part of the software, right after the libraries inport, there is the data acquisition, that is the way to obtain some data that will be elaborated later.

The process of data acquisition can be performed in different ways: some data are inserted manually while running the script, depending on the necessities of the user (like the starting/final points), others can be modified only in the software code, for an easier use of them (like the coefficients of cost and penalties); meanwhile others could be virtually obtained just connecting with a data center or a remote server/sensor, like the position of an obstacle or the altitude rilevation scanned by a satellite[14].

It is important to notice that these data are not trivial, and the correct tuning of them can make the software works in very different ways, in particular this is bounded to the selection of some parameters of work and the resolution of the ground DEM.

In particular, the main data elaborated in input are the following:

- DEM grid
- Starting and arrival points
- Angles of inclination of the Sun
- Location of obstacles
- Coefficients of cost/penalties
- Data matrices (for the 3rd case study)

All these kind of data analyzed are hereby reported in each relative subparagraph

7.2.1 DEM grid

DEM is the acronym of "Digital Elevation Model" and rapresent a 3D-like model of a geographic zone, and it identify each point with 3 cartesian coordinates.

These coordinates with which a DEM works, are a classical (x,y) cartesian coordinates to indicate a precise latitude and longitude, meanwhile the third coordinate is used to identify the value of altitude above the sea level, for each pixel.

[14]This software has been developed to be as much autonomus as possible, so it could download the DEM matrices via direct link with an external server. This has not been done here due to complexity reasons

These DEM grid are imported from real data obtained from the constellation of satellites "Profumo" and "Sentinel I", and have a resolution of 8m and 10m respectively[15].

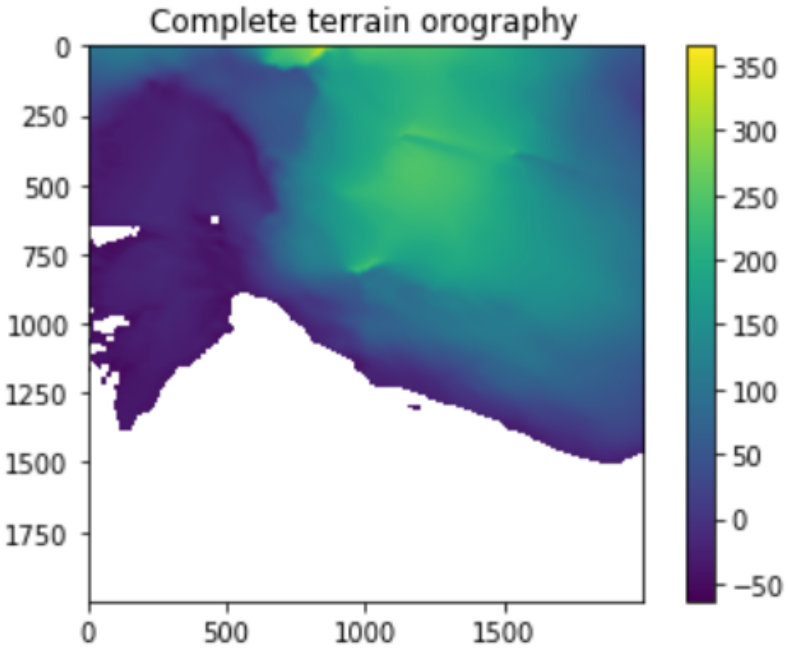

Figure 11: Orography of the terrain for the 2nd case study

The DEM used for the third case study, although, has a resolution that is much lower due to the huge zone of interest (the whole Mediterranean Sea), and it has been reduced by a factor 1000, so each pixel of the DEM in the third case study has a length of 10 kilometers instead of 10 meters.

Every Digital Elevation Model is analyzed in this software as a 2D matrix[16], and is associated to a usability matrix that indicate a boolean value of accessibility of that particular region of space. In a more complex software it could be implemented a system to recognize the obstacle from the satellite or from the modulus of the slope; although for this particular code it has been chosen to avoid any complex evaluation of the spatial accessibility, not (only) due to the unnecessary complexity, but mostly because the evaluation of the eventual possibility to use a particular pixel is strictly bounded to the capabilities of the rover. For this reasons, each DEM is correlated to a random-generated matrix of accessibility value.

[15]These data are provided by Telespazio Belgium
[16]From the computational point of view, this is managed by the library "numpy"

To better simulate the stochastic distribution of the obstacles, it can be selected the density of eliminated pixels along the whole region.

The DEM matrix is usually acquired straight insiede the code, and not requested by the user, to facilitatate the use of it.

7.2.2 Starting and arrival points

In this software, it is necessary to have an initial point (indicated with A) and a final point (B), and the choice of them is strictly requested at the user.

During the input phase, the reading of those points is associated with a check to be sure that the two points are inside the DEM matrix, in a usable zone of space and that they do not coincide. If these parameters are respected, then the points are recorded as input.

There is also the possibility to consider the path between more points, just inserting a third point between A and B. In this case the work of the software will be facilitated and the computational cost reduced.

For the third case study there is also the possibility to insert in input the name of the starting and final city, instead of the coordinates. This has been done to facilitate the use of the software.

7.2.3 Sun inclination

Considering a device mostly powered by the sun light, as it happens in most of the cases when dealing with martian/lunar rover, it is important to include in the software the research of a path that is sunlighted as much as possible. This is obviously given by the necessity to be able to power up the drive system in every possible situation, even if this could require a longer road or a steeper one.

Even if this is not strictly necessary for rover on the surface of the earth, it could be still useful to be able to choose a sunlighted path in case of electric/hybrid propulsor with an electricity generation driven by the sun.

For this reason, it has been necessary to indicate the inclination of the sun, and this must be measured with respect to two different directions. In the software, these angles are simply indicated as alpha and beta, and their functioning are indicated hereby:

- Alpha: is the angle of inclination of the sun with respect to the horizon, or the complementary of the angle between the sun and the zenith.

- Beta: is the angle between the projection of the sunrays on the ground and the equatorial line.

Both these angles are used in the following chapters to determine whether if the pixel considered is sunlighted or not, and determine the relative cost for the movement.

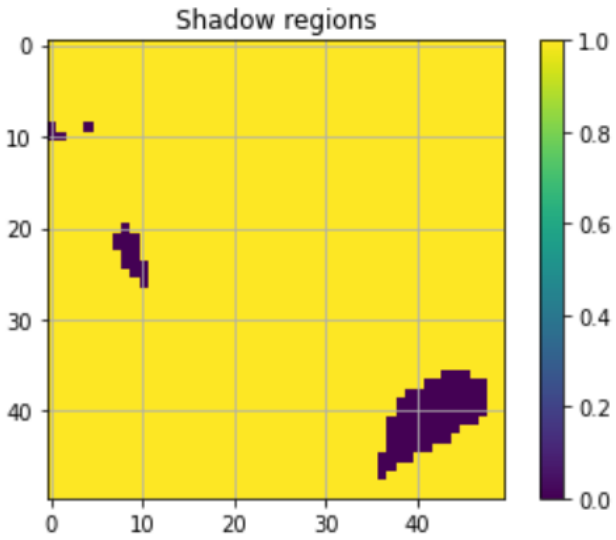

Figure 12: Example of shadow/illuminated region

7.2.4 Location of obstacles

The movement of a rover is not only subjected to a cost due to various factors, but can also be totally obstacled by not crossable elements. These elements could be inserted manually one by one, to have a realistic placement in space, or can be imported with an external matrix (as done for the third case study). The main version of the software is based on the random localization of these obstacles to have the most unpredictability cases, and avoid "particular cases" in which the software could run in a simplified form. Obviously, for a realistic situation the exact location of obstacles can be implemented with an external database or with on-board sensors like stereocameras or infrared sensors.

Regarding the kind of variables used, the obstacles are stored both in a 2D array, to compare it with the DEM array, and in two different 1D vectors[17] to enable a quick representation and utilization in the code.

[17]These vectors are used to indicate the X and Y of each obstacle

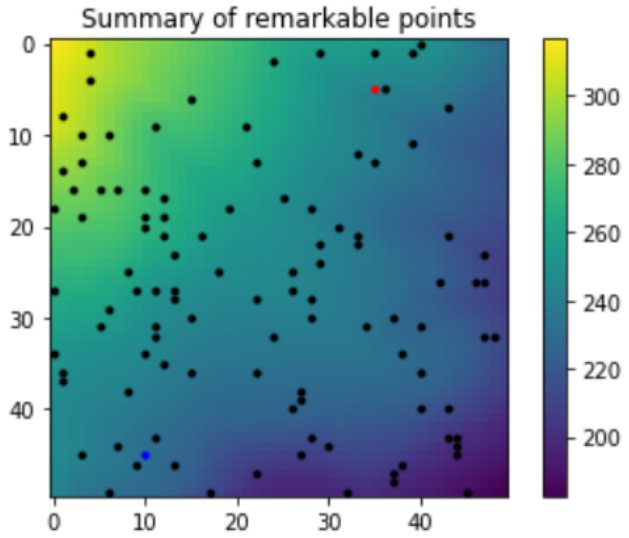

Figure 13: Example of obstacles disposition

7.2.5 Coefficients of cost/penalties

This software is mainly based on a cost approach, assigning a penalty to eveything that can be considered as unfavorable for a rover, like an elevated slope, obstacles or opposite wind. These coefficients are totally arbitrary, and can be decided depending on the situation.

The input of these coefficient is done inside the code, and is not requested by the user, but it's important to note that a different set of coefficients can change drastically the final result of the path.

An important note regarding the cost coefficients is about their values. Since there is no link with actual power usage, it's not important the absolute value of them but only their proportion with the other cost coefficients[18].

The sign of the coefficients could be chosen as negative, if it is associated to something positive for the rover (like a descent). In that case the total cost is reduced while passing in a zone with a negative cost.

7.2.6 Data matrices (for the 3rd case study only)

In the third case study the software is modified to be used in a different situation, that is the motion of a boat in the Mediterranean Sea. For this purpose it is not feasibile to use a

[18] e.g. choosing two cost coefficients like 1 and 2 is equal to choose them as 10 and 20 or 0.1 and 0.2

difference of altitude as a cost parameter, although there are other kind of data that can be deployed..

Variable name	*Code*	*Minimum value*	*Maximum Value*	*Unit of measure*
Wind speed	fg10	0	16	[m/s]
Variation of atmosferic pressure	msl	-2	12	[hPa]
Cloud Area Fraction	tcc	0	1	[-]
Waves speed	v0	0	0.7	[m/s]

Table 1: Data used for the 3rd case study

Those data sets are taken from the Profumo constellation, on 31 of May 2021 and can indicate various variables. The ones used in this version of the software deal with the wind and waves speed, the pressure distribution, the cloud fraction area and the precipitations. All of them are expressed in the table below, with the characteristics of maximum and minimum values, units of measure and fill values.

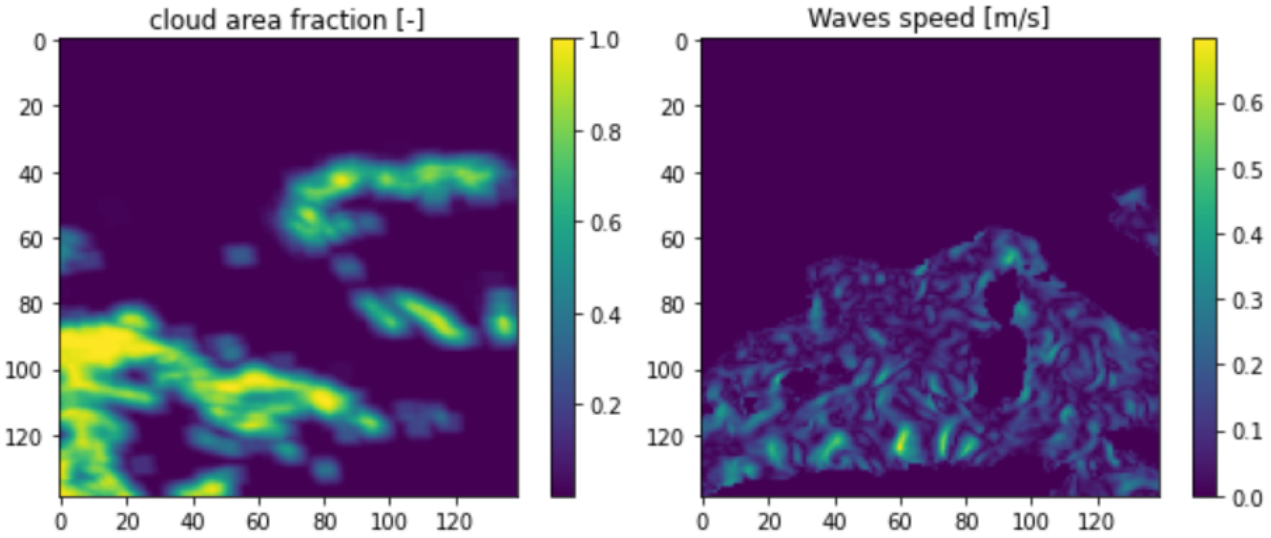

Figure 14: Example of Meteorological data

It is important to notice that this kind of data can be obtained in different ways, considering also onboard devices and sensors; If the external elements are evaluated with a satellite the

result is a long-range prevision and a good extimation of the global situation. At the same time the data evaluated with onboard devices are valid for a local scale and could be used for a very accurate evaluation of the near future prevision. One example of this is a meteorological prevision[19], that can be evaluated in advance by a satellite but only a local sensor can measure it with an excellent precision and in real time.

Let's notice that the third case study is the only one to have an excess of data to be used, since every kind of meteorological data is almost negligible for a planetary rover (and its main cost function is given by difference of altitude and terrain impredictability)

All these values of metereological informations can variate with time, unlike the usability of a zone due to presence of rocks or cliffs, so these data matrices must variate with respect to time. For this reason the input of data matrices happens with a precise temporal slots, and these matrices are alternated and automatically updated with the passing of time. For a reason of data limitations, only 6 interval of 1 hours are considered, but all the next data are easily available.

[19] A meteorological prevision can be more useful for the third case study, since a boat in the sea is supposed to be more influenced by the meteorological condition than a planetary rover could be

7.3 Cost functions

The cost functions are the base of this software, since it deals with the evaluation of different paths and the selection of some of them based on the cost that they implies.

The evaluation of cost is done by different cost functions, that are implemented in the code and are used in each step taken.

For what concern the unit of measure of these coefficients, they're considered all as the inverse of the maximum value.

The different cost functions are associated to different event. hereby are reported the main ones:

7.3.1 Coefficient of cost for difference of elevation

These coefficients are indicated as η_{up} and η_{down}, and can be used to penalize the presence of a determined slope of the terrain. Their use can be linked with a linear proportionality or a quadratic one (to penalize more the steeper zones). The coefficients of cost may be different from a descent phase rather than an ascent one, since the mechanical power required is totally different. It's important to notice that the values of those coefficients must be tuned depending on the characteristics of the drive system of each single rover. Let's consider the two different possibilities:

- η_{up}: upward movement. The coefficient of cost of upward movement is used multiplied to the terrain slope to find a value of cost. This value is usually considered as unitary since it is the main cost function. All the other cost coefficients is expressed as multiple or submultiple of this value
- η_{down}: downward movement. This coefficient is still multiplied by the modulus of the slope, so the coefficient itself is positive in order to to represent a cost (because otherwise the slope should be negative in a descend road). Notice that in any case it must be followed the relation $|\eta_{down}| < \eta_{up}$: this is due to the fact that a descent road must always be considered less expensive rather than an uphill. It's also important to notice that the coefficient η_{down} could be considered negative in presence of systems of energy recovery (such as KERS or similar[20]), because in that case the effective cost bounded to the difference of altitude would be negative.

[20]These kind of devices are called "Kinetic Energy Recovery System" an can accumulate energy from breaking or from a descent

7.3.2 Penalty for the move

Every step taken, in any kind of direction, is subjected to a minimum cost that is necessary to balance the motion resistance of the ground and (eventually) atmosphere, and this cost is indicated with π_{move}

This penalty can have a different range of values, depending on the external situation: a slippery terrain[21] shall have a low value of π_{move} due to the low resistance; a rover that is passing over a viscous or rough terrain should be penalized more.

From the computational point of view, a lower value of this penalty allows the rover to turn around some point that are just considered "not convenient", even if they are crossable.

7.3.3 Penalty of usability

This element of the cost function is not a coefficient of cost but a simple penalty cost, because is not multiplied by a physical quantity, but is just added to the cost function in case the rover pass over an unusable zone.

It is indicated with the symbol $\pi_{usability}$, and has a theretical value of $+\infty$. Actually, it is considered as an outscale value of 10^9 since it is easier to manage with respect to an infinite value.

Actually this value is only used in case of computational errors, since the matrix of usability avoid any movement in unavailabile zones of the DEM map.

The value of $\pi_{usability}$ is simply evaluated with the following relation:

$$\pi_{usability} = \begin{cases} 10^9 & \text{if } usability[i][j] = True \\ 0 & \text{if } usability[i][j] = False \end{cases}$$

7.3.4 Penalty for sunlight shadow

The penalty for the sunlight shadow is related to the cost to be considered for a movement in a zone without the positive presence of sunlight.

For a planetary rover, the sunlight can be literally vital, since it's used to obtain energy through the solar arrays and to keep the temperature in the working range. The presence of sunlight can also be positive for the visibility of some scientific devices that does not work with infrared rays.

[21] E.g. ice or snow

It is indicated with the symbol π_{shadow}, and has a value that can be chosen depending on the kind of constraint and environment. For a rover that is powered only by solar arrays, the value of π_{shadow} should be really high, almost comparable with the value of $\pi_{usability}$; meanwhile for a rover with other power sources[22] or in different situations, the value of π_{shadow} can be considered significantly lower or even neglected.
The function of shadow penalty can be easily found as:

$$\pi_{shadow} = \begin{cases} 10^2 & \text{if } SolarObstacle[i][j] = True \\ 0 & \text{if } SolarObstacle[i][j] = False \end{cases}$$

where the boolean value of the variable SolarObstacle is stored in a 2D array and can be found with an algorithm that keeps in count the position of the sun through a polar reference frame, and the position of the nearest obstacle. This algorithm can be found in the paragraph dedicated to the functioning of the algorithms.

7.3.5 Coefficients of cost for environmental conditions

These coefficients are implemented to make the third case study work since is based on the optimization of a naval route, so the cost of movement cannot be evaluated just with the difference of altitude (also because it would be hard, with no difference of altitude), so the set of data used have been chosen between a major database of possible information regarding the available rilevations. All the data used are taken from Profumo satellite on 31 of May 2021, and have been obtained with an hourly sampling. For this reason is possible to update the metereological condition every hour.
All these kind of datas are reported hereby, with the relative cost coefficients:

- Wind Speed: η_{Wind}
- (difference of) Pressure: $\eta_{Pressure}$
- Cloud fraction: $\eta_{CloudFraction}$
- Wave speed: $\eta_{WaveSpeed}$

[22] e.g. RTG generators

7.3.6 Coefficient of cost for DEM variance

The cost coefficient for the variance of the reduced DEM is a value indicated as $\eta_{Variance}$; it is used to evaluate a cost bounded to the variance of the reduced DEM for the global optimization.

After having obtained the DEM in a reduced scale (5x5 or 10x10 pixels), at every single pixel is associated a region of space with a bigger area.

The reduced DEM is obtained just with an average altitude of the considered zone, and is associated to a matrix (with the same size of the reduced DEM) that contains in each pixel the value of altitude variance with respect to the mean one.

This data is useful for the global optimization, to penalize the routes that go through an unpredictible terrain.

This kind of penalization is no more used for a local optimization.

7.3.7 Coefficient of cost for local approximation

This coefficient is simply indicated as $\eta_{LocalApprox}$ and is the bound between the global and local optimization.

Actually, its functioning is to keep the local path as near as possible to the global one, but still without compromise the optimization for each single step.

The value of $\eta_{LocalApprox}$ must be chosen carefully, since an elevate value could make all the other costs absolutely futile, and in this case the rover would only follow the generic path evaluated with the reduced elevation map, with no (relative) cost for the height variance, obstacles and similar; on the opposite side, for a low value of $\eta_{LocalApprox}$ the path could be totally detached from the global path prevision, ignoring de facto the generic path previously hypotized.

7.4 Global optimization

The global optimization is based on the path finding for the whole journey, from the starting point A until the final point B.

This mechanism is an important part of the software, because using only a local approximation could lead to a path that starts with few optimal steps, but soon could just end in a very expensive zone, from the energetic point of view.

For this reason, the optimization needs to be conveyed through these two different steps: global and local.

The algorithm written has the purpose to find a generic road, that could be partially followed in the secondary step of the software, that is the local optimization and movement.

This goal can be achieved reducing the global altimetry grid into a smaller one, called "reduced DEM", and perform a cost evaluation for all the possible roads. This allow to find the most convenient global path, that will be refined with the local optimization in the last part of the software.

7.4.1 Steps needed for global optimization

In order to perform a global optimization, the algorithm uses some precise steps that are hereby indicated in chronological order of deployment.

- DEM reduction: in this step the DEM matrix is reduced into a smaller one (5x5 or 10x10 pixels), and at every single pixel is associated a region of space with a bigger area, called quadrant. The value of altitude of the single pixel is obtained with an altitude average, and can be used to have a global summary of the terrain orography. It is important also to evaluate the value of variability for each single pixel, since they represent a bigger area; this value is obtained with a simple standard deviation and is stored into a matrix with the same shape of the reduced DEM.

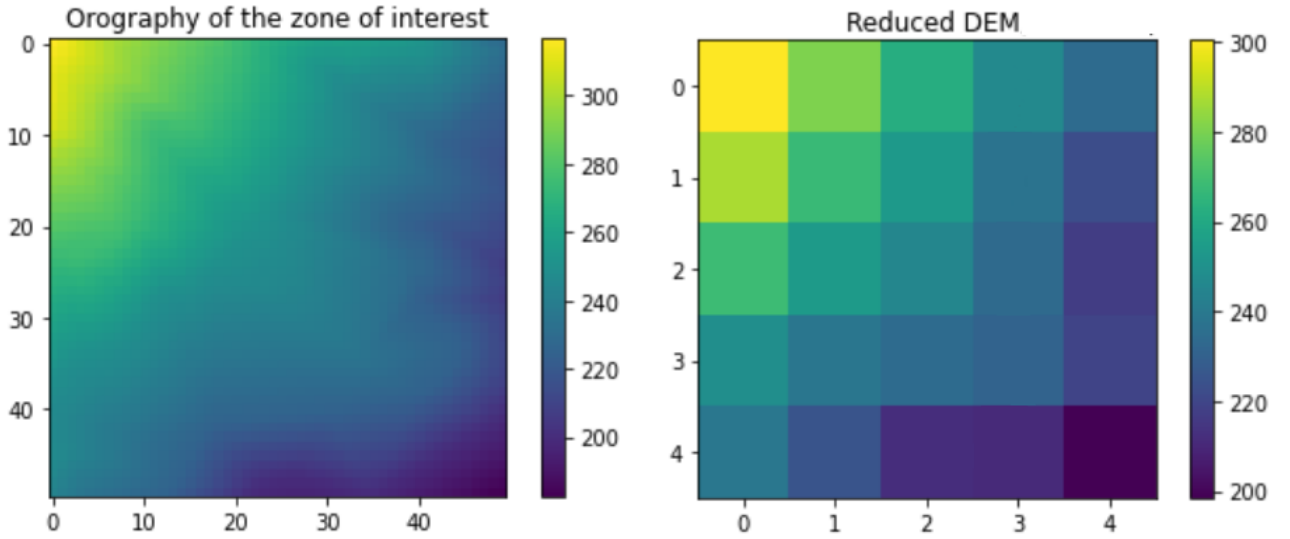

Figure 15: Example of DEM reduction

- Reduced coordinates of the significant points: once the DEM matrix has been reduced, it is necessary convert the coordinates of the initial and final points into the quadrants of the reduced DEM. To do so, it is performed an algorithm that is able to indicate in which part of the new grid the significant points will be, and give those coordinates as output.
- Analysis of the possibles moves: once the starting quadrant and final quadrant are indicated, it is possible to obtain the variation of quadrants on both X and Y axis, and evaluate all the possible roads to perform this kind of movements, keeping into account some possible variations due to the presence of big areas unavailable for crossing.
- Cost evaluation: this part of the global optimization is fundamental to select a particular road between all the ones found before. The cost evaluation for this stage is quite trivial, and keeps into account only two main driver parameters: difference of altitude (or the slope) and the value of reduced DEM variance; both these parameters are multiplied by a cost coefficient. These parameters are very reductive with respect to the complexity of the problem, but are a perefct compromise between computational cost and precision in global path evaluation.
- Path evaluation: once the roads are selected, and for each of them is evaluated the cost needed to be completed, and the whole path is then completed, from the initial quadrant until the last one.

- Evaluation of the global path nodes: the global path evaluation is needed to have a generic information about the direction to be taken in the local optimization. For this reason is necessary a further step in which every single quadrant crossed can be converted into a precise point (the center of the quadrant itself) to be used as a markpoint in the cost evaluation for the single moves in the local optimization. These coordinates are then stored into two different vectors.
- Data output and graphic rapresentation: the last part of the global optimization is based on export the data obtained and inform the user of the software about what has been done. For this reason the output consist in:

  - Kind of movements necessary to be done to reach the final quadrant
  - Number of strategies evaluated and strategy chosen.
  - Specific data about the chosen strategy (cost, distance, etc)
  - Graph plot of the reduced DEM
  - Graph plot of the DEM variation
  - Graph plot with the reduced DEM and all the fundamental information about the global path (starting and arrival quadrants, path plot, intermediate points markdown)

All these steps are then analyzed from a technical point of view in the third part of the thesis, in which every single algorithm is explained in details.

7.5 Local optimization

The local optimization is the second main step for the functioning of the software; the aim of this part of the software is NOT to conduct the rover from the starting point to the arrival, but just to get closer to the target, going through the optimal subset of steps.

For this reason, it must be integrated with the global optimization, that helps the rover to keep the right route toward the target.

During the local optimization, the algorithm looks for the necessary displacements that must be computed, and achieve a part of them after having analized all the possible routes and the relative costs.

Note that this part of the software is recursive, and is carried on while the location of the rover does not coincide with the location of the target.

7.5.1 Steps needed for local optimization

In order to perform a global optimization, the algorithm uses some precise steps that are hereby indicated in chronological order of deployment:

- Obstacle placement[23]: in the first two case studies there is the necessity to simulate the presence of uncrossable obstacle all along the DEM grid. For this reason it has been implemented a function, called "RandomUsability" that places obstacles in a stochastic way in the working zone of the map. This function has a probability of 5% to convert a pixel of the grid into an obstacle that cannot be crossed. The location of these obstacles is then stored into the matrix of usability and into two different position vectors (with the X and Y coordinates of these points).
- Target displacement: using a random function, there is the possibility for the target to move in a near location. This funcion may be activated or not, depending on the need. In case of necessity, the target can also follow a regular path using a displacement law. If the target moves, this information is stored into a boolean variable (to know if it has moved or not), and the new coordinates are stored into two different vectors (with the X and Y coordinates of these points).
- Evaluation of displacements: in order to reach the target, it is necessary to obtain the total delta X and delta Y from the local position (called point P) and th final point (B). for this reason, the displacement are indicated as:

$$\Delta X = X_B - X_P$$

$$\Delta Y = Y_B - Y_P$$

- Analysis of the possibles moves: once the global mandatory displacements are computed, is then possible to evaluate all the possible roads to perform this kind of movements, keeping into account all the possible combinations of the movements needed.
- Analysis of the alternative roads: in case the previous part of the algorithm should identify a road that does not include enough vertical or horizontal movements, in that

[23] only for the first two case studies

case the complementary function AlternativeRoadOV include the possibility to do a longer path with steps in the neglected directions. Note that this does not make the chosen road longer, because the road selected will be the most economic one, and if the alternative road ends up to be energetically more expensive than the other, it will be neglected.

- Evaluation of the cost functions: the cost functions include different elements bounded to the efficiency of the movement. These cost and penalty coefficients, explained in details with their respective algorithms, are used to find the cost to be afforded to cross a determined pixel of the DEM grid. The cost functions are based on many parameters:

  - Availability of the region of space
  - Presence of sunlight or shadow
  - Kind of move and difference of altitude to be faced
  - Climatic and environmental conditions
  - Distance from the nodes of the global path

- Cost and feasibility for each road: every single road that has been found in the previous steps is then analized with the cost function in order to evaluate how expensive could be, and each alternative is then associated to an adimensional number that represent the total cost of it.
- Path finding: after having analized the single possible strategies, the best one is selected. It may happen that more than one strategy is considered to be the cheapest one[24]. In that case the strategy is selected randomly from the best ones.
- Perform the movements planned: The strategy found is then applied, step by step, to decrease the distance of the rover from the target. This is not done impulsively, but considering various possible events between each single step. During this phase, the motion of the selected strategy is interrupted in two different cases:

  - If the rover perform too many steps with the same strategy, is then blocked, and the strategy evaluated again.

[24]This happens in case of quite even terrain, because there are many alternative roads

  - If the rover is near the target, and suddenly the random function makes the target moves, then the strategy could change again. In case the target should move mantaining a big distance from the rover, than the direction of the rover is considered to be asymptotically, so the strategy selected does not change.

- Data output and graphic rapresentation: the last part of the global optimization is based on the export of the data obtained to inform the user of the software about what it has been done. For this reason the output consist in:

  - Kind of movements necessary to be done to reach the final target
  - The final position of the rover and the target, and the relative distance (that should be equals to zero)
  - Specific data about the chosen strategies (total cost, total distance, etc)
  - Graph plot of the complete DEM used
  - Graph plot of the rover motion, the target motion and the usability of the terrain
  - Graph plot of the presence of solar sunlight or shadow across the map considered
  - Time needed by the computer to find the necessary road, from the initial data aquired to the last graph plotted.

All these steps are then analyzed from a technical point of view in the third part of the thesis, in which every single algorithm is explained in details.

8 Functioning of the algorithms

The functioning of the overall software is based on the utilization of some algorithms that can discretize the work done and reduce the complexity of the software itself.

These algorithm are implementated as python functions, with input(s) and output(s).

Using function is necessary to decrease the amount of code to be written, and therefore the overall complexity.

Each function has a field of application and a precise purpose, so hereby are reported and analyzed all of them, with the relative diagram of work.

Remind that the variables in the algorithms can be used in a local[25] or in a global[26] way, even if this is not specified in the following simplified diagrams (plus some other features of the code that have been neglected for complexity reasons).

8.1 Data acquisition

The part of data acquisition is strictly related to the input or the production of the data necessary for this program to work, as already mentioned in paragraph 7.2

The process of data acquisition can be performed in different ways: some data are inserted manually while running the script, depending on the necessities of the user (like the starting/final points), others can be modified only in the software code, for an easier use of it (like the coefficients of cost and penalties); meanwhile others could be virtually obtained just connecting with a data center or a remote server/sensor, like the position of an obstacle or the altitude rilevation scanned by a satellite[27].

It is important to notice that these data are not trivial, and the correct tuning of them can make the software works in very different ways, in particular this is bounded to the choose of some parameters of work and the resolution of the ground DEM.

[25] i.e. local variable means that are used ony in that function, and then are deleted
[26] i.e. the global variables are valid through all the code
[27] This software has been developed to be as much autonomus as possible, so it could download the DEM matrices via direct link with an external server. This has not been done here due to complexity reasons

8.1.1 Tiff image reading

This first part of the software is simply an input algorithm to aquire the tiff file that contains the information about the terrain.

The file analized is imported and converted into a numpy matrix to be used further.

Note that this algorithm can be used also to acquire the data necessary for the third case study.

All the variables used here are intended to be globally valid inside the code.

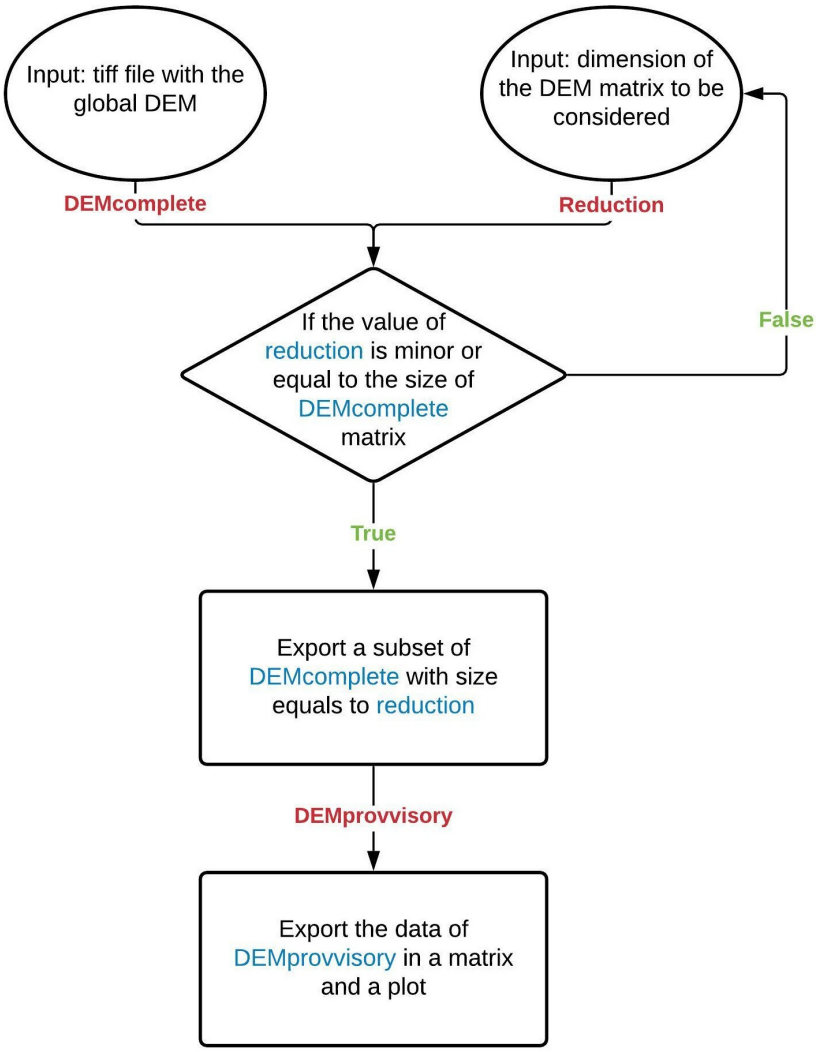

Figure 16: Flow chart of the algorithm of tiff file import

8.1.2 Coordinate input

The second algorithm is still used as a ground preparation for the main software.

It is based on the input of the main parameters used, starting from the DEM that characterize the ground.

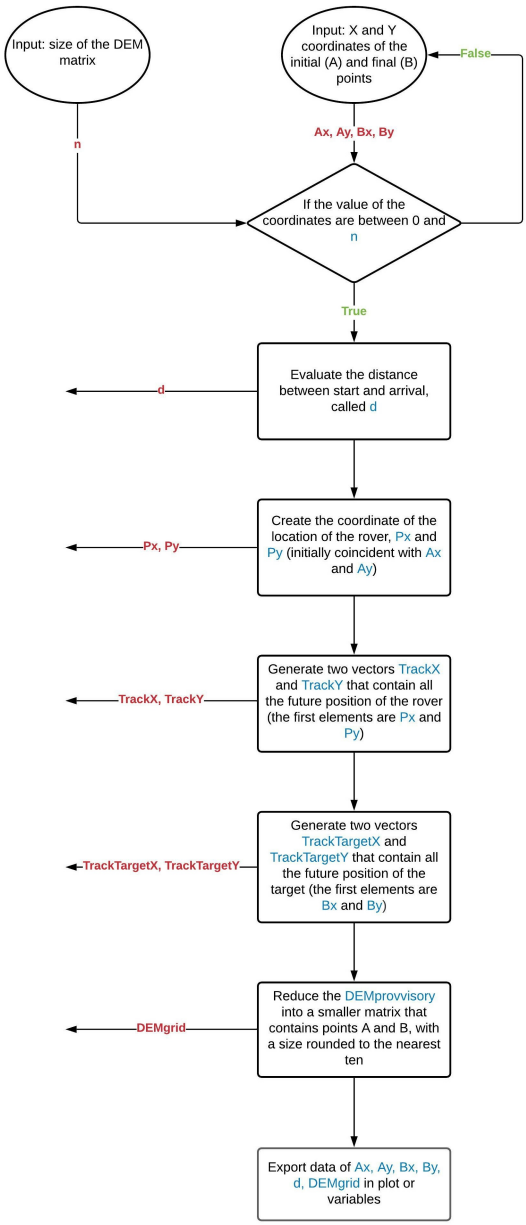

Figure 17: Flow chart of the algorithm of data input

8.1.3 DEM usability

This function is used to perform a generation of a random usability matrix, that corresponds to the presence of obstacles along the path. To create this matrix, it has been used a random function that substitute a real function based on sensor data relevation. It is necessary to control that the initial/final points are not considered as unusable, to avoid unsolvable cases.

Note that all the variables used are intended to be global.

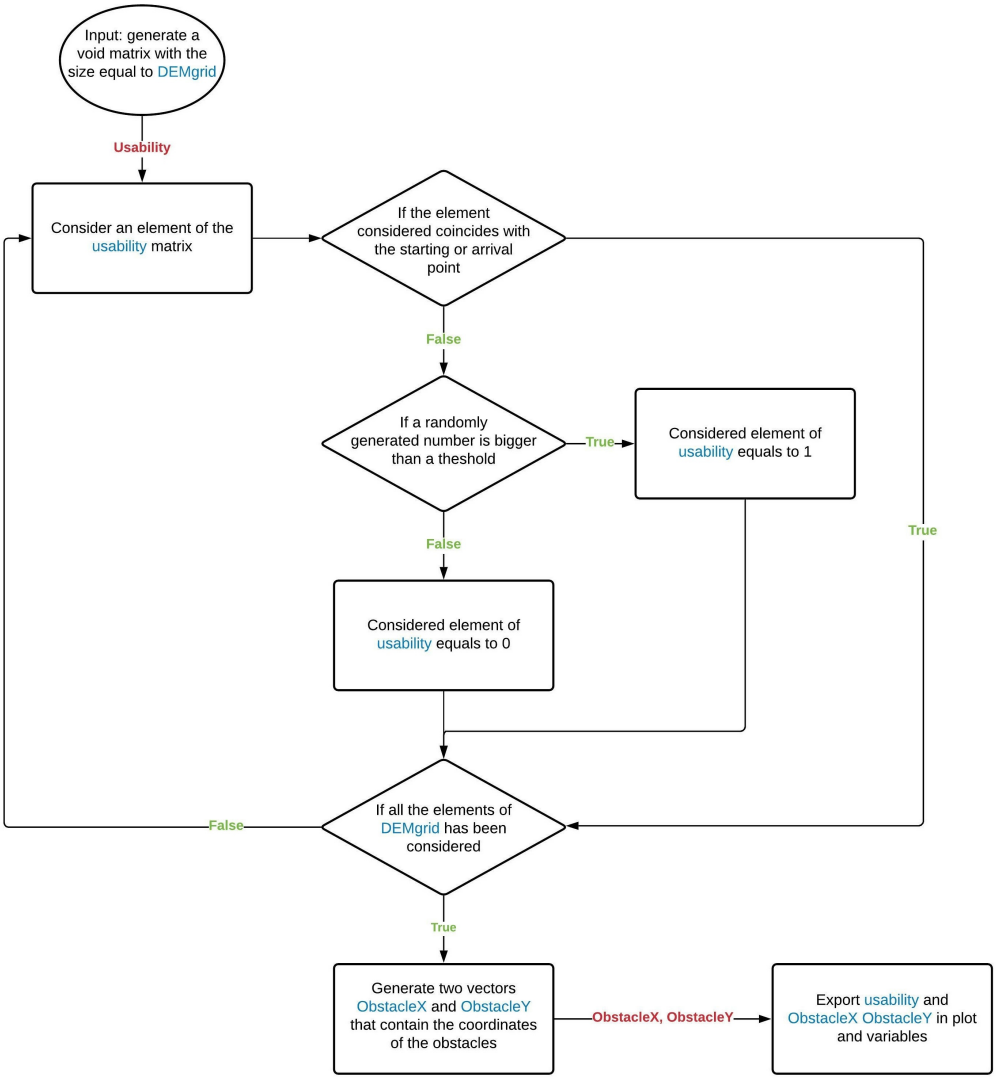

Figure 18: Flow chart of the algorithm of usability matrix generation

8.2 Cost functions

The main characteristic of this software is that it is based on a cost-extimation path selection, for this reason the cost functions are really important to be used in the best possible way.

The cost functions are based on a penalty method, for every "negative" element that can be encountered during the trajectory will be assigned a cost or a penalty.

The cost is composed by two elements: the driving paramenter (e.g. difference of altitude, pressure value, etc) and a cost coefficient (to be hypotized).

The input of these coefficients is done inside the code, and is not requested by the user, but it's important to note that a different set of coefficients can change drastically the final result of the path.

An important note regarding the cost coefficients is about their values. Since there is no link with actual power usage, it's not important the absolute value of them but only their proportion with the other cost coefficients[28].

The sign of the coefficients could be chosen as negative, if it is associated to something positive for the rover (like a descent). In that case the total cost is reduced while passing in a zone with a negative cost.

[28]e.g. choosing two cost coefficients like 1 and 2 is equal to choose them as 10 and 20 or 0.1 and 0.2

8.2.1 FindSolarObstacle

In this algorithm the output required is only a boolean value that indicate whether if the ground is exposed to sunlight or not.

This is helpful to introduce a penalty into the cost function.

The values of alpha and beta are the complementary inclination of the Sun with respect to the Zenith (alpha) and the angle formed by the projection of the sunrays above the terrain with respect to the positive direction (beta).

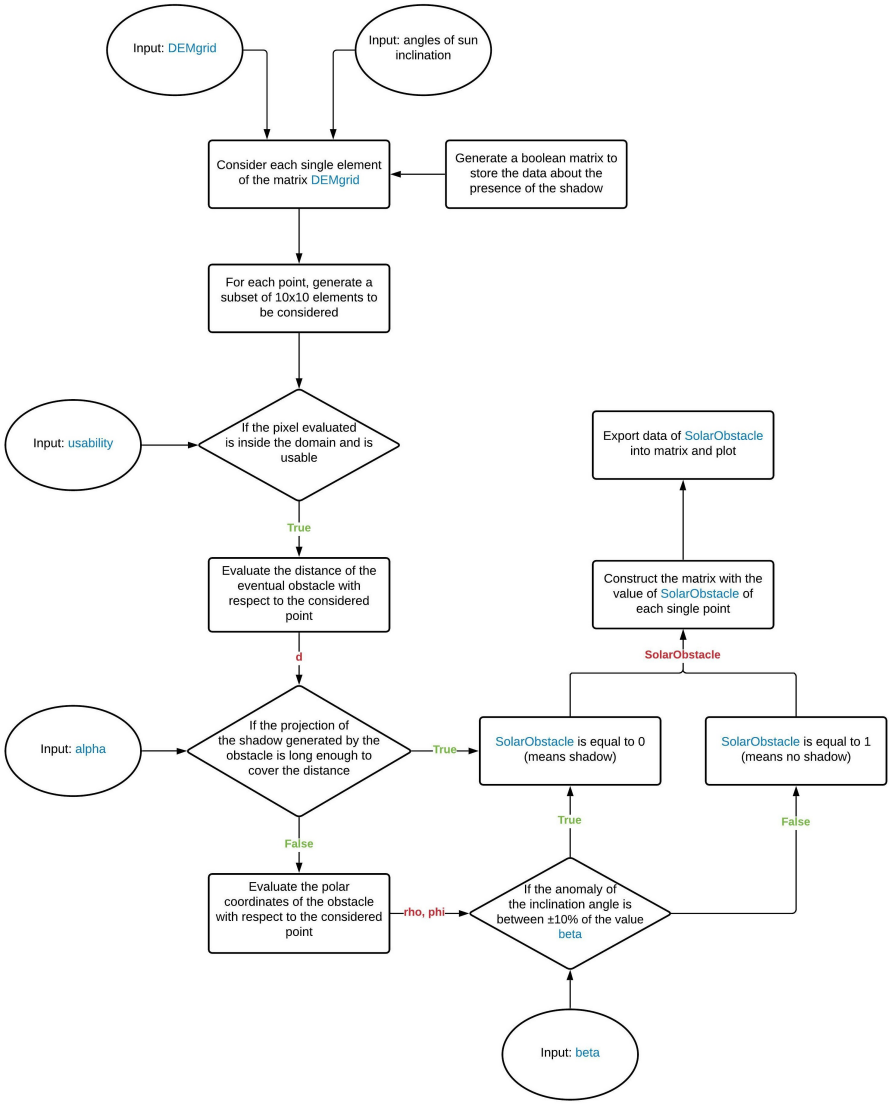

Figure 19: Flow chart of the algorithm to find the shadow zones

8.2.2 Cost function

This is the algorithm of cost function, that include the sum of all the components to be penalized.

The cost bounded to the distance from the next node is the link between the local and the global optimization, and is a cost proportional to the distance of the rover with respect to the ideal road selected with the global optimization.

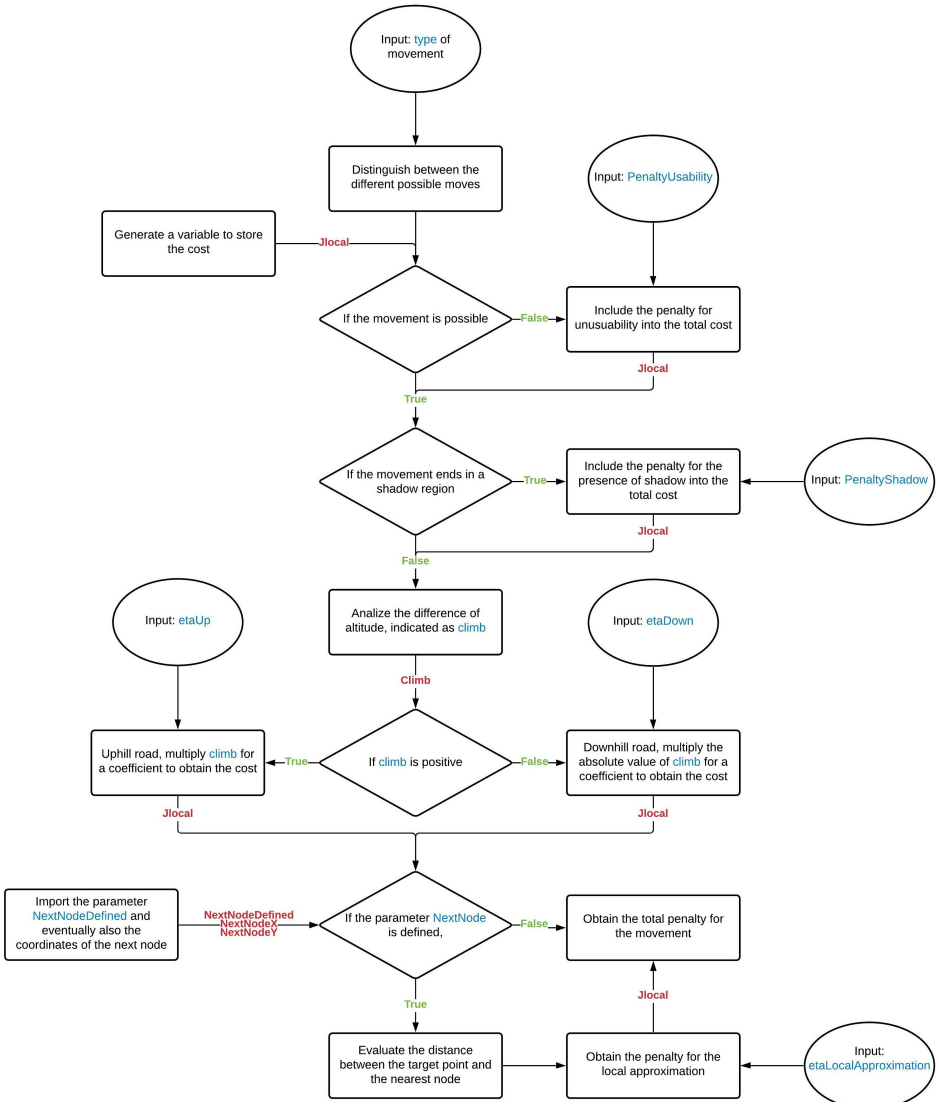

Figure 20: Flow chart of the algorithm of cost function

8.3 Global optimization

The optimization of the path must be done at two different levels. Organize the path in order to have a perfect local optimization could lead to an unfeasibile amount of computational power required, or otherwise a road that only selects the best "first steps", without having care to complete the whole path with a low level of cost.

Otherwise, choosing a path that is only optimized globally means that the rover doesn't have any chance to analyze the single obstacle or the problematic spots, leading to an increasing mechanical power required.

The link between these two components is made by the prevision of the ideal road made by the global path finder, and the penalization of the detachment of the real path with respect the ideal one, in the local optimization.

For this reason, the following steps has been carried out to ensure the utilization of a contracted DEM to obtain a global ideal path, relying on a reduced cost function.

Hereby are reported the main functions used to perform this optimization.

8.3.1 Matrix contraction

This algorithm has the aim to generate a contracted DEM to reproduce the orography of the zone of interest, but with a reduced resolution. The size chosen for the contracted DEM is 10x10, because is a good compromise between good accuracy and low computatonal cost.

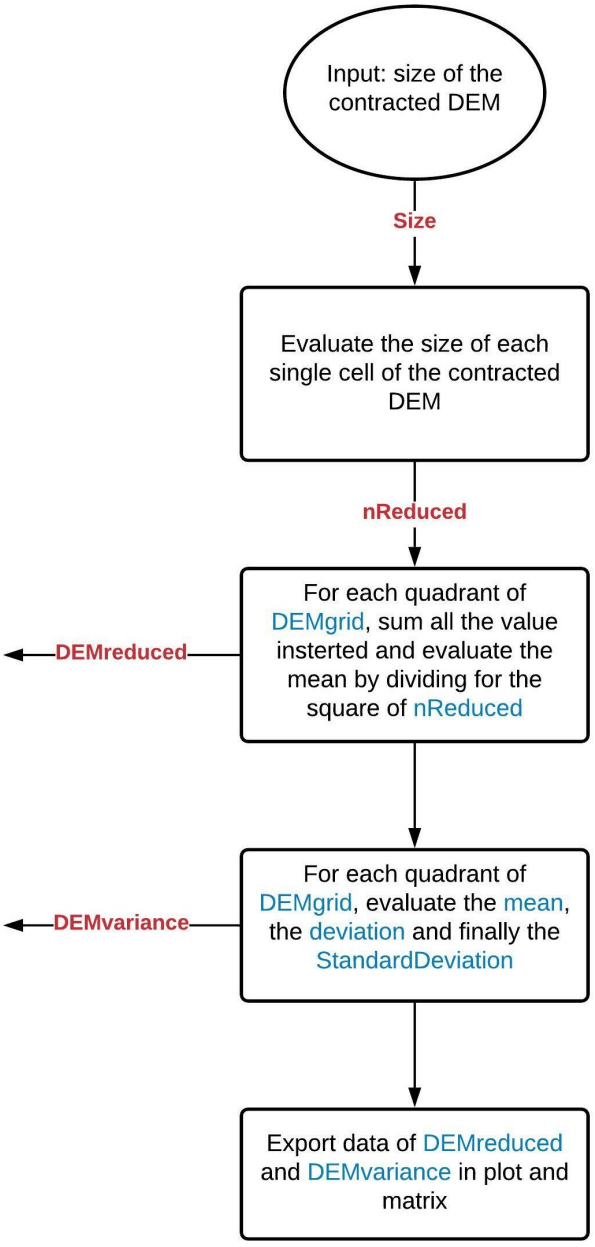

Figure 21: Flow chart of the algorithm of matrix DEM contraction

8.3.2 ReducedCoordinatesConversion

Once the DEM has been contracted, it is important to identify which zones of it are used (also called "quadrants" because each single pixel represent a bigger zone of the orography).

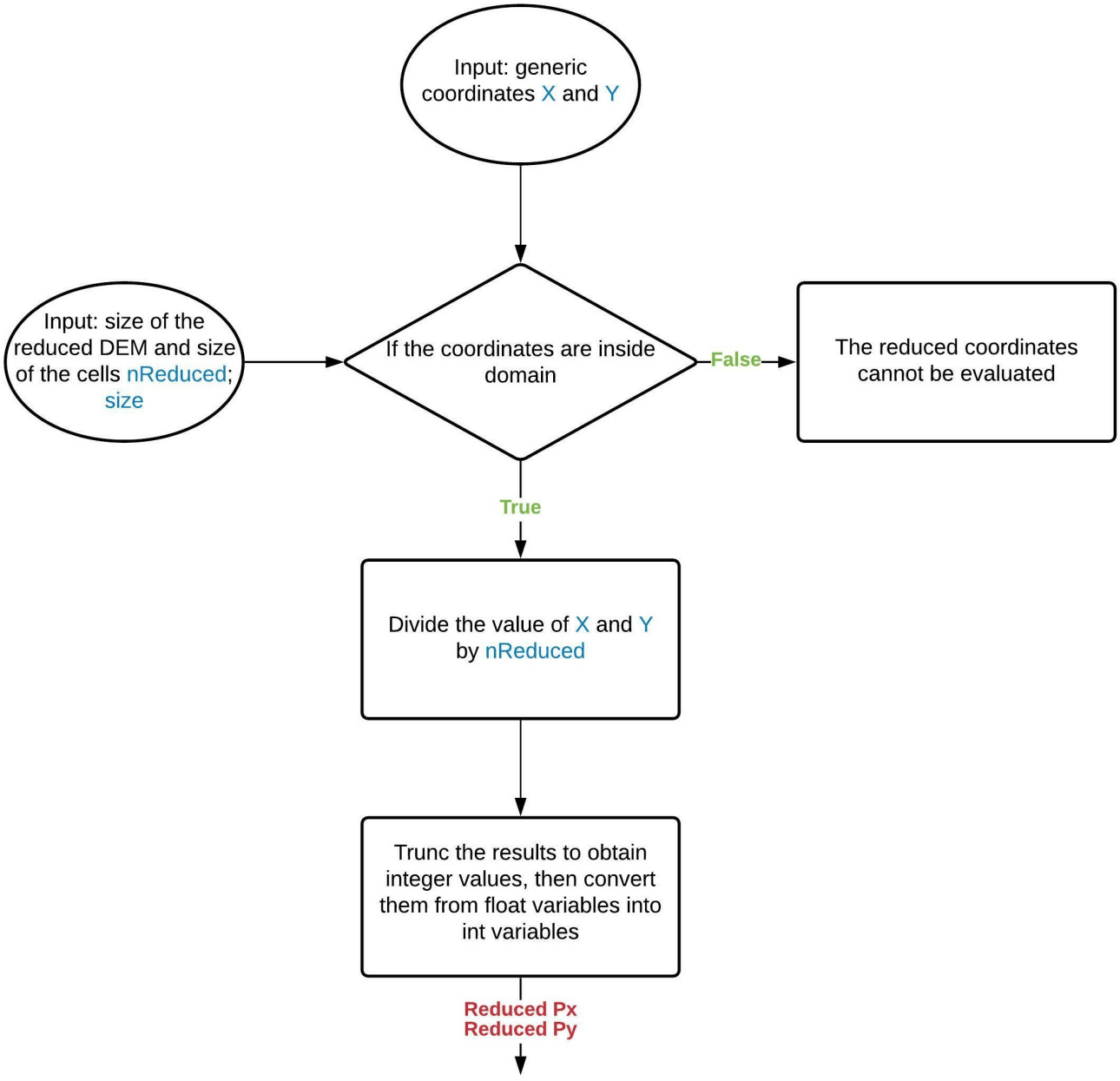

Figure 22: Flow chart of the algorithm of reduced coordinates conversion

8.3.3 ReducedCoordinatesOptimization

Once the DEM has been contracted, it is possible to find the least expensive path between the initial and final quadrants, and this happens with the help of the other functions previously used.

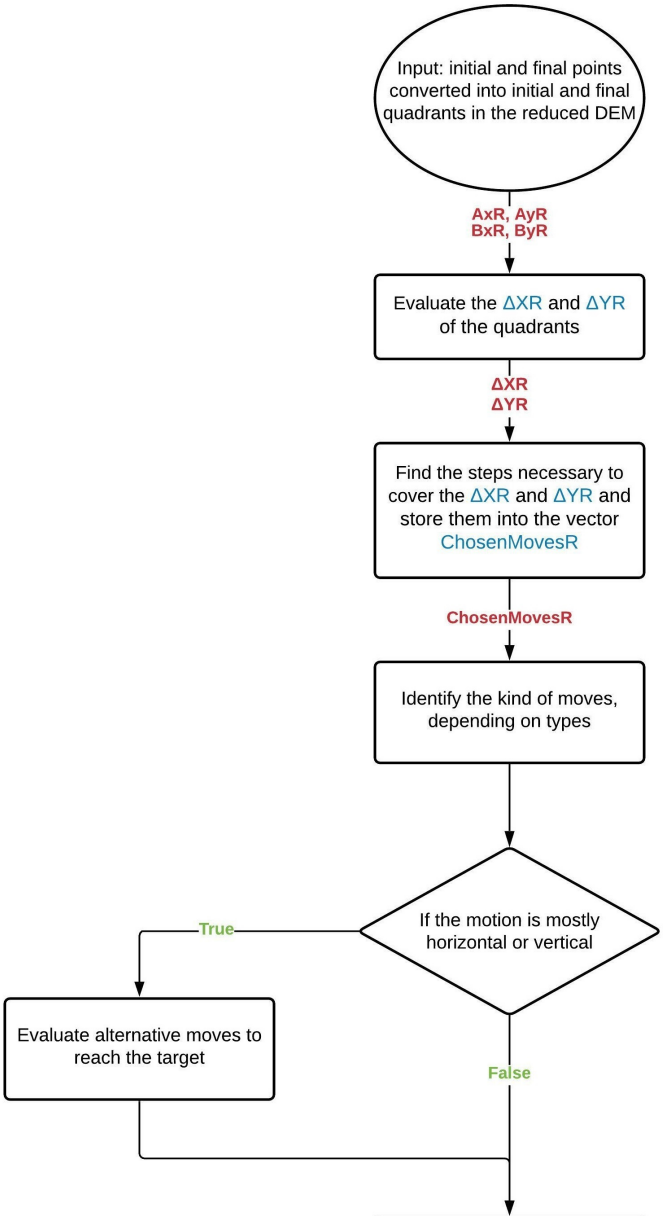

Figure 23: Flow chart of the algorithm of global optimization

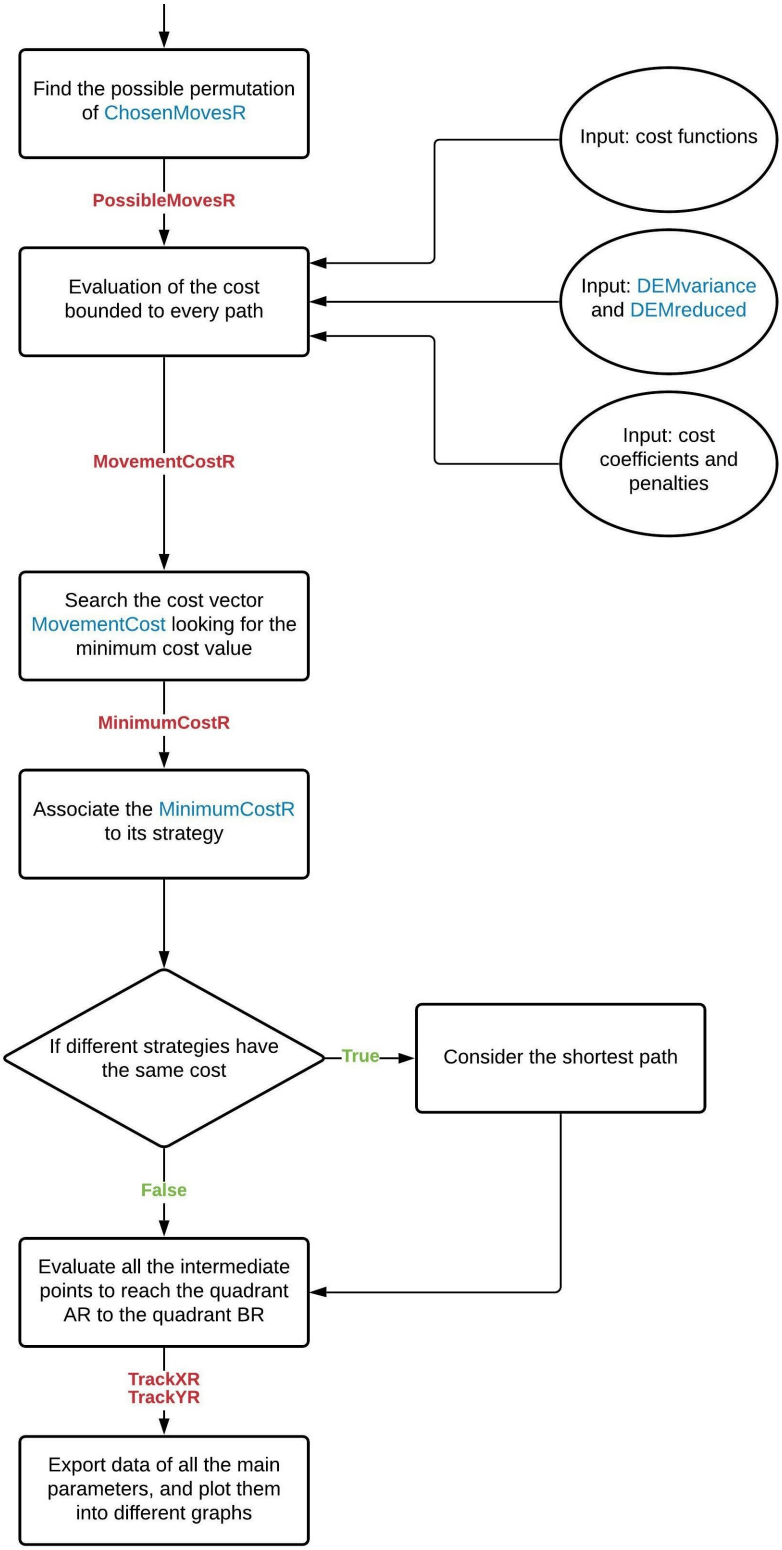

Figure 24: Flow chart of the algorithm of global optimization (continue)

8.3.4 FindNextNodes

This part of the software represent an important connection between the global and local path planning. Its functioning is based on finding the nearest node to be reached by the rover. The following node evaluated is then used in the cost function to penalize the movement that are distant from it.

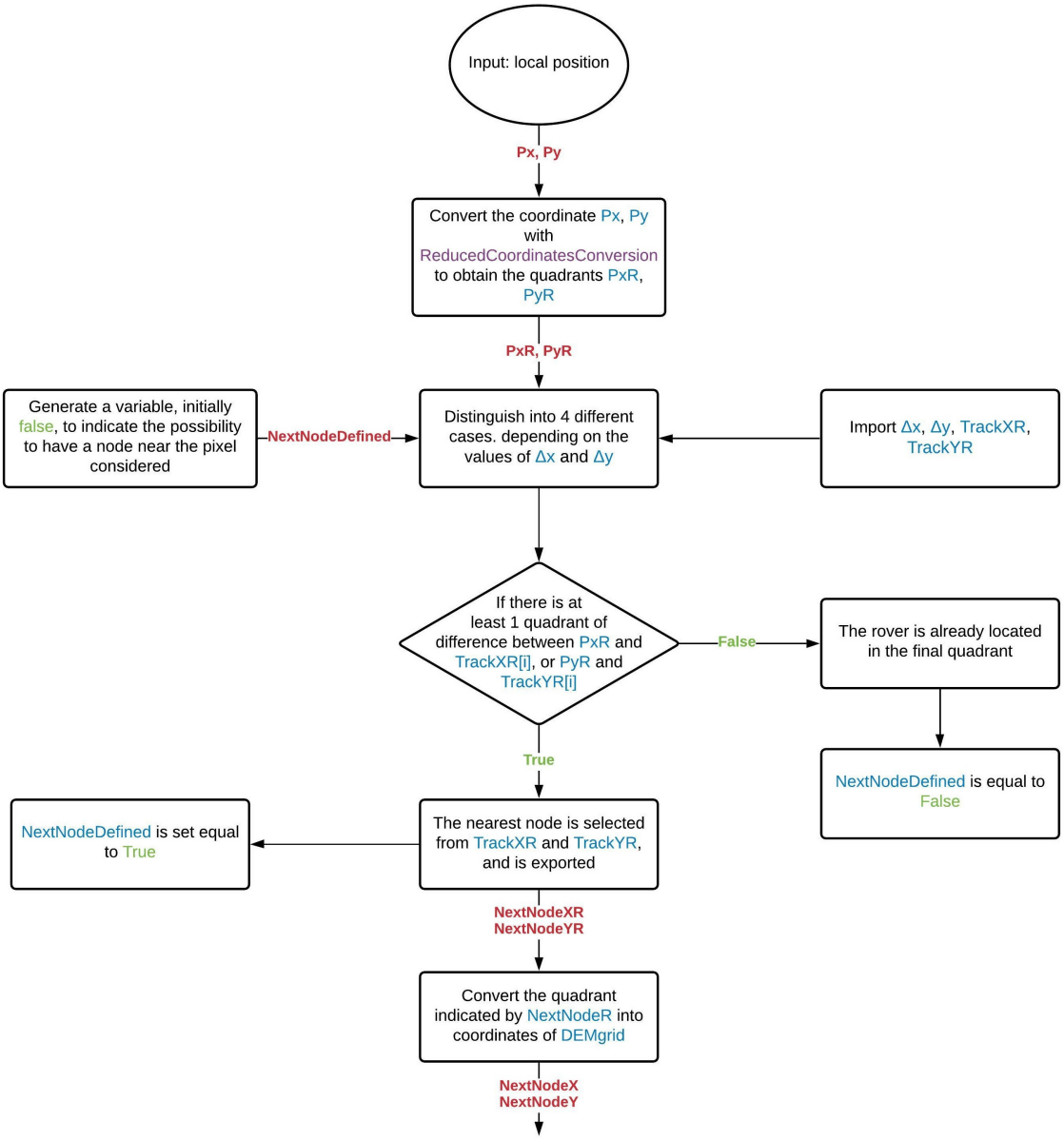

Figure 25: Flow chart of the algorithm of global optimization (continue)

8.4 Local optimization

The algorithms regarding the local optimization are the keys of the software, they deal with the integration of the cost functions and the optimal global path with the features needed to obtain a working road to link the initial and final points.

Note that the link between the local and global optimization is based on the research of the nearest node of the ideal path, at which is associated a penalty based on the distance from the rover.

The main functions used are the following:

- Move function
- Target motion algorithm
- Main road finder
- Alternative road finder
- Path planner function

Each of them has a specific purpose that lead to the final goal.

Note that this part of the software is strongly recursive, and is carried on while the location of the rover does not coincide with the location of the target.

8.4.1 Move function

This algorithm is the "locomotion system" of the software, since it is the part of the code that actually is made to move the rover, and update all the informations about its movement (like cost, trajectory, etc). In this part of the software it is also implemented a part of code that ensure that the movement is feasibile.

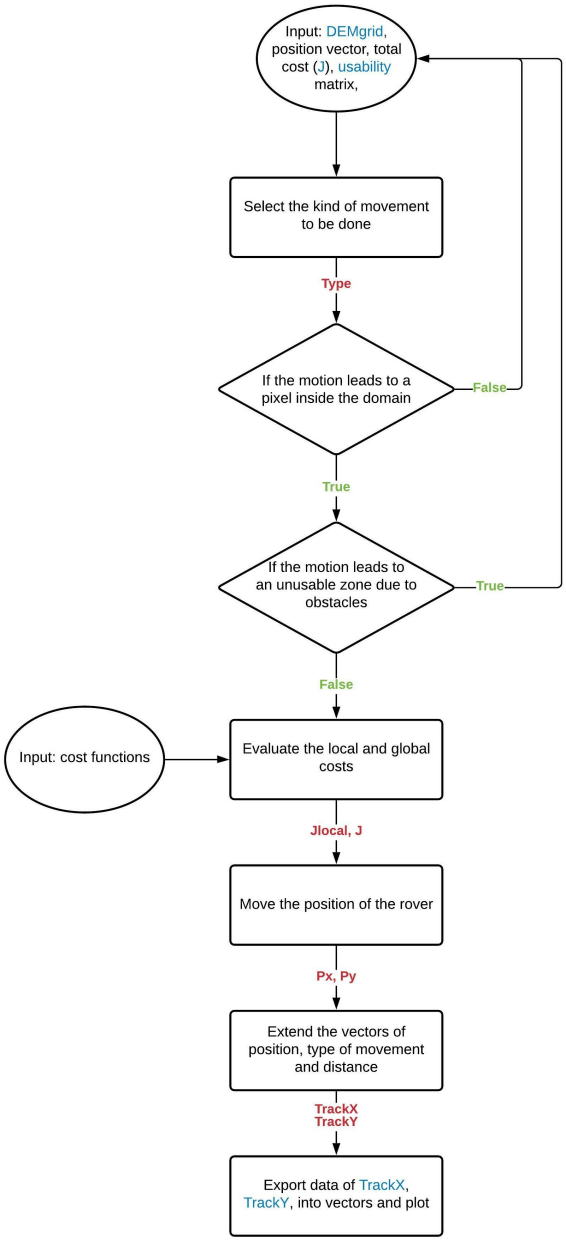

Figure 26: Flow chart of the algorithm of the move function

8.4.2 TargetMotion

In a real utilization of this softwar, it may be necessary to include the eventual motion of the target.

Of course, this is likely to happen in a precise way (following a determined trajectory) or with a randomic function. For this reason, it has been implemented a function that may (or may not) induce the target move casually, with a certain amount of probability[29]

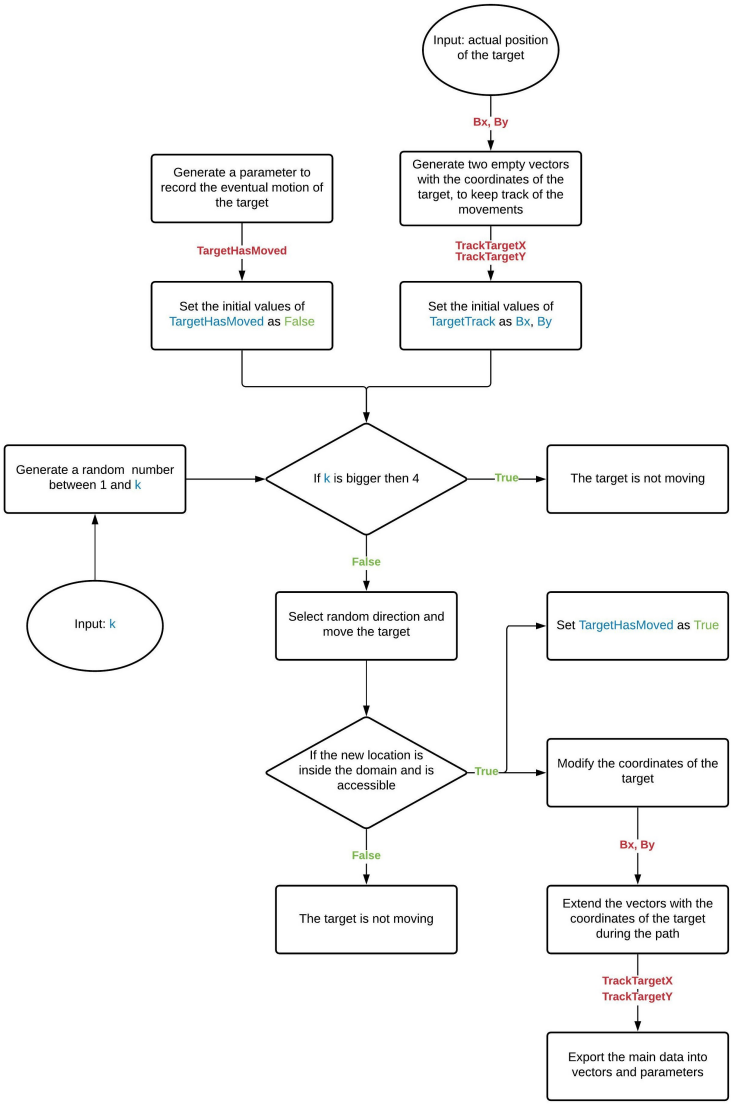

Figure 27: Flow chart of the algorithm of target motion

[29]The probability of the motion of the target can be decided changing the coefficient k

8.4.3 MainRoad

This part of the software is very important to be able to find the final path. The algorithm MainRoad is able to find the step necessary to lead to the final point, and is eventually completed by the algorithm AlternativeRoadOV, that is able to find complementary roads to reach the target.

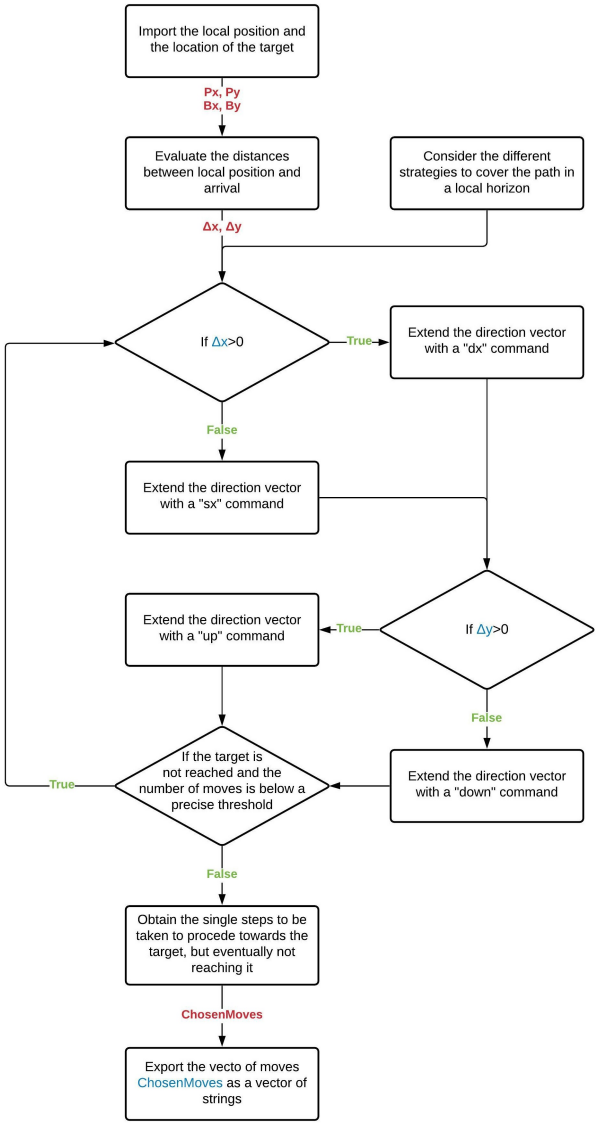

Figure 28: Flow chart of the algorithm of main road selection

8.4.4 AlternativeRoadOV

This part of the software is used to find eventual alternative roads in case of limited capability of motion (i.e. only horizontal/vertical movements hypotized).

Notice that all the kind of alternative motion is totally balanced to avoid the rover to get too far from the target

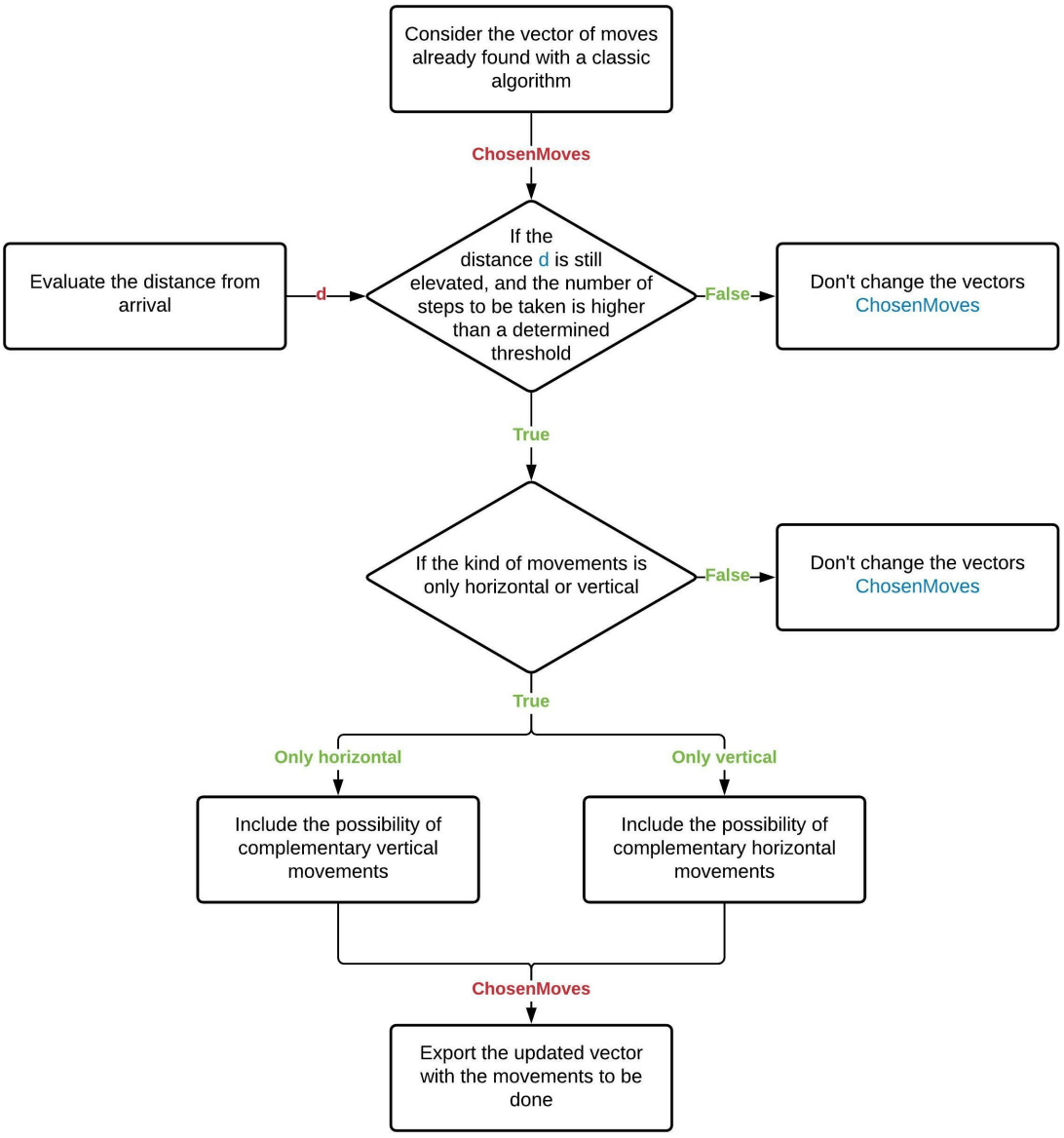

Figure 29: Flow chart of the algorithm of alternative road selection

8.4.5 Path planner

This is the last part of the software, but is the most important. It includes all the previous functions and informations obtained, and has the aim to finalize the research of the best road.

At the end of this algorithm, the data output is done in a clear way to help the user understand the steps taken, the cost evaluated, the target motion and all the other features that can be useful to interface with a software of this kind. All the main data are also shown in plots that are easy to analyze, for a better user experience.

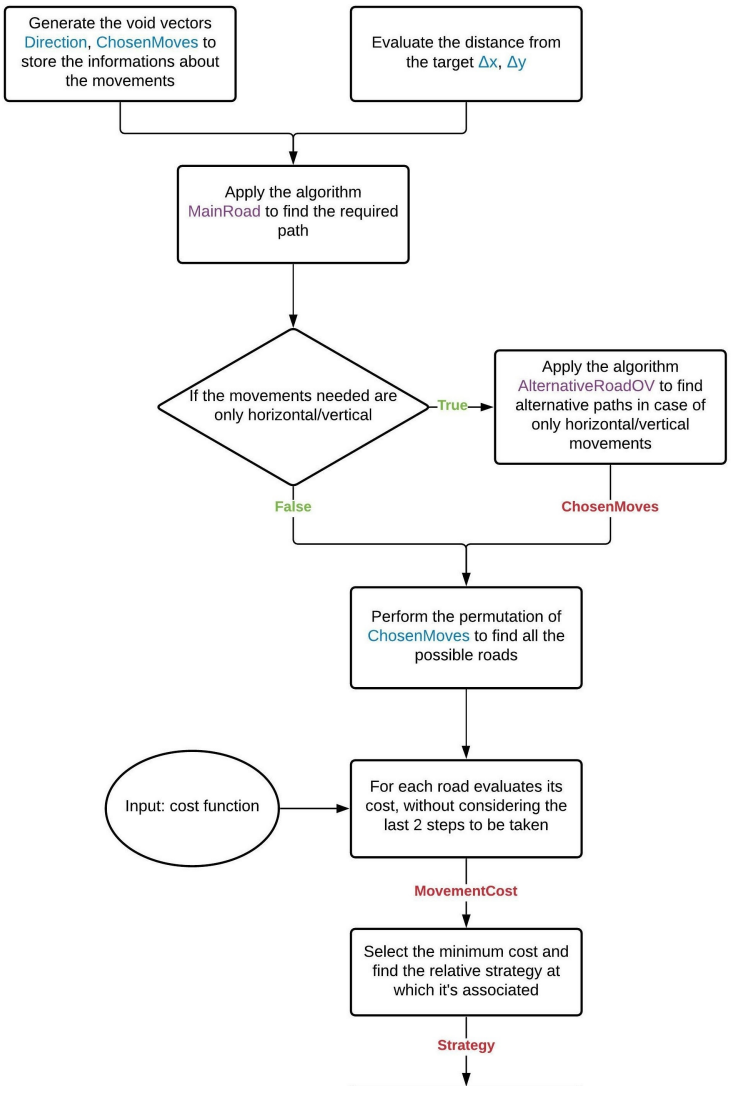

Figure 30: Flow chart of the algorithm of path planner algorithm

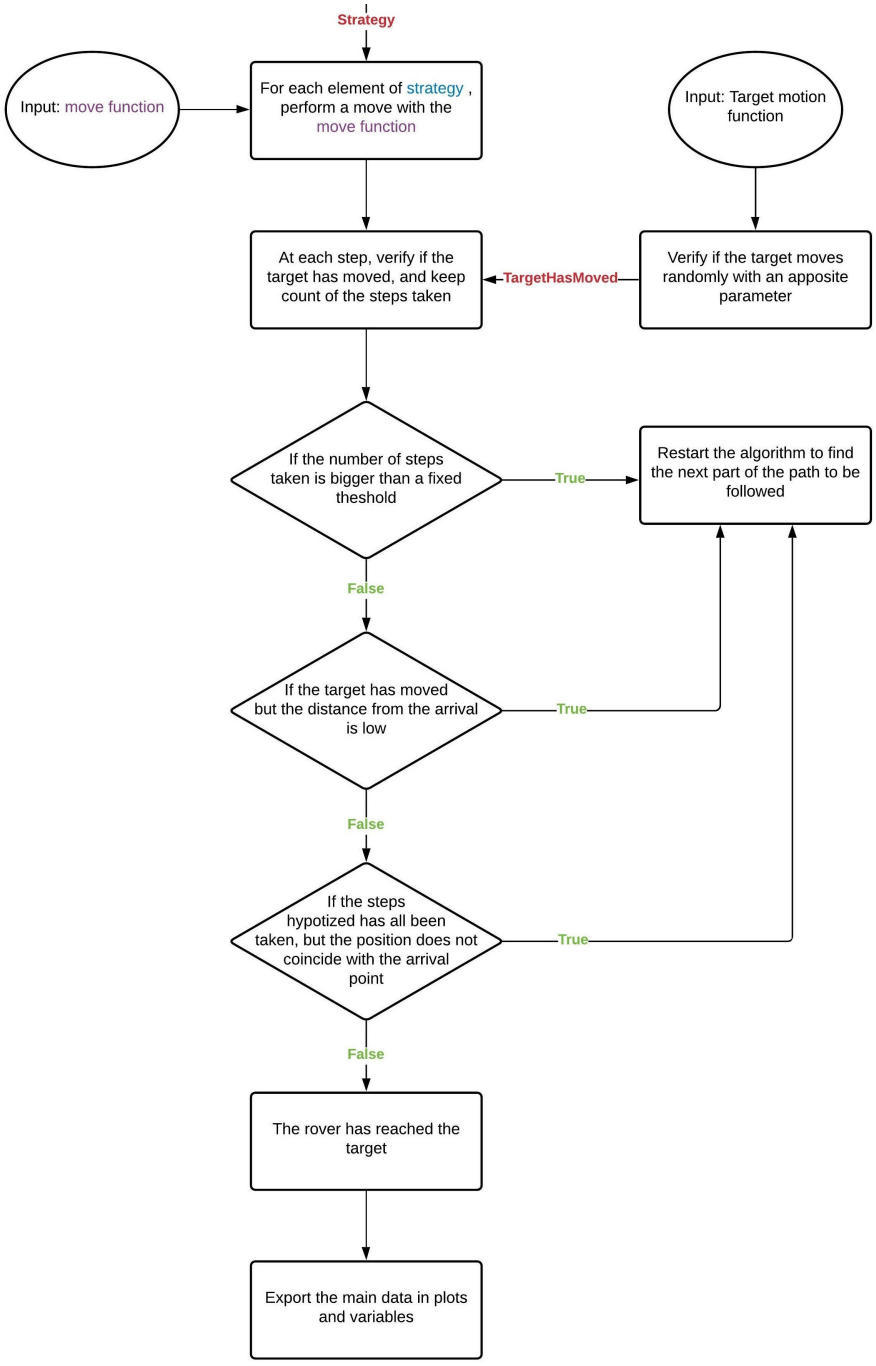

Figure 31: Flow chart of the algorithm of path planner algorithm (continue)

Part III
Case studies and data output

After having realized the path planning software, it was necessary to test it in different situations and with other complexities. For this reasons it has been evaluated the functioning of the software in three different cases, and for each of them present further differentiations to test all the possible functionalities as well.

The three main case studies have strong differencies one from the other, starting from the region of interest.

Hereby are reported briefly the differences between these case studies:

- The first case study is located in a remote zone of Italy, in particular in Calabria[30], to test the basic functionalities of the software: ability to move a rover, capability of selecting a road, weighting a cost function, select some obstacles and avoiding them, and so on. This ensure the working of the main features, necessary to operate in any kind of environment.
- The second case study is located in an Antartica region, the motivation why it has been chosen this zone is simply bounded to two main features. First of all, the orography of the Antartica region is subject to frequent changes depending on the season, and this can be considered as a big obstacle from the data availability. The second feature that suggested to use this place as second case study is the presence of large unusabile zones, and this can be considered as an "advantage" to test the software in extreme conditions. The test done in this region are aimed to prove the ability of the software to lead the rover in border-line region.
- The third and last case study is located in the Mediterranean Sea, in a very vast region, and it is the most complete and complicated version of the software. To realize this case study it has been necessary to modify some features of the software leaving intact the core of it. The reason behind the selection of this particular region of the Earth is bounded to test the ability of the software to carry the rover in a complex and articulated region of space, with the add of a more complete and efficient cost function, given by an increased data availiability.

[30]The place chosen is in Calabria because also robots have feelings

All the three case studies has been evaluated in critical and extreme conditions, with different cases and features, to ensure the ability of the software to overcome every kind of difficulty.

The utilization of these location, all on Earth, instead of extra terrestrial zones, is only given by the abundance of data that can be obtained for this planet, compared to the others.

In the following paragraphs will be described in details the results given by the software, in addition with the main feature of the code and its variation, and all the possible graphs to understand better the output.

It is important to underline that, if not specified differently, for all the case studies the software has been runned on an Asus laptop with a processor Intel Core i7, and a computational power of 16 Gb of RAM memory. Only a single case study has been performed on a low-performance device, to simulate the computational power of a real OBDH system of a planetary rover.

9 First case study

9.1 Description

This first case study is the most basic one between the three hereby proposed, and it has been done to test the main features of the code.

It is located in the southern zone of Italy, in particular the province of Crotone in Calabria. This zone has a variuos orography and it is perfect to test the main ability of the software in standard conditions. The resolution of the map is 10 meters, and the data are obtained with the satellite Sentinel I.

The main features tested in this case study are the following:

- Ability to find a generic route between two points, even without any optimization, and conduct the rover until the end[31].
- Ability to implement different cost functions and make them work, that include the cost based on the difference of altitude (or terrain slope), penalty for the presence of shadow, penalty for obstacles and penalty for external events.
- Ability to analyze the global area considered and obtain informations from it, considering the terrain variance, the average altitude of the ground, the density of uncrossable obstacles, and similar data.
- Ability to select a particular road depending on the cost associated, after having analyzed the possible roads and the relative feasibility, and chose the most economic one.
- Ability to create and then identify the obstacles that can limit its road, both in a local and global perspective, with a penalty-like strategy.
- Ability to bound the global and local optimization, to increase the energetic performance and reducing the computational power.
- Ability of the software to work in case of very long distances, without losing accuracy or giving unreasonable results.
- Ability to be runned on a low computational device, like a Raspberry Pi, and give the result in a reduced time.

[31] Even if it is a goal way simpler than the necessary, it is an important basic feature

It is important to remark that this region of space has been considered only from an altimetric point of view, no roads or real obstacles have been considered. The presence of external elements on the map can be implemented with another software, able to limit the space domain analyzing the terrain and the structures built on it.

Note also that the region really used for the software is a subset of this, for computational and realism reasons

Figure 32: Orography of the zone considered for the 1st case study

9.2 Software code variations

The case study hereby presented uses a basic version of the software, that can been considered as a the "standard" one.

However, in this basic version there are some features that can be changed to adjust the results, and/or to show the differences between the modalities of use.

Here are some characteristics that have been changed in the following try:

- The ability of the software to evaluate the shadow presence on the ground, and include it in the cost function
- The possiblity of the target to move from its initial location and have a generic motion along the domain
- Distance between the starting and final points
- Cost coefficients and penalties, to analyze the differences between varius weightings
- Hardware platform, to obtain different computational evaluations and timings

9.3 Data input

9.3.1 Subcase 1A

The first set of data input regard the most standard version of the software. All the main functionalities are active, and the points chosen as start/arrival have a moderate distance.

The starting point is identified as A(35 ; 5) and the final one is B(10 ; 45), with a distance of about 47 pixels[32], corresponding at a distance of about 470 meters in the reality. The distance chosen is moderate, to allow the software work in the most classic case.

The software runs with the motion of the target enabled, and consider also the optimization depending on the presence of sunlight.

For what concern the sunrays presence, the angles chosen to represent the inclination of the sun were equals to $\alpha = \frac{\pi}{10}rad$ and $\beta = \frac{\pi}{4}rad$, and those angles corresponds to the complementary inclination of the Sun with respect to the Zenit and the Sun inclination with respect to positive direction of x axis. The value chosen for β is totally casual, meanwhile the value of α is set as a low value to allow the presence of both shadow and illuminated soil.

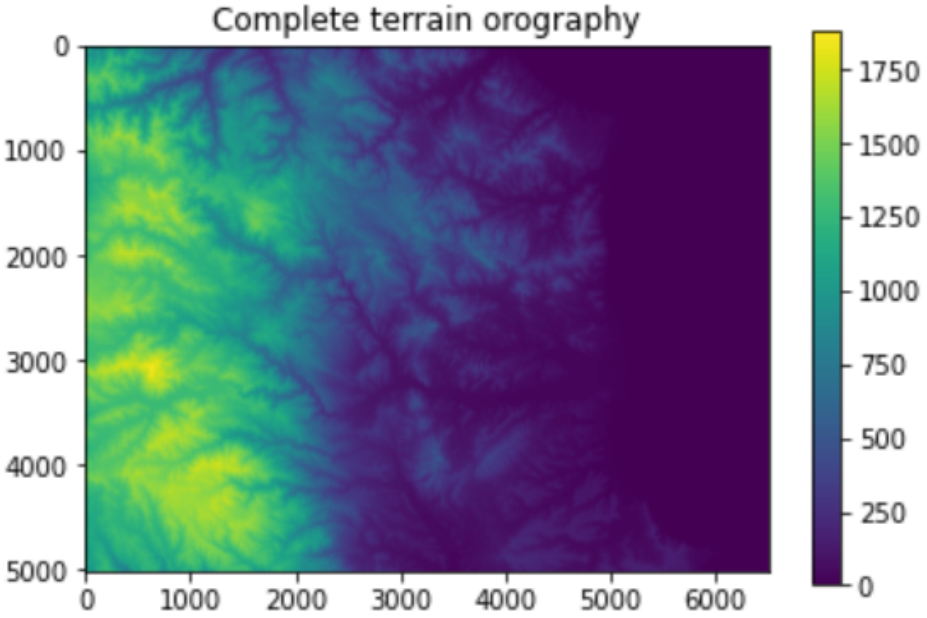

Figure 33: Orography of the terrain for all the subcases of the 1st case study

[32]This is considered as aereal distance

9.3.2 Subcase 1B

This case is quite similar with respect to the previous one, the starting and arrival points are still the same: A(35 ; 5) and B(10 ; 45), all the cost coefficients and penalties have not changed.

The only two differences tested are the features bounded to the motion of the target and the inclusion of the solar shadow penalties in the evaluation of the total movement cost. Both these features have been removed to test the software in a "faster" version.

9.3.3 Subcase 1C

This subcase has been specificately done to test the software in a "heavy" situation. The starting and final points have been changed to be way more distant from each other. In particular, the coordinates chosen are the following: A(0 ; 0) as starting point and B(180 ; 200) as final one.

To not reduce the distance from arrival, only the target motion has been disabled, meanwhile all the other features runs perfectly (including the sun shadow penalty).

9.3.4 Subcase 1D

This subcase is really similar to the one indicated as 1A, but with an important difference. The starting and arrival points are still the same: A(35 ; 5) and B(10 ; 45), all the cost coefficients and penalties have not changed. To evaluate the most complex case, the motion of the target has been used, and also the solar optimization.

The only difference in this subcase is the hardware used to run it: instead of the Asus laptop with a processor Intel Core i7, and a computational power of 16 Gb of RAM memory, this time it has been used a Raspberry Pi 3, Quad Core 1.2GHz Broadcom BCM2837 64bit CPU 1GB RAM.

9.4 Data output

9.4.1 Subcase 1A

The road reached has been selected through an infinite amount of possibilities, but the time elapsed is still reduced. In the following version of the software will be shown the differencies changing some parameters or features.

The software has been able to complete the task, reaching the final point in a time of 96.6 seconds.

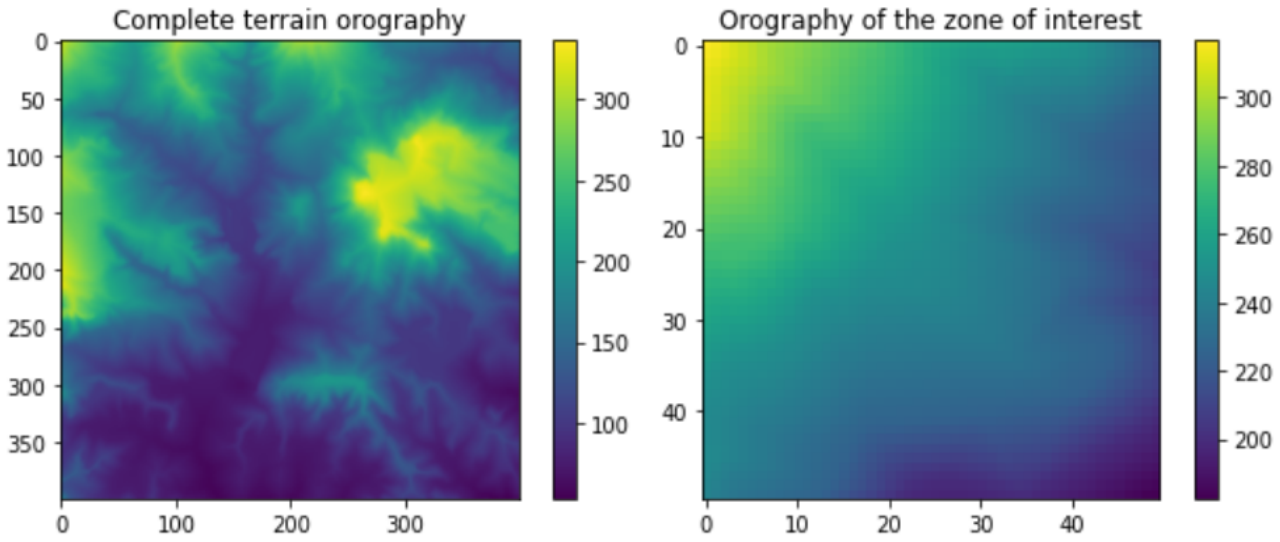

Figure 34: Orography of the terrain for the subcase 1A

As it can be seen from the output, the real roads chosen by the local optimization is slightely different than the one hypotized in the global optimization. This is given by the fact that the global analysis is generic and approximative, and has only the aim to give a constraint for the local optimization, without selecting it a priori.

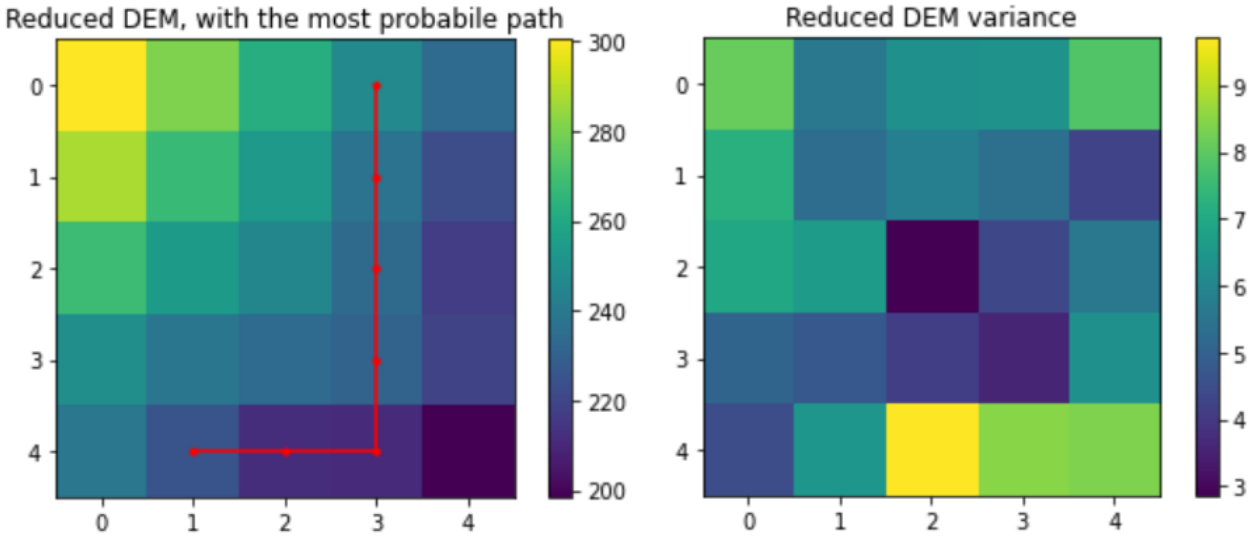

Figure 35: Conctracted version of the DEM matrix

It is important to notice that the software has elaborated a path that is slightly less than half a kilometer, in a time of about 1.5 minutes. This results is totally accettable since it is related to very slow rovers. To reduce further the time needed, other options have been evaluated in the following subcases.

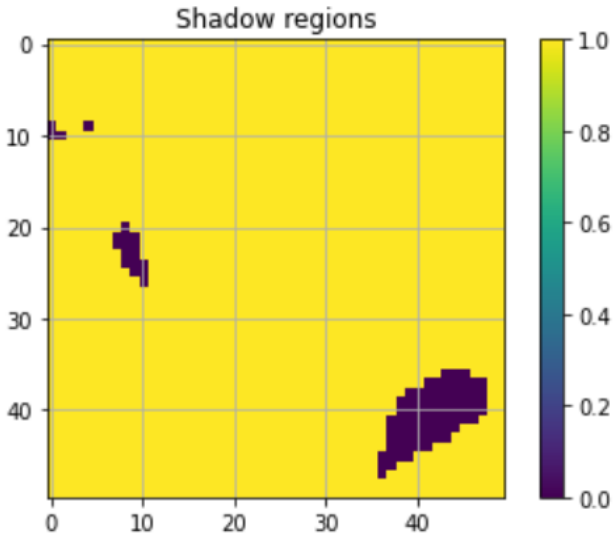

Figure 36: Shadow and illuminated region of the terrain

Note that in the complete version, this software can evaluate the path with a speed[33] of less than 5 m/s, or 18 km/h, so every rover travelling at a lower speed can be guided by this software. Note also that in a reduced version of the software[34], this speed can even increase.

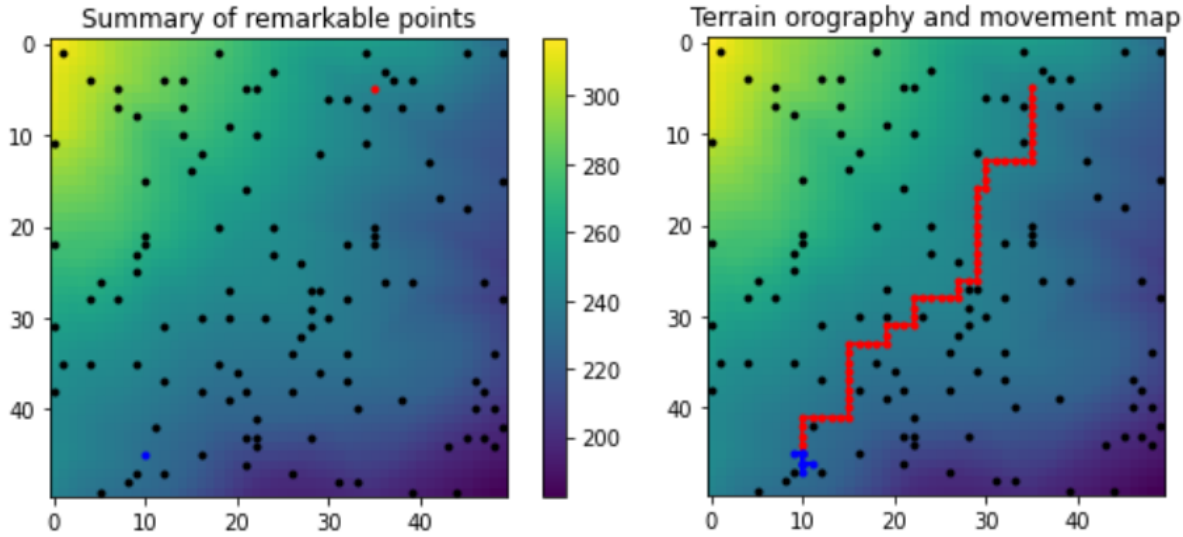

Figure 37: Summary of the main output elements

[33] In this case the speed is calculated as path length divided by time needed for calculation

[34] i.e. See subcase 1B, some part of the software can be evaluated before the rover starts its motion

9.4.2 Subcase 1B

This case is quite similar with respect to the previous one, with the same starting and arrival points, but with a lighter code, since the features of shadow penalization and target movements are no more available. The disposition of obstacle is different with respect to the subcase 1A because they're chosen randomly with every restart of the software.

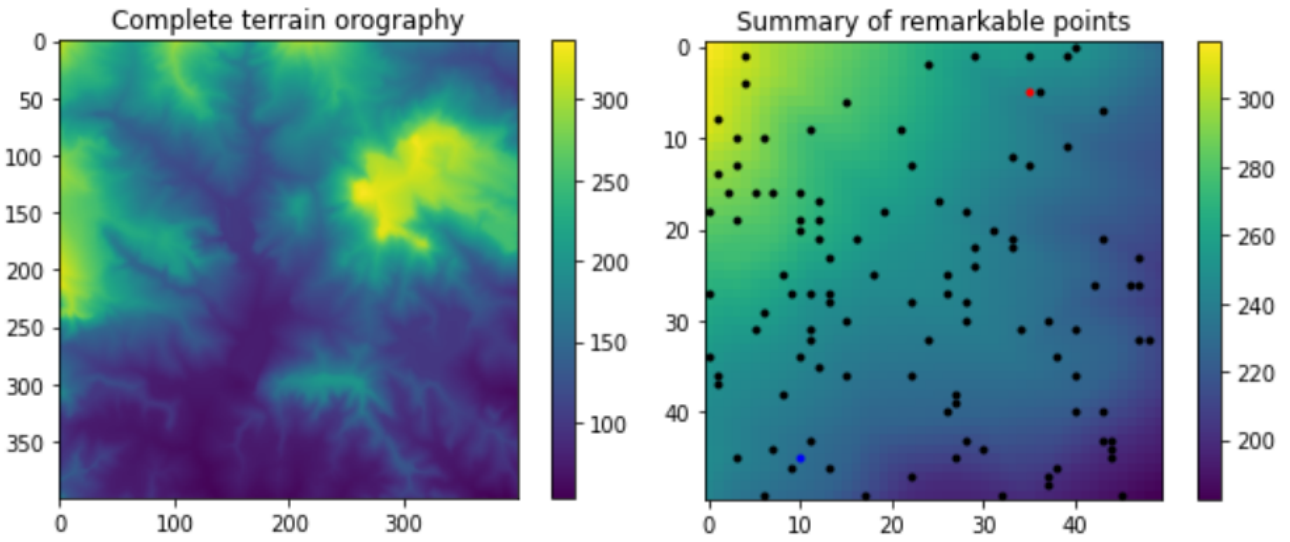

Figure 38: Orography of the terrain for the subcase 1B

The result is similar with respect to the previous one, but in this case the time elapsed is notably reduced. The total time elapsed is about 86.5 seconds, with a reduction of about 10.4% of the initial time needed.

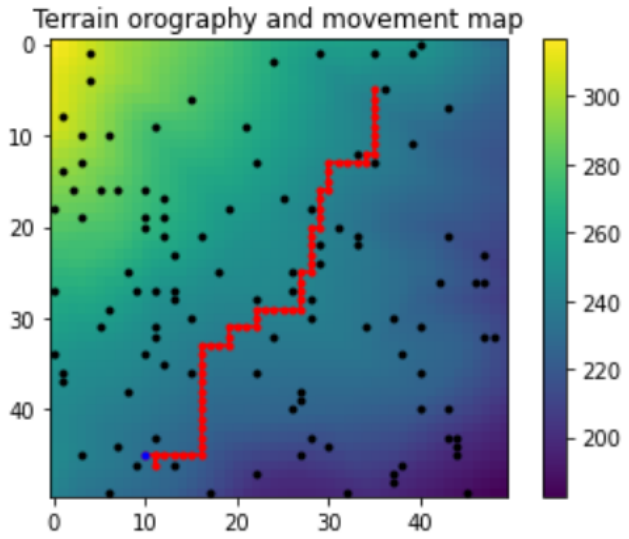

Figure 39: Summary of the main output elements

As it can be seen, in this subcase the terrain is indicated as totally subjected to sunrays (the value associated to 1 means presence of sunlight)

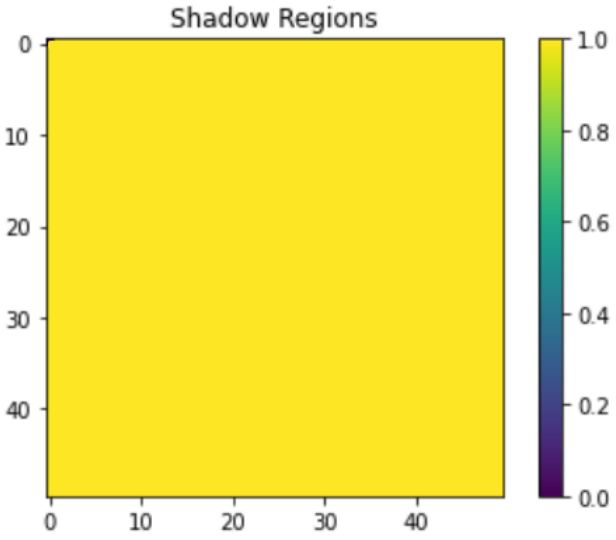

Figure 40: Shadow and illuminated region of the terrain

9.4.3 Subcase 1C

The third subcase of the first case study it is based on the evaluation of a path with a long distance target. For this purpose it has been chosen A(0 ; 0) as starting point and B(180 ; 200) as final one. The total distance is about 269 pixels, or 2.69 kilometers. In this case the presence of target motion was not necessary, so it has been disabled temporarly.

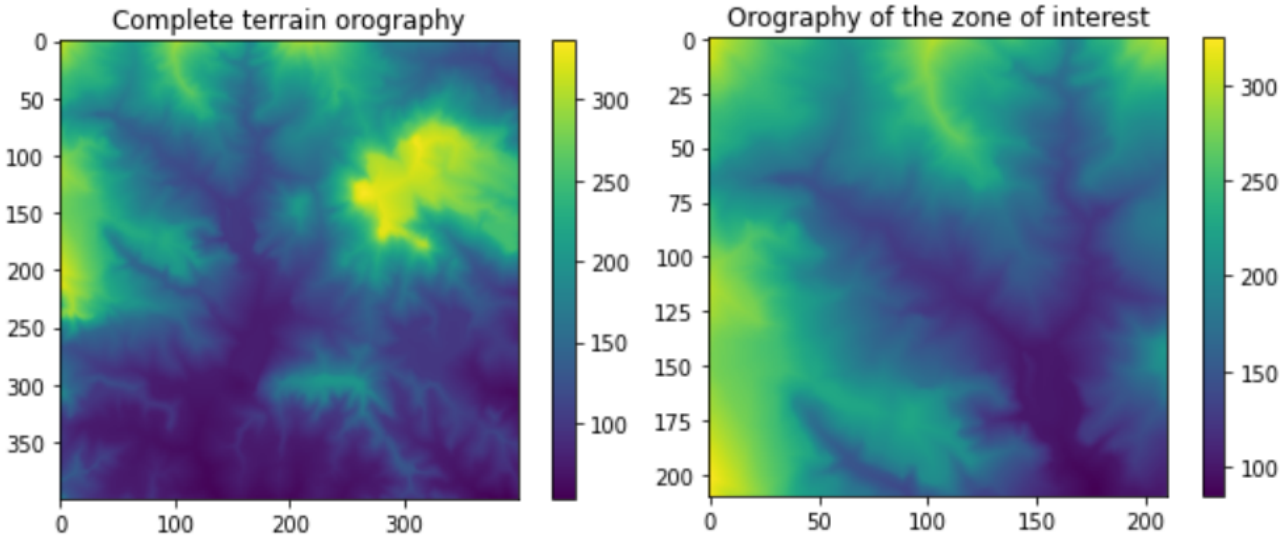

Figure 41: Orography of the terrain for the subcase 1C

Once runned, the code obtained the final path in a total time 635.2 seconds. This means that the speed of calculation allows to obtain informations regarding the best path with a virtual speed of 4.3 m/s or 15.5 km/h, with a result that is similar to the subcase 1A, even if the dimensions of the ground matrix were considerably bigger.

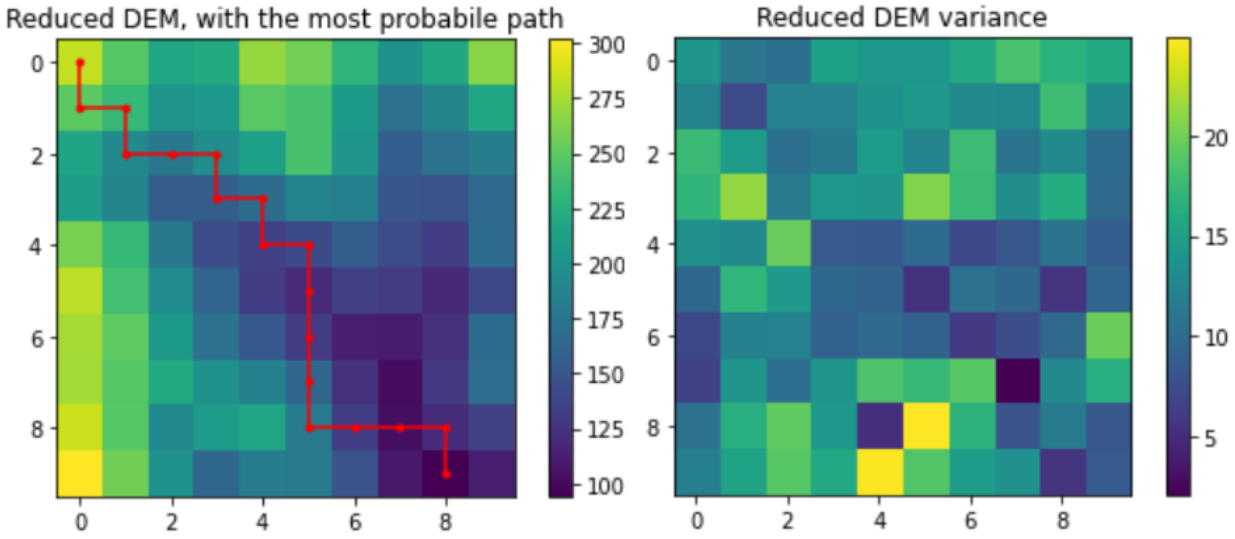

Figure 42: Conctracted version of the DEM matrix

In this (sub)case study, it has been selected a subdivision in 10x10 quadrants in the contracted DEM, to increase the precision in path evaluation. This compromise the initial performance to evaluate the global path, but it is also worthy in terms of energy saving while evaluating every single movement in the local optimization.

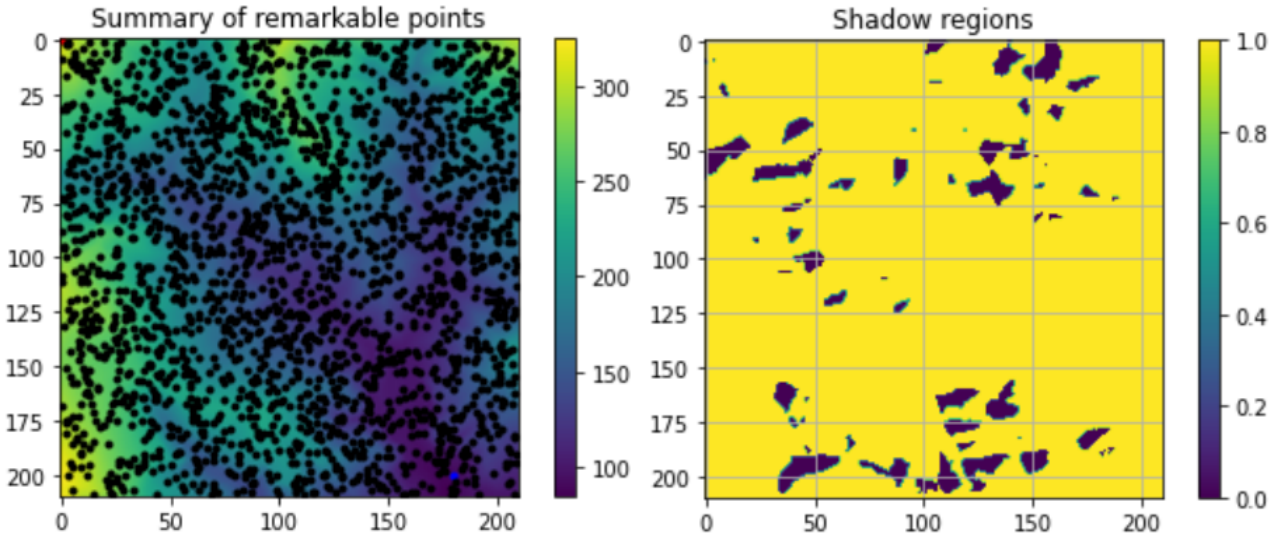

Figure 43: Shadow and illuminated region of the terrain

In the last figure is it possible to notice that the presence of shadow region is elevated and well distributed along the terrain, and also the obstacles are present in an elevated number on the soil. This guarantee the perfect functioning of the software in case of long distance and uncertain terrains.

The last graph shows the path evaluated from the starting point until the target arrival (the blue point indicate the target)

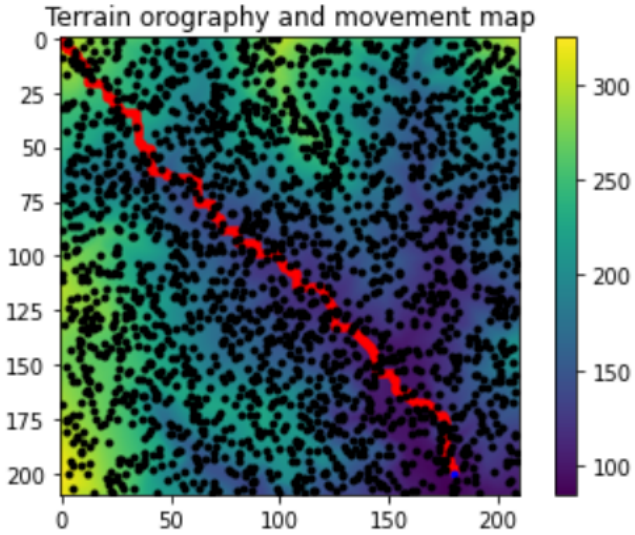

Figure 44: Summary of the main output elements

Subcase 1D

The relevance of this subcase study is bounded to the possibility to evaluate an optimized path also relying on a device with a reduced computational power.

For this reasons, this case study has the same exact features of the subcase 1A, in every details[35], but this time the software has been runned on a Raspberry Pi 3, Quad Core 1.2GHz Broadcom BCM2837 64bit CPU 1GB RAM.

The total time needed for this case is exactly 161.2 seconds, this means only 70% more with respect to the 1A subcase, even if the number of steps evaluated were similar.

The final result has been found after 67 different moves, as indicated in the output screenshot, notice that the number that classify the kind of movements are distinguished as: 1 (up), 2 (down), 3 (right), 4 (left).

```
New altitude=  223.0317
kind of movements:  [1, 1, 1, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4, 1, 1, 1, 1, 1, 4,
 1, 4, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1, 4, 4, 4, 4, 1, 1, 1, 4, 4, 4, 1, 1, 1, 4, 4
, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 1, 2, 4]
Time needed for path evaluation is about  161.2 seconds
```

Figure 45: Output of the software for the case 1D

This subcase has a big relevance in real application of this software for commercial purposes, because it demonstrate the possibility to run the software in devices that are way less efficient than a normal laptop.

[35] Apart from the features that are selected randomly, like obstacles

10 Second case study

10.1 Description

This second case study has enhancement in the code that makes it more suitable for complex region of work.

Differently from the first case study, in this variation of the software it has been analyzed a kind of terrain with strong domain limitations, not only in a discreet form (like single obstacles scattered on the ground) but entire zone of space that are not allowable for geographical reasons.

The zone selected is located in the southern zone of the world, in particular is an Antartica region. This zone has a peculiar orography with smooth changes, and it is perfect to test the main ability of the software in case of huge obstacles that limit its path. The resolution of the map is 8 meters, and the data are obtained with the satellites of the Profumo Constellation.

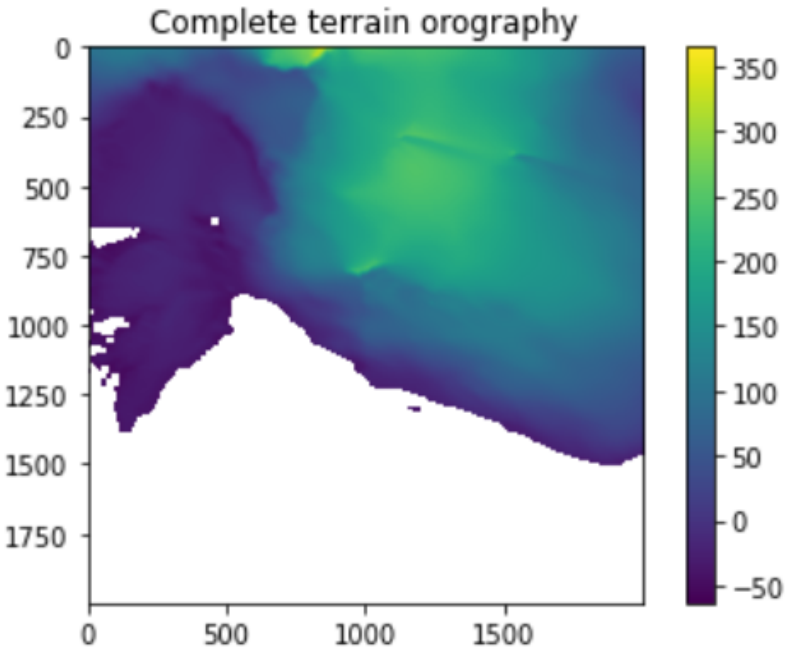

Figure 46: Orography of the terrain for all the subcases of the 2nd case study

The main features tested in this case study are the following:

- Ability to implement and make it work different cost functions, that include the cost based on the difference of altitude (or terrain slope), penalty for the presence of shadow, penalty for obstacles and penalty for external events.
- Ability to analyze the global area considered and obtain information from it, considering the terrain variance, the average altitude of the ground, the density of uncrossable obstacles, and similar data.
- Ability to select a particular road depending on the cost associated, after having analyzed the possible road and the relative feasibility, and chose a solution that should be a good compromise between feasibility and optimization.
- Ability to create and then identify the obstacles that can limit its road, both in a local and global perspective, with a penalty-like strategy.
- Ability to bound the global and local optimization, to increase the possibility to reach the target and reducing the computational power.
- Ability of the software to work in presence of very long distances, without losing the contact with a reachable road.

Note that, even if this case study has been hypotized to control wheter if the rover is able to bypass some large obstacle, is not meant to be a software able to solve a maze, but just a road that is more complex and articulated than a standard one.

10.2 Software code variations

The case study hereby presented uses a more complex version of the software, that can been considered similar to the basic one but with some "enhances"

However, in this middle version there are some features that can be changed to adjust the results, and/or to show the differences between the modalities of use.

With respect to the first case, it has been given way less importance to the energetic optimization, rather than the actual feasibility of the movement. This means that the road could even be not optimized with respect to the energetic point of view, rather than guarantee the ability to reach the target point. It may seems trivial, but since this case study has been organized to be used near domain border (e.g. near the sea), it is way more important to be able to be sure that the rover do not touch the water, to avoid losing it. On the other side, having a path that is so near a border may impact the possibile roads to rely on, and this may affect the energetic performances of the rover (that, however, in this case study are secondary).

Here are some characteristics that has been changed in the following try:

- The ability of the software to evaluate the shadow presence on the ground, and include it in the cost function
- The possiblity of the target to move from its initial location and have a generic motion near the domain border
- Distance between the starting and final points

10.3 Data input

For this second case study, it has been used a DEM with a selected size equal to 2000x2000 pixels, then reduced to only 200x200 pixels for practical reasons.

The data inserted as input starts with the remarkable points: the starting one is set equal to A (65 ; 200), while the position of the target has been decided as B (150 ; 25). These two points have been chosen not only for their elevated distance, but also because they're both near a huge unavailable zone, the Antartica Sea. For this reason, their distance of about 194 pixels in aereal distance corresponds to a bigger one due to a more complex path chosen.

In this version of the software it has been chosen to don't implement the target motion, because it could have lead to a simplier case[36].

Regarding the optimization based on the presence of sunrays on the ground, it has been implemented, but being the terrain a part of Antartica it is easy to notice that the altimetry plan has low level of altitude and reducecd slope of the terrain. For this reason, all the paths are subjected to presence of the Sun. The values used for the Sun inclination are still equal to $\alpha = \frac{\pi}{10}\mathrm{rad}$ and $\beta = \frac{\pi}{4}\mathrm{rad}$, as it were in the previus case study.

The presence of casual obstacles has been implemented, but with the limitation to locate them casually only on the shore, and not in the open sea, to reduce the time needed to place them.

[36]The target location has been chosen specifically to be near the sea, in case of motion it could be moved in the inner part of the shore, reducing the complexity of the problem

10.4 Data output

After having runned the script of the software, the result came in 420.4 seconds, necessary to evaluate a path of about 1.55 kilometers (equivalent to 194 pixels with size 8 meters each). This first data is useful to obtain a virtual speed of 3.7 m/s.

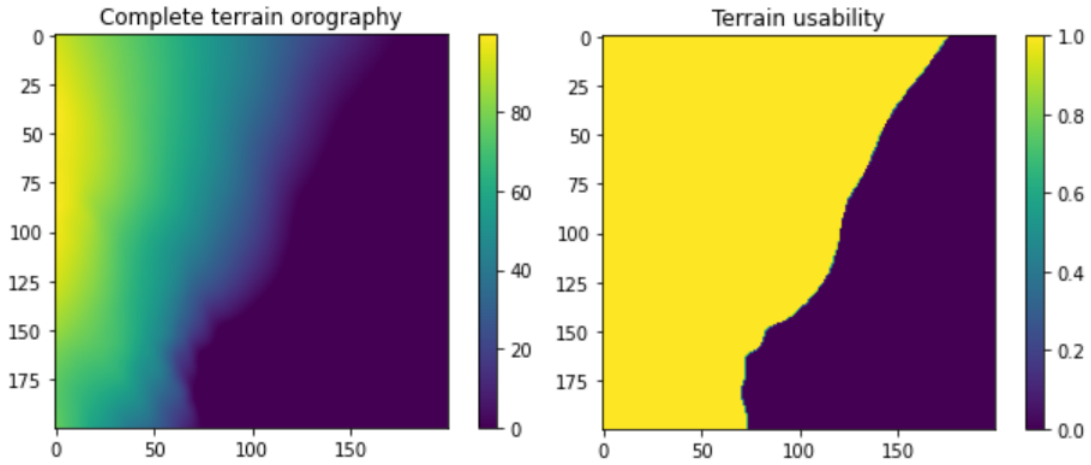

Figure 47: Orography of the terrain for the case study n.2

It is important to notice that in this case the terrain usability plays an important role in the optimization.

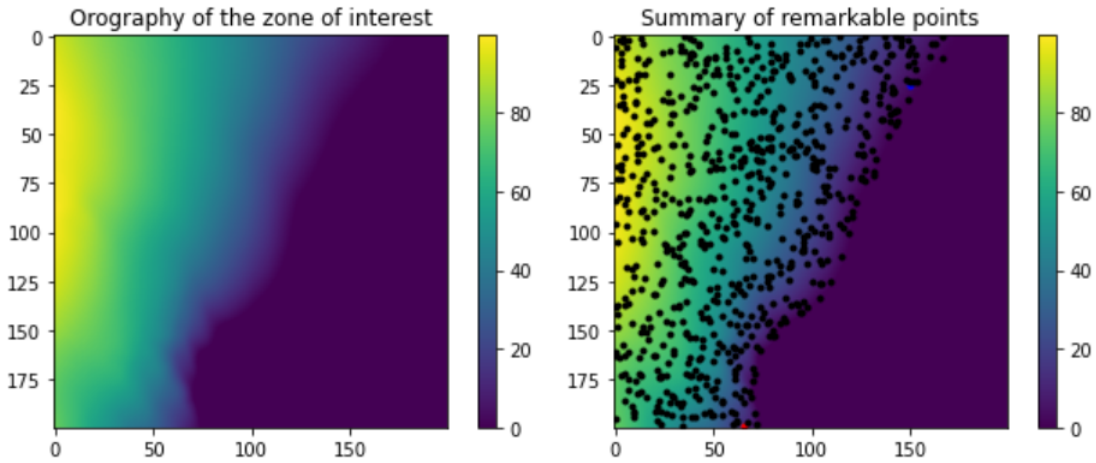

Figure 48: Summary of remarkable points

As it can be seen by the following graph, the evaluation of the presence of the Sun can be obtained with the angle equal to $\alpha = \frac{\pi}{10} rad$ and $\beta = \frac{\pi}{4} rad$. Due to the smoothness of the altrimetry, the whole terrain result illuminated by the Sun. (Remind that in the graph the value of 1 is associated at presence of Sun)

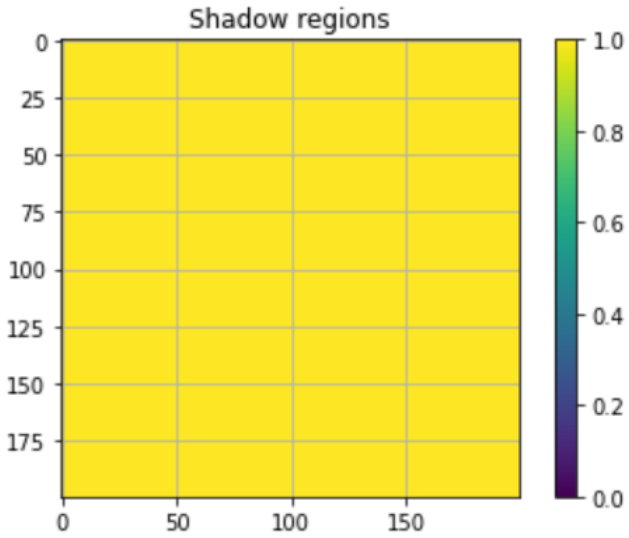

Figure 49: Shadow and illuminated region of the terrain

The low value of terrain variation can be also seen by the following graph that identify the terrain with its variance[37]. The value of variances are quite low, due to the smoothness of the ground.

[37]Obtained as standard deviation, from the average value of altimetry

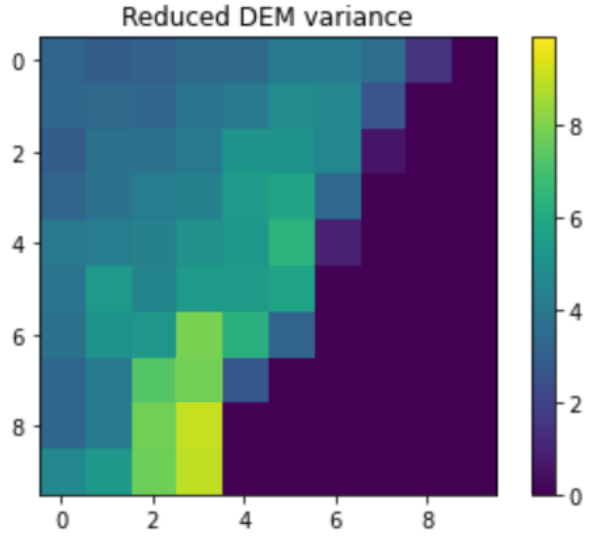

Figure 50: Summary of the main output elements

The last graph shows the path followed by the rover in the Antartica region. The motion of the rover goes through the shore to limit the altitude variation, but at the same time it stays away from the limit of the domain given by the presence of the water.

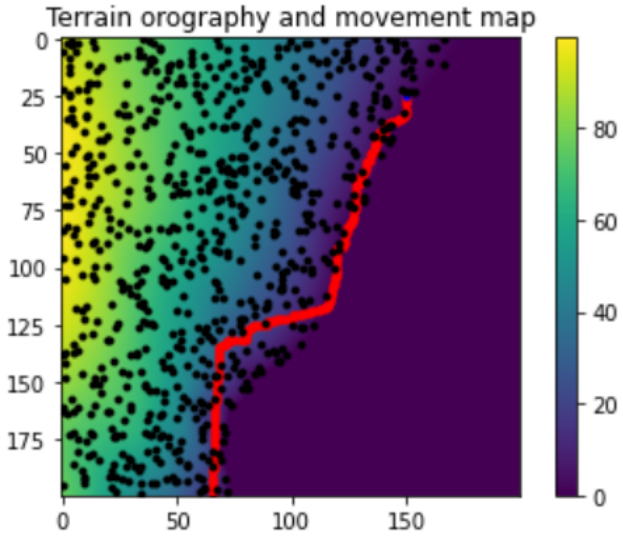

Figure 51: Summary of the main output elements

11 Third case study

11.1 Description

The third case study is the most complex and elaborated between all the ones presented in this thesis. It is an advanced version of the software, with small changes in the algoritms, to be adapted for a nautical purpose. To prove the robustness of this software, it has been made a version of it that is able to guide not only a rover, but also a ship. For this reason the data used are different with respect to the one implemented before, with meteorological data instead of altimetry data, to be able to insert an optimization based on the possibility of navigation.

For this reason, the zone of the Earth chosen for this case study is the "Cradle of civilization", the Mediterranean Sea.

Figure 52: Zone of interest for the 3rd case study

The orography of the zone of interest has a different function with respect to the ones used before, because in this case it is used only to determine whether if the terrain can be used for navigation or not. Although, the resolution of the map is much lower with respect to the previous cases, because otherwise would have been too much expensive from the computational cost point of view. This means that using this software in the real life requires an additional algorithm that is able to recognise the smallest obtacles like reefs, islets or other boats, to be able to achieve the autonomous navigation.

The main parameters that are used to identify the best route are mostly of aeronautical interest, like density of precipitation, cloud area fraction, speed of wind and waves.

All these parameters represent real data, and are taken from Profumo constellation. The different approach used for this case study implies the possibility to identify the starting point and the target as real places, particular cities that have been chosen to represent determinated challenges to be reached.

To increase the realisticness of this case study, it has been considered a change of meteorological condition every hour. For this reason the data imported have different value for each hour step. The total amount of time covered is 6 hours; the limited time is only a matter of limited complexity, but could be extended to a delta time that is virtually infinite).

11.2 Software code variations

The third case study it a version of the software that is more peculiar than the previous two versions. The necessity to use the water as a mean of movement implies the change of paradigma and a different optimization. The following elements represent all the main changes that has been done to implement this version:

- The usability matrix change “shape”: if the previus cases needed a terrain under the wheels, now it is the opposite. While identifying the kind of soil, it is necessary to consider “usable” all the pixels that previusly would have been considered as “unusable”, for obvious resons. This implies a change in the algorithms.
- The DEM contraction makes lower sense: considering a global optimization based on the average altitude and terrain variability is no longer a valid approach, because the only valid altitude for the terrain is equal to zero to allow the navigation. Otherwise, there is the need to obtain a discrimination between convenient or unconvenient points in the contracted DEM, so this will be represented by a mean value of the usability matrix.
- The data import is different: the local optimization is based on 4 different meteorological data: density of precipitation, cloud area fraction, speed of wind and waves.

These data are totally realistic, and are used to optimize the path.

- The cost functions can change with respect to time: Since the meteorological situation can change quickly, it is not possible to evaluate the route without considering the possibility of a variation of them. For this reason, it has been implemented an algorithm that update the meteorological condition every hour. In this way the informations are always up to date.
- The initial and final points are different: instead of indicate the initial and final points, it is possible to insert directly the name of the harbour city that shall be reached (or started from) and the software will find the coordinates.
- The target is no longer moving: The target motion has been disabled, because it is unlike to see a moving harbour. It could be still implemented in case of a moving target like a boat.
- The presence of shadow on the terrain is no longer relevant: since almost all of the ships do not have any solar array on the top, this feature has been removed. As said for the motion of the target, it would be still easy to implement again this algorithm in case of need.

11.3 Data input

11.3.1 Subcase 3A

The first case study analyzed is a classic version that is used only to analyze if the main features of the software are correctly working.

The data used as input for the initial and final points are the following:

- Initial point: A (110 ; 0) corresponding to the city of Valencia
- Final point: B (131 ;32) corresponding to the city of Algeri

These two cities have been chosen because they are not too distant, but at the same time they have a small obstacle between them.

Both in this and the next version, the cost coefficients have been selected as the reciproce of the maximum value of the parameters to be penalized. This make the penalization as equal as possible.

11.3.2 Subcase 3B

In this second subcase the features of the software are the same of the previous subcase 3A, but this time the initial and final points have been chosen to have a bigger obstacle between them (the Corse and minor islands). The starting point and the target selected are the following:

- Initial point: A (131 ; 32) corresponding to the city of Algeri
- Final point: B (65 ;102) corresponding to the city of Livorno

As previously said, the cost coefficients have been chosen as the reciproce of the maximum value of the parameters to be penalized.

11.4 Data output

11.4.1 Subcase 3A

This case study has given a result in about 52.3 seconds, this time is neessary to evaluate the road between two cities distant about 430 kilometers, that in this case study are the equivalent of 38 pixels. This low time is justified by the reduced resolution of the map used. with a more complex orography this time would increase. Notice that the starting city, Algeri (in blue) is separated by the arrival city (Valencia, in red) only by a small obstacle. The path planner is able to reach the target city avoiding the obstacle islands

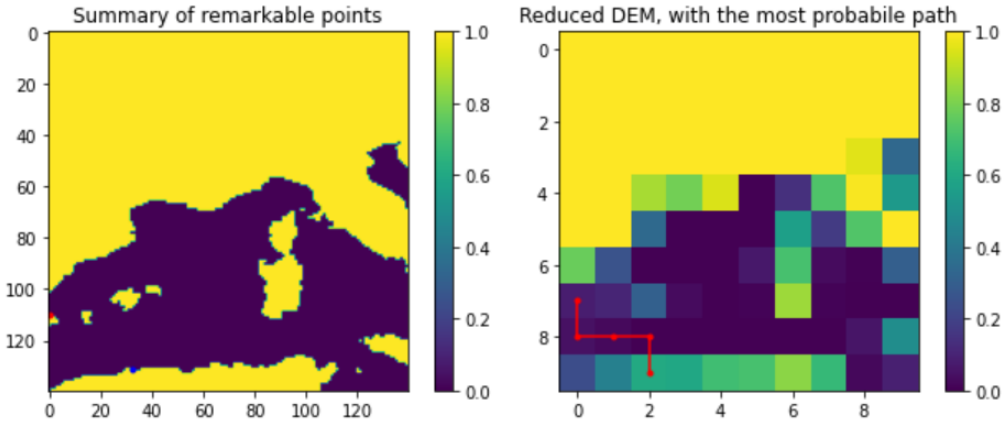

Figure 53: Orography of the terrain for the case study n.3A

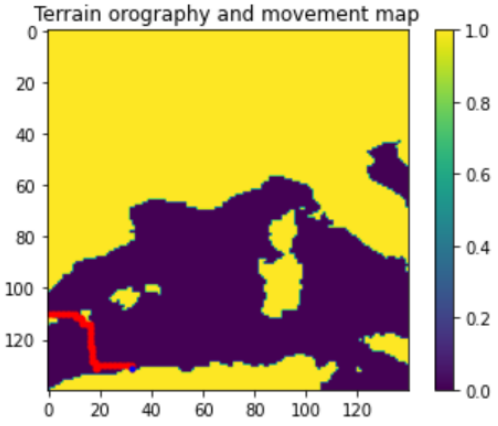

Figure 54: Terrain orography and movement map

These graphs can show quite well the distribution of the meteorological data along the zone of interest. Notice also that all the first three parameters are defined in all the map, while the last paramenter is only defined in the marine zone because it is the wave speed.

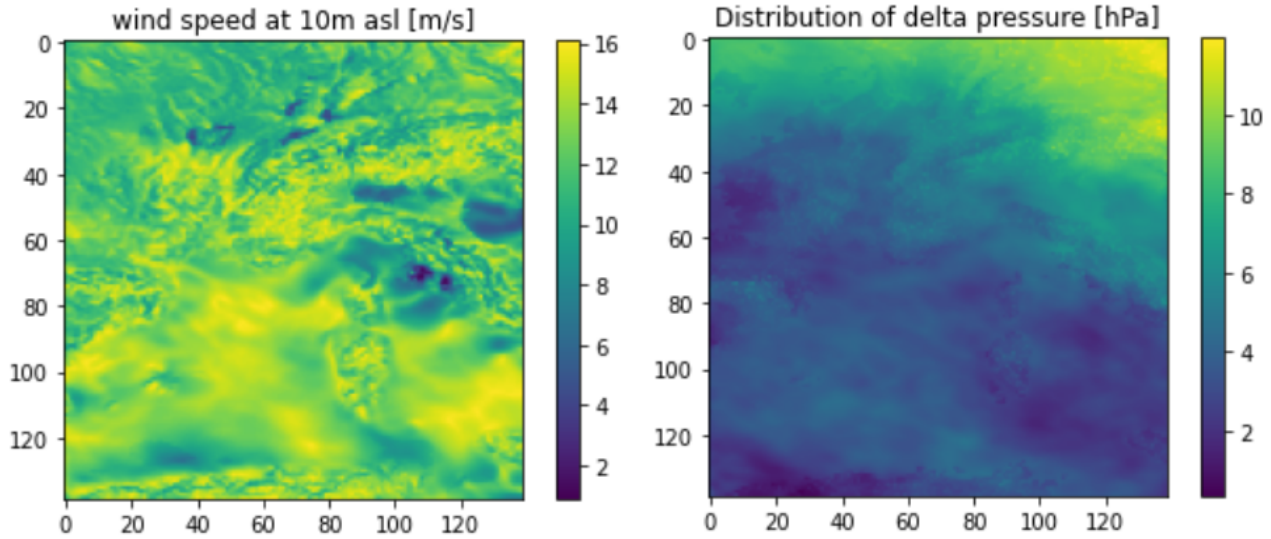

Figure 55: Meteorological data (1st part)

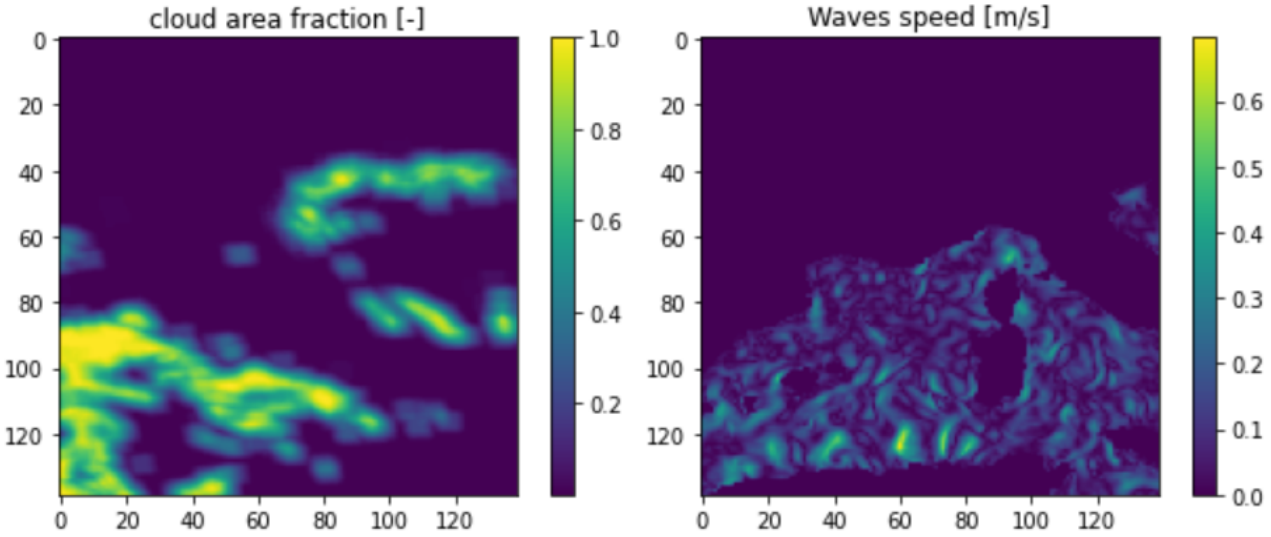

Figure 56: Meteorological data (2nd part)

11.4.2 Subcase 3B

This case study has given a result in about 175.6 seconds, this time is neessary to evaluate the road between two cities distant about 970 kilometers, that in this case study are the equivalent of 96 pixels. This low time is justified by the reduced resolution of the map used. with a more complex orography this time would naturally increase. Notice that the software is able to identify the best route optimizing the cost due to meteorological events, refreshing those values every hour to obtain updated data, and reaching the target without hitting any major obstacle.

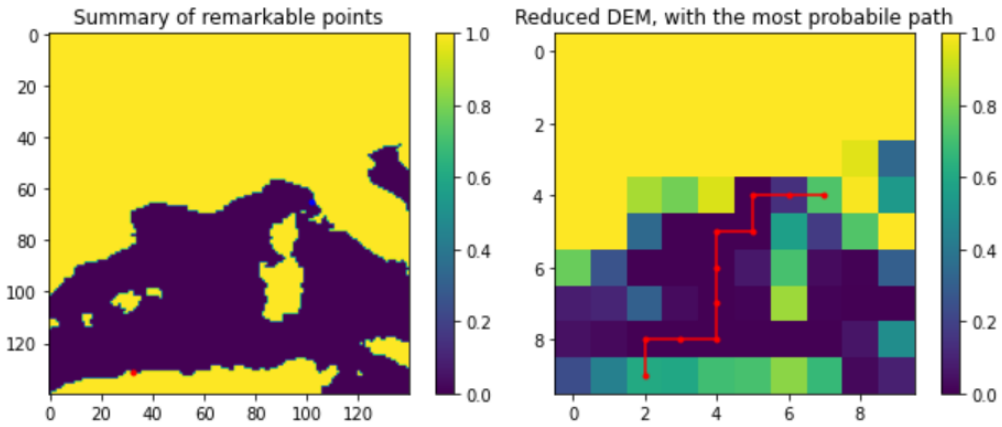

Figure 57: Orography of the terrain for the case study n.3B

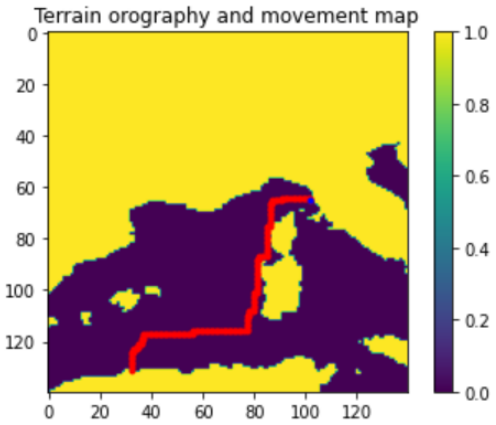

Figure 58: Terrain orography and movement map

Part IV

Future plans and conclusion

12 Future plans

The software realized is totally able to achieve the goal of finding a path that links two points in a determinated zone of a planet, using an energetic optimization.

This algorithm can work fine in many different situations, and is able to run also on a device with small computational power like the one used for lunar/martian rovers.

The work done so far is the result of a thorough research in the literature, and a huge analysis of the existing softwares. The main features of the algorithm can describe the environment and the possible behaves of the rover used, and the final result is complete and optimized.

As many things in the engineering world, this software does not have the will to be perfect and done, even if it works, but many things can be added or modified to make this software more realistic, practical or precise.

Hereby, in the following chapters, are reported some modification at the code that can improve the overall performances of the single algorithms or of the final result achieved.

12.1 Sun inclination and variable shadow penalty

For the cost evaluation, it has been set a penalty value for the presence of the shadow on the surface of the ground. The presence of shadow is considered as a boolean value, with the possible results of “True” or “False”.

This value can be updated considering a float value depending on the percentage of sunlight (or shadow) present in the considered zone of space. Note that in this case there should be a theshold due to the minimum amount of solar radiation needed to obtain energy from the solar arrays; otherwise, instead of a constant penalty it could be used a coefficient to be multiplied by the fraction of shadow on the terrain.

Regarding the accuracy in the determination of the shadow placement, the angles used for the determination of the position of the sun in the sky are considered totally constant with the passing of time, while for a long lasting motion could be useful to calculate the displacement law of the shadow to analyze how it interfers with the rover motion. However, notice that this can interfers with the computational efficiency.

Moreover these values are considered in a detached way with respect to the reality: these values are totally hypotized[38], but they could be realistic since the the location of the motion is totally known, not depending on the operational planet. For this reason it could be obtained automatically the displacement law for the motion of the Sun in the sky.

12.2 Cost function based on soil characteristics

The software realized can achieve its goals optimizing the route between two different points, but there are many efforts that can be done to make it more realistic: one fo them is the capability of extimation of the real penalty to be assigned to a movement depending on the kind of soil considered.

For this purpose, a wheel moving over a rocky terrain will behave differently with respect to a similar one over an icy or moisty terrain, and for this reason those kind of soil should have a personal penalization.

This has not been included in the present software just for data availability: just using the DEM altimetry is not possible to obtain the real kind of terrain, which however can be obtained from the relevations of other devices.

This feature shall be considered while dealing with a rover that moves on a terrain that has deeply different features, like an alternation of an icy and rocky soil that is typical of the (sub)artic regions.

Note that this addition to the software should not increase significantly the complexity of the code, nor the computational time, since this data could be considered as a constant during the journey of the rover[39].

12.3 Implementation of diagonal moves

While considering a path made by a rover, is it difficult to have a frequent change of direction, moreover with turns of 90 degrees. However, this is the only kind of turn used to change direction while moving, and it is not casual. Apart from the obvius reasons due to the characteristics of the cartesian discretization of space, there is also a more technical reason: almost always, the best choice for a path between two point is given by the simplest path of all, that is the one that links the initial and final point in a linear way. In this case,

[38] The peculiar values chosen for the sun inclination angles are given by the computational necessity to have both shadow and sunlight zones in the considered DEM

[39] under the assumption that the morphology of the terrain should not change in the delta time considered

the optimization gets totally a different meaning, as "how much a single road cost more than the straight one", and the optimization would be very different. For this reason, the diagonal movements are deliberately ignored just to allow the software to work in cases that are not the simples or the most obvious ones.

However, for a real path planning software, it is not necessary to complicate everything just for science's sake, altough it can be introduced a kind of movement previously ignored, the diagonal one.

The movement can be considered both with an angle of $\frac{\pi}{4} + k\frac{\pi}{2}$ radians, that corresponds to an odd multiple of 45 degrees, or otherwise a generic angle without any kind of discretization. Let's just notice that the second option is not (easily) supported by this kind of cartesian coordinates.

12.4 Realistic relevation of obstacles

The value of terrain usability has been evaluated in different ways, but mostly in a randomic way for a better representation of the unpredictible location of the obstacles.

However, with a more precise data set, is it possible to implement an algorithm able to recognize autonomously the presence of obstacles depending on different factors like an eccessive slope of the terrain[40], or the change in soil composition, and so on.

This analysis can be performed specifically with more realistic data, taken from sensors or devices, connected to the rover. In this way it could be possible to analyze also the real time data take from the surface, with a significant increase in softare precision and reliability.

Notice that this kind of approach is already in use for this software but only with the third case study, due to the quantity of realistic data.

12.5 Boundary escape development

Another possible improvement in this software can be done implementing a different concept of final point. Instead of using a single pixel as a target point, it could be indicate a whole line of pixels that can be reached. This kind of approach is called "boundary escape" and can be useful to reduce the amount of energy required to complete the path. This modification, however, can be dangerous for the computational side because the possible roads to be analyzed would increase with the size of the boundary.

[40] Both positive or negative, even with different margins of value

13 Conclusion

As it has been shown, this software has been able to efficiently guide a rover (or a ship) upon an uncertain terrain, with the help of a satellite orography and few other data. This has been done efficiently from the computational cost point of view, to match the necessity of the real martian/lunar rover that now and in the future will be running on the surface of other planets.

It has been tested in many ways, 7 of them are reported condensated in 3 case studies, with all the results shown graphically.

This software has been planned to reduce the physical cost of the movement and at the same time reduce the computational cost, and it has been demonstrated to be both robust to variation and versatile for other purposes. It could be used for many different applications: from the space sector to the aeronautical one, and even as a base for an autonomous drive software for cars or bike (even if with some modifies, like the third case study).

The code of this software has also been tested upon a digital twin, a low computational power device that can simulate the performance of the OBDH system of a rover, to guarantee the use of this software for space application, as it was settled as one of the fundamental features of the software. The result has been great, as the code runned with only a 70% of time delay with respect to the other simulation done by a normal laptop (case studies 1A and 1D).

This master thesis has come to an end, but this software is far to be considered concluded because, as previously hypotized, there are many innovations and improvements that can be implemented to make it even more accurate, reliable and efficient.

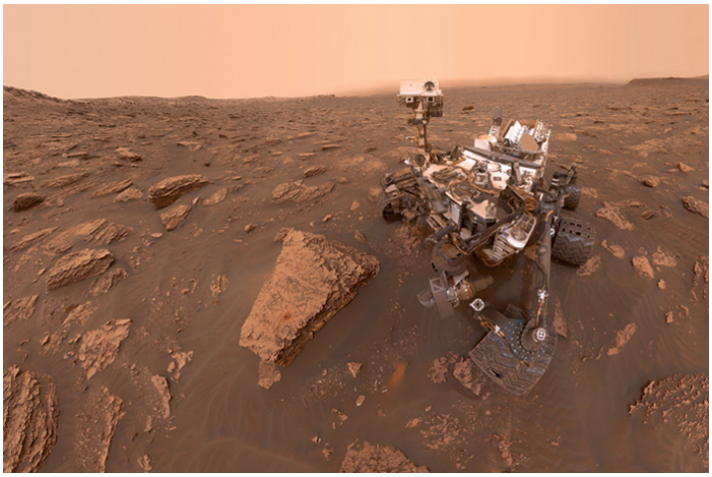

Figure 59: Example of space application

Ringraziamenti

Questi ultimi anni al Politecnico sono stati particolarmente intensi e impegnativi, ma hanno anche saputo regalarmi soddisfazioni per tutte le sfide superate; è un capitolo che si sta chiudendo per lasciare spazio a qualcosa di nuovo, e a posteriori sceglierei nuovamente questo percorso mille volte. I miei ringraziamenti vanno a tutti coloro i quali, con pazienza, mi hanno supportato in questi tanti(ssimi) anni di fatica e sacrifici, accompagnandomi nel mio percorso.

In primis ringrazio i miei professori universitari e l'Alma Mater, che hanno saputo tramandare concetti preziosi, mi hanno reso una persona più razionale e mi hanno trasformato da un ragazzo ad un Ingegnere. Ringrazio in particolare il professor Mauro Massari, mio docente per due diversi corsi e relatore di questa tesi, per i preziosi insegnamenti e il tempo dedicatomi.

Ringrazio anche i miei referenti dello stage in Telespazio, Nicola de Quattro e Filippo Iodice, per il supporto tecnologico, logistico e morale durante lo sviluppo della tesi.

Un ringraziamento va anche a tutti i miei amici, vicini e lontani, che mi sono stati accanto nel corso degli anni, inclusi tutti i ragazzi e gli ingegneri conosciuti nei progetti di Skyward e Polispace. Ovviamente un ringraziamento speciale va a Edoardo per la pazienza con cui mi sopporta da poco meno di un quarto di secolo.

Ricordo e ringrazio anche tutti i miei parenti, e soprattutto la mia famiglia per essermi stata vicina in ogni occasione e per avermi aperto le porte del mondo. Sarò sempre riconoscente ai miei genitori per come mi hanno cresciuto e sono molto orgoglioso di essere loro figlio. Ovviamente ringrazio anche mio fratello Stefano e mia sorella Mariachiara, che mi sostengono da sempre.

L'ultima persona che voglio ringraziare è in realtà quella che per me viene prima di tutto, e alla quale è dedicata questa tesi. Sara, la mia futura moglie, mi è stata accanto durante questo cammino universitario dandomi la forza di proseguire e facendomi diventare una persona migliore; lei è sicuramente la parte migliore di me, e non potrei desiderare una persona diversa con cui passare il resto della vita.

References

[1] Winter, Matthias & Barclay, Chris & Pereira, Vasco & Lancaster, Richard & Caceres, Marcel & Mcmanamon, Kevin & Nye, Ben & Silva, Nuno & Lachat, Daisy & Campana, Marie. (2015). ExoMars Rover Vehicle: Detailed Description of the GNC System. Link

[2] J. Carsten, A. Rankin, D. Ferguson and A. Stentz, "Global Path Planning on Board the Mars Exploration Rovers," 2007 IEEE Aerospace Conference, 2007, pp. 1-11, doi: 10.1109/AERO.2007.352683. Link

[3] E. Gat, M. G. Slack, D. P. Miller and R. J. Firby, "Path planning and execution monitoring for a planetary rover," Proceedings., IEEE International Conference on Robotics and Automation, 1990, pp. 20-25 vol.1, doi: 10.1109/ROBOT.1990.125939. Link

[4] Rekha Raja, Ashish Dutta, K.S. Venkatesh, New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover, Robotics and Autonomous Systems, Volume 72, 2015, Pages 295-306, ISSN 0921-8890, doi: 10.1016/j.robot.2015.06.002 Link

[5] Tsitouridis, George (2020): Terramechanics and soil–wheel interactions for road vehicle applications. Loughborough University. Thesis. https://doi.org/10.26174/thesis.lboro.11893947.v1 Link

[6] M. Ono, T. J. Fuchs, A. Steffy, M. Maimone and J. Yen, "Risk-aware planetary rover operation: Autonomous terrain classification and path planning," 2015 IEEE Aerospace Conference, 2015, pp. 1-10, doi: 10.1109/AERO.2015.7119022. Link

[7] R. Diankov and J. Kuffner, "Randomized statistical path planning," 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 1-6, doi: 10.1109/IROS.2007.4399557. Link

[8] X. Liu, D. Zhang, H. Yan, Y. Cui and L. Chen, "A New Algorithm of the Best Path Selection Based on Machine Learning," in IEEE Access, vol. 7, pp. 126913-126928, 2019, doi: 10.1109/ACCESS.2019.2939423. Link

[9] Michael Otte, *A Survey of Machine Learning Approaches to Robotic Path-Planning*, 2008, University of Colorado at Boulder Link

[10] Ee Soong Low, Pauline Ong, Kah Chun Cheah, Solving the optimal path planning of a mobile robot using improved Q-learning, Robotics and Autonomous Systems, Volume 115, 2019, Pages 143-161, ISSN 0921-8890, doi: 10.1016/j.robot.2019.02.013. Link

[11] S. Chhaniyara, C. Brunskill, B. Yeomans, M.C. Matthews, C. Saaj, S. Ransom, L. Richter, Terrain trafficability analysis and soil mechanical property identification for planetary rovers: A survey, Journal of Terramechanics, Volume 49, Issue 2, 2012, Pages 115-128, ISSN 0022-4898, doi: 10.1016/j.jterra.2012.01.001. Link

[12] Mahmoud Tarokh, Hybrid intelligent path planning for articulated rovers in rough terrain, Fuzzy Sets and Systems, Volume 159, Issue 21, 2008, Pages 2927-2937, ISSN 0165-0114, doi: 10.1016/j.fss.2008.01.029. Link

[13] Meng Wang and Liu, "Fuzzy logic based robot path planning in unknown environment," 2005 International Conference on Machine Learning and Cybernetics, 2005, pp. 813-818 Vol. 2, doi: 10.1109/ICMLC.2005.1527055. Link

[14] Paul Tompkins, Anthony Stentz, David Wettergreen, Mission-level path planning and re-planning for rover exploration, Robotics and Autonomous Systems, Volume 54, Issue 2, 2006, Pages 174-183, ISSN 0921-8890, doi: 10.1016/j.robot.2005.09.027. Link

[15] S. Upadhyay and A. Ratnoo, "Continuous-Curvature Path Planning With Obstacle Avoidance Using Four Parameter Logistic Curves," in IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 609-616, July 2016, doi: 10.1109/LRA.2016.2521165. Link

[16] Chunhui Zhou, Shangding Gu, Yuanqiao Wen, Zhe Du, Changshi Xiao, Liang Huang, Man Zhu, The review unmanned surface vehicle path planning: Based on multi-modality constraint, Ocean Engineering, Volume 200, 2020, 107043, ISSN 0029-8018, doi: 10.1016/j.oceaneng.2020.107043. Link

[17] The *Cameras on the Mars 2020 Perseverance Rover*, March 2020, Jet Propulsion Laboratory. Link

[18] Bölter, Manfred, Hans-Peter Blume, and Holger Wetzel. "Properties, formation, classification and ecology of arctic soils: Results from the Tundra Northwest Expedition 1999 (Nunavut and Northwest Territories, Canada)." Polarforschung 73.2/3 (2006): 89-101. Link

[19] Perko, Howard & Nelson, John & Green, Jacklyn. (2006). Mars Soil Mechanical Properties and Suitability of Mars Soil Simulants. Journal of Aerospace Engineering - J AEROSP ENG. 19. 10.1061/(ASCE)0893-1321(2006)19:3(169). Link

[20] S. Upadhyay and A. Ratnoo, "Continuous-Curvature Path Planning With Obstacle Avoidance Using Four Parameter Logistic Curves," in IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 609-616, July 2016, doi: 10.1109/LRA.2016.2521165. Link